```
//XALGCL JOB  (T193***,914,1,2,200),' ED SATTERTHWAITE ',MSGLEVEL=1      JOB  18
//JOBLIB  DD   DSNAME=T123.PLLIB,UNIT=2314,VOLUME=SER=SYS11,          X   2.
//             DISP=(OLD,PASS)                                            3.
//PL36C EXEC   PGM=PL360                                                  4.
//SYSPRINT DD  SYSOUT=A,DCB=BLKSIZE=133                                   5.
//SYSGO    DD  DSNAME=&LOADSET,UNIT=2314,SPACE=(CYL,(3,1)),           X   6.
//             DISP=(MOD,PASS),DCB=(LRECL=80,BLKSIZE=3200,RECFM=FB)       7.
//SYSIN    DD  *                                                          8.
IEF236I ALLOC. FOR XALGOL   PL360
IEF237I JOBLIB   ON 434
IEF237I SYSGO    ON 330
IEF237I SYSIN    ON 00C
```

```
01 0000  C0 0000  0001  GLOBAL PROCEDURE LOADCARD(R14);                                                    10.
01 000C  C0 0000  0002   BEGIN                                                                            11.
06 0000  00 0000  0003                                                                                    12.
06 000C  00 0000  0004     COMMON BASE R11;                                                               13.
06 0004  C7 0000  0005      ARRAY 256 INTEGER ESDTABLE;  COMMENT GLOBAL SYMBOL DICTIONARY;                14.
06 0004  07 0400  0006       ARRAY 64 LOGICAL NAME1  SYN ESDTABLE, NAME2  SYN ESDTABLE(4);               15.
06 0004  07 0400  0007       ARRAY 64 INTEGER ADDR   SYN ESDTABLE(8);                                    16.
06 0004  07 0400  0008       ARRAY 64 INTEGER LENGTH SYN ESDTABLE(12); COMMENT BYTES 1-3;                 17.
06 0004  07 0400  0009       ARRAY 64 LOGICAL FLAGS  SYN ESDTABLE(12); COMMENT BYTE  0;                   18.
C6 0004  07 0400  0010      INTEGER ESDINDEX;                                                             19.
06 0004  07 0404  0011      INTEGER LOCORE, HICORE, LOADBASE;                                             20.
06 0004  07 0410  0012      INTEGER ENTRYPOINT; COMMENT EXECUTION ENTRY ADDRESS;                          21.
C6 0004  07 0414  0013      BYTE ENTRYSET, INIT, LOADERR;                                                 22.
06 0004  07 0417  0014      ARRAY 133 BYTE ERRBUF;                                                        23.
C6 0004  07 049C  0015                                                                                    24.
06 0004  07 049C  0016      INTEGER REGISTER                                                              25.
06 0004  07 049C  0017       I SYN R3, J SYN R4, K SYN R5, T SYN R6, W SYN R7, LINK SYN R8;               26.
C6 0004  07 049C  0018                                                                                    27.
C6 0004  07 049C  0019      ARRAY 65 SHORT INTEGER MODTABLE;                                              28.
06 0004  07 051E  0020      SHORT INTEGER ESDID;                                                          29.
C6 0004  07 0520  0021      FUNCTION DECR(6,#0600), POINT(6,#0500);                                       30.
C6 0004  07 0520  0022      ARRAY 12 INTEGER REGSAVE;                                                     31.
06 0004  07 0550  0023                                                                                    32.
C6 0004  07 0550  0024      INTEGER REGISTER XR SYN R9;                                                   33.
06 0004  07 0550  0025      BYTE  MEMOVFL   SYN 1,    ESDOVFL   SYN 2,    DUPSEG    SYN 3,              34.
06 0004  07 0550  0026            UNDEFSEG  SYN 4,    INVFMT    SYN 4,    ESDTYPE   SYN 4,              35.
06 0004  07 0550  0027            ESDSEQ    SYN 4;                                                      36.
06 0004  07 0550  0028     PROCEDURE ERROR(LINK);                                                         37.
C6 0004  07 0550  0029      BEGIN  COMMENT  XR - ERROR TYPE;                                              38.
06 0008  07 0550  0030         STC(XR,LOADERR);  GOTO EXIT;                                               39.
06 001C  07 0550  0031      END;                                                                          40.
06 0012  07 0550  0032                                                                                    41.
06 0012  07 0550  0033     PROCEDURE LOOKUP(LINK);                                                        42.
06 0012  07 0550  0034      BEGIN  COMMENT R0,R1 - NAME OF SEGMENT TO LOCATE OR ENTER, SETS I;            43.
C6 0012  07 0550  0035         LOGICAL WSAVE; WSAVE := W;                                                 44.
06 0016  07 0554  0036         FOR I := 0 STEP 16 UNTIL ESDINDEX DO                                       45.
C6 001A  07 0554  0037         IF R0 = NAME1(I) AND R1 = NAME2(I) THEN GOTO X;                            46.
C6 003A  07 0554  0038         I := ESDINDEX + 16; IF I >= 1024 THEN                                      47.
06 004A  07 0554  0039         BEGIN  XR := @ESDOVFL; ERROR;                                              48.
C6 0052  07 0554  0040         END;                                                                       49.
C6 0052  07 0554  0041         ESDINDEX := I; W := @ESDTABLE(I); STM(R0,R1,MEM(W));                       50.
06 005E  07 0554  0042         R0 := #1;  COMMENT END OF CHAIN MARKER;                                    51.
C6 0062  07 0554  0043         R1 := 0;  STM(R0,R1,MEM(W+8));                                             52.
06 006A  07 0554  0044   X:    W := WSAVE;                                                                53.
06 006E  07 0554  0045      END;                                                                          54.
C6 0070  07 0554  0046                                                                                    55.
06 0070  07 0554  0047     PROCEDURE ALLOCATESEG(LINK);                                                   56.
06 0070  07 0554  0048      BEGIN  COMMENT I - ESDTABLE POINTER, W - SEG LENGTH;                          57.
06 0070  07 0554  0049         LOGICAL WSAVE;  WSAVE := W;                                                58.
06 0074  07 0558  0050         W := FLAGS(I) AND #8C000000; IF ¬= THEN                                    59.
06 0080  07 0558  0051         BEGIN  W := @NAME1(I); MVC(7,ERRBUF(44),MEM(W));                           60.
06 008A  07 0558  0052            XR := @DUPSEG;  ERROR;                                                 61.
06 0092  07 0558  0053         END;                                                                       62.
C6 0092  07 0558  0054         R0 := LOADBASE - WSAVE AND #FFFFF8;  IF R0 < LOCORE THEN                   63.
06 00A6  07 0558  0055         BEGIN  R0 := _1;  REGSAVE(0) := R0;  COMMENT NO SPACE;                     64.
06 00AE  07 0558  0056            W := ESDID - 1;  ESDID := W;  GOTO EXIT;                               65.
06 00BE  07 0558  0057         END;                                                                       66.
06 00BE  07 0558  0058         LOADBASE := R0;  W := FLAGS(I) OR #80000000; FLAGS(I) := W;                 67.
```

```
06 00CE  07 0558  0059          W := ADDR(I);                                                    68.
C6 00D2  C7 0558  0060          WHILE W ¬= #1 DO                                                69.
06 00DA  C7 0558  0061          BEGIN  R1 := MEM(W);                                            70.
06 00DE  07 0558  0062             COMMENT  (ADDRESS/4096) BEFORE RELOCATION IN BYTE 0;         71.
C6 00DE  C7 0558  0063             R0 := R1 SHRL 24 SHLL 12 + LOADBASE; MEM(W) := R0;           72.
06 00FC  C7 0558  0064             W := R1 AND #FFFFFF;                                         73.
06 00F6  07 0558  0065          END;                                                            74.
C6 00FA  07 0558  0066          R0 := LOADBASE;  R1 := LENGTH(I) AND #FF000000 OR WSAVE;        75.
C6 010A  07 0558  0067          ADDR(I) := R0; LENGTH(I) := R1;                                 76.
06 0112  07 0558  0068       END;                                                               77.
C6 0114  07 0558  0069                                                                          78.
06 0114  07 0558  0070       ARRAY 20 LOGICAL CARD SYN MEM(R10);                                79.
06 0114  07 0558  0071        ARRAY 40 SHORT INTEGER SCARD SYN CARD;                            80.
C6 0114  07 0558  0072                                                                          81.
06 0114  C7 0558  0073       STM(R0,R10,REGSAVE);  R10 := R0;                                   82.
06 011A  07 0558  0074       IF R1 < LOCORE THEN LOCORE := R1;                                  83.
06 0126  07 0558  0075       CLI(#00,LOADERR);  IF = THEN                                       84.
06 012E  07 0558  0076       BEGIN  COMMENT PROCESS BUFFER CARD;                                85.
06 012E  07 0558  0077          IF INIT THEN                                                    86.
06 0136  07 0558  0078          BEGIN  RESET(INIT);                                             87.
06 013A  07 0558  0079             R0 := 0; ESDID := R0;  R1 := #1;                             88.
C6 0146  07 0558  0080             FOR K := 0 STEP 2 UNTIL 128 DO                               89.
06 014A  07 0558  0081             MODTABLE(K) := R1;  COMMENT MARK EMPTY;                      90.
C6 015E  07 0558  0082          END;                                                            91.
C6 015E  07 0558  0083          CLC(78,CARD(1),CARD); IF ¬= THEN                                92.
06 0168  07 0558  0084          BEGIN  R0 := CARD(0);                                           93.
06 016C  07 0558  0085             IF R0 = "BESD" THEN                                          94.
06 0174  07 0558  0086             BEGIN  COMMENT EXTERNAL SYMBOL DICTIONARY;                   95.
06 0174  07 0558  0087                W := ESDID + 1;  IF W ¬= SCARD(14) THEN                   96.
06 0184  07 0558  0088                BEGIN  XR := @ESDSEQ; ERROR;                             97.
06 018C  07 0558  0089                END;                                                     98.
06 018C  07 0558  0090                FOR J := 16 STEP 16 UNTIL SCARD(10) DO                   99.
06 0190  07 0558  0091                BEGIN  T := 0; IC(T,CARD(J+8));                         100.
C6 019C  07 0558  0092                   W := ESDID + 1;                                      101.
06 01A4  07 0558  0093                   IF W > 64 THEN                                       102.
C6 01AC  07 0558  0094                   BEGIN  XR := @ESDSEQ; ERROR;                         103.
06 01B4  07 0558  0095                   END;                                                 104.
06 01B4  07 0558  0096                   ESDID := W; K := W + W;                              105.
06 01BC  07 0558  0097                   R0 := CARD(J); R1 := CARD(J+4); LOOKUP;              106.
06 01C8  07 0558  0098                   MODTABLE(K) := I;                                    107.
C6 01CC  07 0558  0099                   IF T = 0 THEN  COMMENT  SD;                          108.
06 01D2  07 0558  0100                   BEGIN  W := CARD(J+12); ALLOCATESEG;                 109.
06 01DA  07 0558  0101                   END ELSE                                             110.
06 01DA  07 0558  0102                   IF T = 2 THEN NULL  COMMENT  ER;  ELSE               111.
06 01E6  07 0558  0103                   BEGIN  XR := @ESDTYPE; ERROR;                        112.
06 01F2  07 0558  0104                   END;                                                 113.
C6 01F2  07 0558  0105                END;                                                    114.
06 01FE  07 0558  0106             END ELSE                                                    115.
06 01FE  07 0558  0107             IF R0 = "BTXT" THEN                                         116.
06 020A  07 0558  0108             BEGIN  COMMENT EBCDIC TEXT;                                 117.
06 020A  07 0558  0109                K := SCARD(14) SHLA 1; I := MODTABLE(K);                 118.
06 0216  07 0558  0110                IF I ¬= #1 THEN W := FLAGS(I) AND #80000000;             119.
C6 0226  07 0558  0111                IF I = #1 OR W = 0 THEN                                  120.
06 0234  07 0558  0112                BEGIN  XR := @UNDEFSEG; ERROR;                           121.
06 023C  07 0558  0113                END;                                                     122.
06 023C  07 0558  0114                R1 := ADDR(I) + CARD(4); W := SCARD(10);                 123.
06 0248  07 0558  0115                BEGIN  DECR(W); POINT(R2);  GOTO SKIP;                   124.
C6 0250  07 0558  0116                   MVC(0,81,CARD(16));    COMMENT  EXECUTED;            125.
```

```
06 0256  07 0558   0117  SKIP:       EX(W,B2(4));   COMMENT EXECUTE MVC;                    126.
06 025A  07 0558   0118             END;                                                    127.
06 025A  07 0558   0119          END ELSE                                                   128.
06 025A  07 0558   0120          IF R0 = "BRLD" THEN                                        129.
06 0266  07 0558   0121          BEGIN  COMMENT RELOCATION DICTIONARY;                      130.
06 0266  07 0558   0122             W := SCARD(10) + 8; SCARD(10) := W;                     131.
C6 C272  07 0558   0123             FOR J := 16 STEP 8 UNTIL SCARD(10) DO                  132.
06 0276  07 0558   0124             BEGIN  K := SCARD(J+2) SHLA 1; I := MODTABLE(K);       133.
06 0286  07 0558   0125                IF I ¬= #1 THEN W := FLAGS(I) AND #80000000;        134.
06 02S6  07 0558   0126                IF I = #1 OR W = 0 THEN                             135.
06 02A4  07 0558   0127                BEGIN  XR := @UNDEFSEG; ERROR;                      136.
06 02AC  07 0558   0128                END;                                                137.
06 02AC  07 0558   0129                W := CARD(J+4) AND #FFFFFF + ADDR(I);               138.
06 02B8  07 0558   0130                K := SCARD(J) SHLA 1; I := MODTABLE(K);             139.
06 02C4  07 0558   0131                IF I = #1 THEN                                      140.
06 02CC  07 0558   0132                BEGIN  XR := @UNDEFSEG; ERROR;                      141.
06 02D4  07 0558   0133                END;                                                142.
06 02D4  07 0558   0134                R0 := 0; IC(R0,CARD(J+4));                          143.
06 02DC  07 0558   0135                R0 := FLAGS(I) AND #80000000;                       144.
06 02E4  07 0558   0136                IF ¬= THEN R0 := ADDR(I) + MEM(W)  ELSE             145.
06 02FC  07 0558   0137                BEGIN  COMMENT CHAIN FOR FIXUP;                     146.
06 02F4  07 0558   0138                   R0 := MEM(W) SHLL 12 OR ADDR(I);                 147.
C6 0300  07 0558   0139                   ADDR(I) := W;                                    148.
06 0304  07 0558   0140                END;                                                149.
06 0304  07 0558   0141                MEM(W) := R0;  COMMENT ADDRESS OR LINK;             150.
C6 0308  07 0558   0142             END;                                                   151.
C6 C314  07 0558   0143          END ELSE                                                   152.
06 0314  07 0558   0144          IF R0 = "BEND" THEN                                        153.
06 032C  07 0558   0145          BEGIN  T := CARD(4); IF T ¬= "    " AND ¬ENTRYSET THEN     154.
06 0334  07 0558   0146             BEGIN  K := SCARD(14) SHLA 1; I := MODTABLE(K);         155.
06 0340  07 0558   0147                IF I ¬= #1 THEN W := FLAGS(I) AND #80000000;        156.
06 0350  07 0558   0148                IF I = #1 OR W = 0 THEN                             157.
06 035E  07 0558   0149                BEGIN  XR := @UNDEFSEG; ERROR;                      158.
06 0366  07 0558   0150                END;                                                159.
06 0366  07 0558   0151                T := T AND #FFFFFF + ADDR(I);                       160.
06 036E  07 0558   0152                ENTRYPOINT := T; SET(ENTRYSET);                     161.
06 C376  07 0558   0153             END;                                                   162.
06 0376  07 0558   0154             SET(INIT);                                             163.
06 037A  07 0558   0155          END ELSE                                                   164.
C6 C37A  07 0558   0156          BEGIN  XR := @INVFMT; ERROR;                               165.
06 0386  07 0558   0157          END;                                                       166.
06 0386  07 0558   0158       END;                                                          167.
06 C386  07 0558   0159    END;                                                           168.
C6 0386  07 0558   0160  EXIT:                                                           169.
06 0386  07 0558   0161       LM(R0,R10,REGSAVE);                                        170.
06 038A  07 0558   0162    END.                                                          171.
```

SEGMENT C7 NAME = LENGTH = 0558 BASE REG = 11

SEGMENT 06 NAME = LOADCARD LENGTH = 03D8 BASE REG = 15

```
C1 C00C  00 0000   0001  GLOBAL PROCEDURE ENDLOAD(R14);                                                    173.
01 0000  C0 0000   0002   BEGIN                                                                           174.
C6 CCCC  00 0000   0003                                                                                   175.
06 CCCC  00 0000   0004     COMMON BASE R11;                                                              176.
C6 0004  C7 0000   0005      ARRAY 256 INTEGER ESDTABLE;    COMMENT GLOBAL SYMBOL DICTIONARY;             177.
06 0004  07 0400   0006       ARRAY 64 INTEGER NAME1  SYN ESDTABLE, NAME2  SYN ESDTABLE(4);              178.
C6 CCC4  07 0400   0007       ARRAY 64 INTEGER ADDR   SYN ESDTABLE(8);                                   179.
06 0004  07 0400   0008       ARRAY 64 LOGICAL FLAGS  SYN ESDTABLE(12);                                  180.
06 0004  07 0400   0009      INTEGER ESDINDEX;                                                            181.
C6 0004  07 0404   0010      INTEGER LOCORE, HICORE, LOADBASE;                                            182.
06 0004  07 0410   0011      INTEGER ENTRYPOINT;                                                          183.
C6 0004  07 0414   0012      BYTE ENTRYSET, INIT, LOADERR;                                                184.
06 0004  07 0417   0013      ARRAY 133 BYTE ERRBUF;                                                       185.
06 0004  07 049C   0014                                                                                   186.
C6 CCC4  07 049C   0015      INTEGER REGISTER                                                             187.
06 0004  07 049C   0016        I SYN R3, J SYN R4, K SYN R5, N SYN R6, T SYN R7, W SYN R8;                188.
06 0004  07 049C   0017                                                                                   189.
C6 0004  07 049C   0018     DUMMY BASE R9;     COMMENT LIBRARY DIRECTORY FORMAT;                          190.
06 0004  08 0000   0019      INTEGER LIBTABLEN;  COMMENT LENGTH OF DIRECTORY;                             191.
06 0004  08 0004   0020      INTEGER LIBENTRY, LIBERROR, LIBDATA;                                         192.
C6 CCC4  08 0010   0021      ARRAY 48 LOGICAL LIBTABLE;                                                   193.
06 0004  08 00D0   0022       ARRAY 16 INTEGER LIBNAME1 SYN LIBTABLE, LIBNAME2 SYN LIBTABLE(4);          194.
C6 0004  C8 00D0   0023       ARRAY 16 INTEGER LIBADDR  SYN LIBTABLE(8);                                 195.
C6 0004  C8 00D0   0024     CLOSE BASE;                                                                  196.
C6 0004  C7 049C   0025                                                                                   197.
C6 0004  07 049C   0026      ARRAY 10 INTEGER REGSAVE;                                                    198.
C6 0004  C7 04C4   0027      BYTE UNDEFSYM SYN 5;                                                         199.
C6 0004  C7 04C4   0028                                                                                   200.
C6 0004  C7 04C4   0029      STM(R0,R9,REGSAVE);  R9 := R0;                                               201.
06 000A  07 04C4   0030      CLI(#00,LOADERR); IF = THEN                                                  202.
06 0012  07 04C4   0031      BEGIN  COMMENT MERGE DIRECTORIES, FIX UP ADDRESSES;                          203.
06 0012  07 04C4   0032        FOR I := 0 STEP 16 UNTIL ESDINDEX DO                                       204.
06 0016  07 04C4   0033        BEGIN  R0 := NAME1(I); R1 := NAME2(I);                                     205.
06 0022  07 04C4   0034          W := FLAGS(I) AND #80000000; IF ¬= THEN                                  206.
06 002E  07 04C4   0035          BEGIN  COMMENT TEST FOR SEGMENTS WITH SPECIAL PROCESSING;                207.
C6 002E  07 04C4   0036            INTEGER ENTRYADDR SYN #80;     COMMENT ALGOLRUN DISPL.;                208.
06 002E  07 04C4   0037            ARRAY 64 LOGICAL RECTAB SYN #84;                                       209.
06 002E  07 04C4   0038            IF R1 = "C001" AND R0 = "AWXS" THEN                                    210.
06 003E  07 04C4   0039            BEGIN  R1 := LIBDATA;  R0 := ADDR(I);                                  211.
06 0046  07 04C4   0040              ENTRYADDR(R1) := R0;                                                 212.
C6 004A  07 04C4   0041            END ELSE                                                               213.
06 004A  07 04C4   0042            IF R1 = "CTBL" AND R0 = "AWXR" THEN                                    214.
C6 005E  07 04C4   0043            BEGIN  R1 := LIBDATA;  R2 := ADDR(I);                                  215.
06 0066  07 04C4   0044              MVC(255,RECTAB(R1),B2);                                              216.
C6 006C  07 04C4   0045            END;                                                                   217.
06 006C  07 04C4   0046          END ELSE                                                                 218.
06 006C  07 04C4   0047          BEGIN   N := #1;                                                         219.
C6 0074  07 04C4   0048            J := 0;  K := LIBTABLEN;   COMMENT BINARY SEARCH;                      220.
C6 007C  07 04C4   0049            WHILE J <= K AND N = #1 DO                                             221.
C6 008A  07 04C4   0050            BEGIN  W := J + K SHRL 1;  T := W * 12S;                               222.
C6 0098  07 04C4   0051              IF R0 = LIBNAME1(T) THEN                                             223.
06 00AC  07 04C4   0052              BEGIN                                                                224.
06 00A0  07 04C4   0053                IF R1 = LIBNAME2(T) THEN N := T ELSE                               225.
C6 00AA  07 04C4   0054                IF > THEN J := W + 1 ELSE K := W - 1;                              226.
06 00C2  07 04C4   0055              END ELSE                                                             227.
06 00C2  07 04C4   0056              IF > THEN J := W + 1 ELSE K := W - 1;                                228.
06 00DA  07 04C4   0057            END;                                                                   229.
06 00DE  07 04C4   0058            IF N = #1 THEN                                                         230.
```

```
06 00E6  07 04C4  0059              BEGIN  W := @UNDEFSYM; STC(W,LOADERR);                          231.
06 00EE  07 04C4  0060                 W := @NAME1(I);  MVC(7,ERRBUF(44),MEM(W)); GOTO EXIT;      232.
C6 00FC  07 04C4  0061              END;                                                           233.
C6 00FC  07 04C4  0062              T := LIBADDR(N);  W := ADDR(I);                                234.
06 0104  07 04C4  0063              WHILE W ¬= #1 DO                                               235.
06 010C  07 04C4  0064              BEGIN  R1 := MEM(W);                                           236.
06 0110  07 04C4  0065                 R0 := R1 SHRL 24 SHLL 12 + T;  MEM(W) := R0;               237.
06 0120  07 04C4  0066                 W := R1 AND #FFFFFF;                                       238.
06 0126  07 04C4  0067              END;                                                           239.
06 012A  07 04C4  0068           END;                                                              240.
06 012A  07 04C4  0069        END;                                                                 241.
06 0136  07 04C4  0070  EXIT:                                                                     242.
C6 0136  07 04C4  0071     END;                                                                   243.
06 0136  07 04C4  0072     CLI(#00,LOADERR);  IF ¬= THEN                                          244.
06 013E  07 04C4  0073     BEGIN  MVC(18,ERRBUF,"1*** LOADING ERROR,");                           245.
C6 0144  07 04C4  0074        W := 0;  IC(W,LOADERR);  CASE W OF                                  246.
06 014C  07 04C4  0075        BEGIN                                                               247.
C6 014C  07 04C4  0076           MVC(19,ERRBUF(20),"INSUFFICIENT STORAGE");                       248.
C6 015A  07 04C4  0077           MVC(18,ERRBUF(20),"TOO MANY PROCEDURES");                        249.
06 0164  07 04C4  0078           MVC(22,ERRBUF(20),"DUPLICATE GLOBAL NAME -");                    250.
06 016E  07 04C4  0079           MVC(21,ERRBUF(20),"INVALID OBJECT RECORDS");                     251.
C6 0178  07 04C4  0080           MVC(22,ERRBUF(20),"UNDEFINED GLOBAL NAME -");                    252.
06 0182  07 04C4  0081        END;                                                                253.
06 019A  07 04C4  0082     END ELSE                                                               254.
06 019A  07 04C4  0083     BEGIN  IF ¬ENTRYSET THEN                                               255.
C6 01A6  07 04C4  0084        BEGIN  R0 := LIBENTRY;  ENTRYPOINT := R0;                           256.
06 01AE  07 04C4  0085           SET(ENTRYSET);                                                   257.
06 01B2  07 04C4  0086        END;                                                                258.
C6 01B2  07 04C4  0087     END;                                                                   259.
C6 01B2  07 04C4  0088  END.                                                                      260.
```

SEGMENT 07 NAME = LENGTH = 04C8 BASE REG = 11

SEGMENT 06 NAME = ENDLOAD LENGTH = 0258 BASE REG = 15

```
IEF285I   T123.PLLIB                                      PASSED
IEF285I   VOL SER NOS= SYS11 .
IEF285I   SYSOUT                                          SYSOUT
IEF285I   VOL SER NOS=       .
IEF285I   LOADSET.XALGOL                                  PASSED
IEF285I   VOL SER NOS= SYS02 .
//ASM      EXEC  ASMGCL,PARM.ASM='LOAD,LIST,NODECK,NORLD,NOXREF'          262.
//ASM     EXEC  PGM=ASMGASM,PARM='LOAD,NODECK,FULLXREF,ESD'              00000000
//SYSLIB DD     DSNAME=SYS1.MACLIB,DISP=OLD                              00000100
//           DD  DSNAME=SYS2.DCLIB,DISP=OLD                             00000200
//    DD    DDNAME=MACLIB                                               00000300
//MACLIB  DD  DSNAME=SYS2.DUMMYMAC,DISP=OLD                            00000400
//SYSUT1    DD  DSNAME=SYS1.UT1,DISP=OLD,DCB=(KEYLEN=0,BLKSIZE=3520)   00000500
//SYSUT2    DD  DSNAME=SYS1.UT2,DISP=OLD,DCB=(KEYLEN=0,BLKSIZE=3520)   00000600
//SYSUT3    DD  DSNAME=SYS1.UT3,DISP=OLD,DCB=(KEYLEN=0,BLKSIZE=3520)   00000700
//SYSPRINT DD   SYSOUT=A                                               00000800
//SYSLIN     DD DSNAME=&LOADSET,UNIT=2314,SPACE=(CYL,(3,1)),          *00000900
//              DISP=(MOD,PASS),DCB=(LRECL=80,BLKSIZE=3200,RECFM=FB)   00001000
//SYSPUNCH  DD UNIT=SYSCP                                              00001100
//ASM.SYSIN DD  *                                                       263.
IEF236I ALLOC. FOR XALGOL    ASM       ASM
IEF237I JOBLIB    ON 434
IEF237I SYSLIB    ON 331
IEF237I           ON 430
IEF237I           ON 430
IEF237I SYSUT1    ON 1C1
IEF237I SYSUT2    ON 330
IEF237I SYSUT3    ON 530
IEF237I SYSLIN    ON 330
IEF237I SYSPUNCH ON 00D
IEF237I SYSIN     ON 00C
```

OS/360 ASSEMBLER

LEVEL=G RELEASE=16JUL69 SYSTEM=MFT TIME=18:02:48 DAY=MONDAY DATE=22 DEC 69

```
ASSEMBLER OPTIONS=LIST,LOAD,NOESD,NORLD,NODECK,NORENT,NOTEST,NOXREF,EXTIME=5,NOBATCH,UTBUFF=3,INSTSET=1,
                  LINECNT=55,NOEXECUTE,SPACE=MAX-2K.
```

```
 LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                                22 DEC 69
                                     1          ICTL    1,71,18              ESTABLISH FORMAT                  264.
                                     2          PRINT   NOGEN                                                 265.

                                     4 ************************************************************************* 267.
                                     5 *                                                                       * 268.
                                     6 *                    STANFORD ALGOL W SUBMONITOR                        * 269.
                                     7 *                     OS/360 STUDENT SUBSYSTEM                          * 270.
                                     8 *                          AUGUST 1969                                  * 271.
                                     9 *                                                                       * 272.
                                    10 ************************************************************************* 273.


000000                              12 AWXGDR01 CSECT                                                          275.

000000                              14          USING   AWXGDR01,15                                            277.
                                    15          SAVE    (14,12),,*           ESTABLISH OS CONVENTIONS          278.
000012 18CF                         20          LR      12,15                                                  279.
000000                              21          USING   AWXGDR01,12                                            280.
                                    22          DROP    15                                                     281.
000014 182D                         23          LR      2,13                                                   282.
000016 58D0 C4D8          004D8     24          L       13,=A(SAVE)                                            283.
00001A 50D0 2008          00008     25          ST      13,8(,2)                                               284.
00001E 5020 D004          00004     26          ST      2,4(,13)                                               285.
000022 58B0 C4DC          004DC     27          L       11,=A(COMMON)        ESTABLISH COMMON ADDRESSING       286.
000000                              28          USING   COMMON,11                                              287.

000026 9200 CC93          00C93     30          MVI     RDRSTAT,0            INITIALIZE READER FLAGS           289.
00002A 9200 CC94          00C94     31          MVI     REREAD,0                                               290.
00002E 9200 CC92          00C92     32          MVI     ENDDS,0                                                291.
000032 9240 B417          00417     33          MVI     BLANK,C' '                                             292.
                                    34          OPEN    ,MF=(E,DMGLIST)      OPEN DATA SETS                    293.
                                    37          BLDL    0,LIBDIR             CONSTRUCT PDS DIRECTORY IN CORE   294.
000046 1800                         41          SR      0,0                  INDICATE NO STORAGE HELD          295.
000048 5000 C81C          0081C     42          ST      0,SDVLEN                                               296.
00004C 4000 CBAC          00BAC     43          STH     0,ISTATE             INITIALIZE INPUT STATE (SYSIN)    297.
000050 9200 CC90          00C90     44          MVI     CONTCARD,0           INITIALIZE JOB FLAGS              298.
000054 9200 CC91          00C91     45          MVI     EOF,0                                                  299.

                                    47 *                INPUT SYSTEM STATE CODES                          *  301.
000001                              48 SA1      EQU     1                    IN STREAM A1                      302.
000002                              49 SB       EQU     2                    IN STREAM B                       303.
000003                              50 SA2      EQU     3                    IN STREAM A2                      304.
```

```
 LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT
                                    52 ********************************************************************** 306.
                                    53 *     THE FOLLOWING CODE SCHEDULES EACH JOB THROUGH THE SUBSYSTEM,    * 307.
                                    54 *     DECODING CONTROL CARDS AND MANAGING STORAGE ALLOCATION.         * 308.
                                    55 ********************************************************************** 309.

000058                              57 LOOP   DS      0H                    PROCESSING CYCLES HERE FOR JOB    311.

000058 95FF CC90      00C90         59 A0     CLI     CONTCARD,X'FF'        TEST FOR CONTROL CARD IN BUFFER   313.
00005C 4780 C06E            0006E   60        BE      A1                                                      314.
000060 4100 CC40            00C40   61        LA      0,CONBUFF             READ TO CONTROL CARD              315.
000064 5830 C4E0            004E0   62        L       3,=A(GETCARD)                                           316.
000068 0523                         63        BALR    2,3                                                     317.
00006A 47F0 C058            00058   64        B       A0                                                      318.

00006E 9200 CC90      00C90         66 A1     MVI     CONTCARD,0            RESET JOB FLAGS                   320.
000072 9200 CC91      00C91         67        MVI     EOF,0                                                   321.
000076 92FF CC96      00C96         68        MVI     XMIT,X'FF'                                              322.
00007A 9200 B416      00416         69        MVI     LOADERR,0                                               323.
00007E 4800 CBAC            00BAC   70        LH      0,ISTATE              TEST FOR SYSIN INPUT              324.
000082 1200                         71        LTR     0,0                                                     325.
000084 4770 C0B4            000B4   72        BNZ     B0                                                      326.
000088 D5C7 CC5D C4D0 00C5D 004D0   73        CLC     CONBUFF+29(8),=CL8' ' TEST FOR SOURCE FILE NAME         327.
00008E 4780 C0B4            000B4   74        BE      B0                                                      328.
000092 D2C7 CBA0 CC5D 00BA0 00C5D   75        MVC     EPNAME(8),CONBUFF+29  SAVE MODULE NAME                  329.
000098 9200 CC96      00C96         76        MVI     XMIT,0                INSERT IN FIRST OUTPUT LINE       330.
                                    77        LOAD    EPLOC=EPNAME,DCB=0    LOAD SOURCE MODULE                331.
0000A6 5000 CBA8            00BA8   81        ST      0,INEXT               INITIALIZE SCAN POINTER           332.
0000AA 4100 0001            00001   82        LA      0,SA1                 SET STATE = A1                    333.
0000AE 4000 CBAC            00BAC   83        STH     0,ISTATE                                                334.

                                    85 *   THE FOLLOWING SECTION ALLOCATES DYNAMICALLY OBTAINED STORAGE     * 336.
                                    86 *   LET    SS    =  SYSTEM WORK SPACE (ABEND,EOV,ETC)               * 337.
                                    87 *          T+SS  =  TOTAL STORAGE OBTAINED                          * 338.
                                    88 *          L     =  STORAGE RESERVED FOR LINK/LOAD (FREED)          * 339.
                                    89 *          WMIN  =  MINIMUM WORK AREA SIZE                          * 340.
                                    90 *          WMAX  =  MAXIMUM WORK AREA SIZE                          * 341.
                                    91 *          W     =  ACTUAL WORK AREA SIZE (FREED)                   * 342.
                                    92 *          CMIN  =  MINIMUM COMMON SIZE                             * 343.
                                    93 *          CMAX  =  MAXIMUM COMMON SIZE                             * 344.
                                    94 *          C     =  ACTUAL COMMON SIZE                              * 345.
                                    95 *   SET    DX    =  T - (L + WMIN + CMIN)                           * 346.
                                    96 *          DW    =  IF DX <= CMIN THEN 0 ELSE                       * 347.
                                    97 *                     MIN( (DX-CMIN)/32, WMAX-WMIN )                * 348.
                                    98 *   THEN   W     =  WMIN + DW                                       * 349.
                                    99 *          C     =  CMIN + (DX - DW)                                * 350.

000400                             101 K      EQU     1024                                                    352.
000800                             102 SS     EQU     2*K                                                     353.
00D000                             103 L      EQU     52*K                                                    354.
003000                             104 WMIN   EQU     12*K                                                    355.
005800                             105 WMAX   EQU     22*K                                                    356.
006000                             106 CMIN   EQU     24*K                                                    357.
080000                             107 CMAX   EQU     512*K                                                   358.
016800                             108 CCREMIN EQU    L+SS+WMIN+CMIN        MINIMUM GETMAIN REQUEST           359.
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT
C93000                           109 COREMAX  EQU     L+SS+WMAX+CMAX        MAXIMUM GETMAIN REQUEST          360.

                                 111 B0       GETMAIN VU,                   INITIAL GETMAIN                1 362.
                                                      LA=ALLCORE,            OBTAIN ALL FREE STORAGE        2 363.
                                                      A=SDVLIST,SP=0                                          364.
0000C4 9823 C818           00818 119          LM      2,3,SDVLIST           R3 = T + SS,  R2 = BASE          365.
0000C8 5810 C4E4           004E4 120          L       1,=A(SS)              R1 = SS                          366.
0000CC 4102 3000           00000 121          LA      0,0(2,3)                                               367.
0000D0 1B01                      122          SR      0,1                   R0 = BASE OF SYSTEM AREA         368.
0000D2 1B31                      123          SR      3,1                   R3 = T                           369.
0000D4 9001 C818           00818 124          STM     0,1,SDVLIST           RELEASE SYSTEM SPACE             370.
                                 125          FREEMAIN ,MF=(E,SMGLIST)                                       371.

0000DE 1853                      129 A2       LR      5,3                   R5 = T                           373.
0000E0 5B50 C4E8           004E8 130          S       5,=A(L+WMIN+CMIN)     R5 = DX                          374.
0000E4 1835                      131          LR      3,5                                                    375.
0000E6 5B30 C4EC           004EC 132          S       3,=A(CMIN)            R3 = DX - CMIN                   376.
0000EA 4720 C0F4           000F4 133          BH      A3                                                     377.
0000EE 1B33                      134          SR      3,3                   R3 = 0 = DW  (DX <= CMIN)        378.
0000F0 47F0 C108           00108 135          B       A5                                                     379.
0000F4 8830 0005           00005 136 A3       SRL     3,5                   R3 = (DX - CMIN)/32              380.
0000F8 5930 C4F0           004F0 137          C       3,=A(WMAX-WMIN)                                        381.
0000FC 47D0 C104           00104 138          BNH     A4                                                     382.
000100 5830 C4F0           004F0 139          L       3,=A(WMAX-WMIN)       R3 = WMAX - WMIN                 383.
000104 5430 C4F4           004F4 140 A4       N       3,=XL4'00FFFFF8'      R3 = DW  (DX > CMIN)             384.
000108 1B53                      141 A5       SR      5,3                   R5 = DX - DW                     385.
00010A 5A30 C4F8           004F8 142          A       3,=A(L+WMIN)          R3 = L + W                       386.
00010E 9023 C818           00818 143          STM     2,3,SDVLIST           RELEASE LINK AND WORK SPACE      387.
                                 144          FREEMAIN ,MF=(E,SMGLIST)       (NOT CONTIGUOUS WITH SYS SPACE) 388.
000118 4142 3000           00000 147          LA      4,0(2,3)              R4 = COMMON BASE                 389.
00011C 5A50 C4EC           004EC 148          A       5,=A(CMIN)            R5 = C                           390.
000120 9045 C818           00818 149          STM     4,5,SDVLIST           SAVE FOR LATER RELEASE           391.
000124 1A54                      150          AR      5,4                                                    392.
000126 5050 B40C           0040C 151          ST      5,LOADBASE                                             393.
00012A 0650                      152          BCTR    5,0                                                    394.
00012C 9045 C858           00858 153          STM     4,5,COMMLIM           ESTABLISH COMMON LIMITS          395.
000130 4540 C41C           0041C 154          BAL     4,ANALYZE             ANALYZE CONTROL CARD FIELDS      396.
000134 4540 C2FA           002FA 155          BAL     4,SETID               SET SYSTEM ID STRING             397.
```

```
 LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                                    22 DEC 69
                                   157 ********************************************************************** 399.
                                   158 *     THE FOLLOWING CODE CONTROLS LOADING OF THE COMPILER PHASES     * 400.
                                   159 ********************************************************************** 401.


000138                             161 COMPILE  DS      0H                    EXECUTE COMPILER PHASES          403.

000138 1B00                        163          SR      0,0                   MASK INTERRUPTIONS               405.
00013A 0400                        164          SPM     0                                                      406.
00013C 5800 C4FC          004FC    165          L       0,=XL4'7FFFFFFF'                                       407.
000140 5000 CBB0          00BB0    166          ST      0,TIMEINIT                                             408.
                                   167          STIMER  TASK,TUINTVL=TIMEINIT SET TASK TIMER FOR CLOCKING      409.
00014C 4110 C858          00858    171          LA      1,ENTVECT             LOAD PARAMETER REGISTER          410.
                                   172          LINK    DE=DPHASEA,DCB=0,MF=(E,(1))   EXECUTE PHASE A          411.
00015E 12FF                        179          LTR     15,15                 TEST FOR ERRORS                  412.
000160 4770 C1AC          001AC    180          BNZ     C1                    IF ANY, BYPASS PHASE B           413.

                                   182 *      INITIALIZE VARIABLES FOR LOADER COROUTINE (LOADCARD)          * 415.
000164 5800 C500          00500    183          L       0,=F'-16'             MARK ESDTABLE EMPTY              416.
000168 5000 B400          00400    184          ST      0,ESDINDEX                                             417.
00016C 5810 C85C          0085C    185          L       1,COMMLIM+4           SET UP STORAGE BOUNDS            418.
000170 4100 1001          00001    186          LA      0,1(,1)                                                419.
000174 5000 B404          00404    187          ST      0,LOCORE                                               420.
000178 5000 B408          00408    188          ST      0,HICORE                                               421.
00017C 5000 B40C          0040C    189          ST      0,LOADBASE                                             422.
000180 9200 B414    00414         190          MVI     ENTRYSET,0            SET LOADER STATUS FLAGS          423.
000184 9200 B416    00416         191          MVI     LOADERR,0                                              424.
000188 92FF B415    00415         192          MVI     INIT,X'FF'                                             425.
00018C D283 B418 B417 00418 00417 193          MVC     ERRBUF(132),BLANK                                      426.

000192 4110 C858          00858    195          LA      1,ENTVECT                                              428.
                                   196          LINK    DE=DPHASEB,DCB=0,MF=(E,(1))   EXECUTE PHASE B          429.
0001A6 12FF                        203          LTR     15,15                                                  430.
0001A8 4780 C1C2          001C2    204          BZ      LOADTEXT              NO ERRORS - EXECUTE              431.

                                   206 C1       TTIMER  CANCEL                CANCEL TIME INTERVAL             433.
                                   209          FREEMAIN ,MF=(E,SMGLIST)      RELEASE COMMON AREA              434.
0001B8 1B00                        212          SR      0,0                   MARK STORAGE FREE                435.
0001BA 5000 C81C          0081C    213          ST      0,SDVLEN                                               436.
0001BE 47F0 C058          00058    214          B       LOOP                  RECYCLE                          437.
```

```
 LCC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                                    22 DEC 69
                                   216 ************************************************************************  439.
                                   217 *     THE FOLLOWING CODE PROCESSES ANY OBJECT CARDS, LOADS           *  440.
                                   218 *     ALGOLRUN, AND EXECUTES THE COMPILED PROGRAM                    *  441.
                                   219 ************************************************************************  442.


0001C2                             221 LOADTEXT DS      0H                  LOAD OBJECT CARDS, EXECUTE          444.

0001C2 95FF CC91       00C91       223 X1       CLI     EOF,X'FF'           TEST FOR END-OF-FILE                446.
0001C6 4780 C1D8             001D8 224          BE      X2                                                      447.
0001CA 41C0 CC40             00C40 225          LA      0,CONBUFF                                               448.
0001CE 5830 C4E0             004E0 226          L       3,=A(GETCARD)                                           449.
0001D2 0523                        227          BALR    2,3                                                     450.
0001D4 47F0 C1C2             001C2 228          B       X1                                                      451.

0001D8 5840 B40C             0040C 230 X2       L       4,LOADBASE          R4 = BASE OF AREA IN USE            453.
0001DC 5B40 C818             00818 231          S       4,SDVBASE           R4 = LENGTH OF AREA FREE            454.
0001E0 5850 C81C             0081C 232          L       5,SDVLEN            R5 = LENGTH OF ORIGINAL AREA        455.
0001E4 1B54                        233          SR      5,4                 R5 = LENGTH OF AREA IN USE          456.
0001E6 5040 C81C             0081C 234          ST      4,SDVLEN                                                457.
                                   235          FREEMAIN ,MF=(E,SMGLIST)    RELEASE FREE AREA                   458.
0001F0 5840 B40C             0040C 238          L       4,LOADBASE          R4 = BASE OF AREA IN USE            459.
0001F4 9045 C818             00818 239          STM     4,5,SDVLIST         SAVE FOR LATER RELEASE              460.
0001F8 D200 C854 B416 00854 00416  240          MVC     NOLOAD(1),LOADERR   SAVE ERROR STATUS                  461.
0001FE 9500 C854       00854       241          CLI     NOLOAD,0            TEST FOR ERRORS                     462.
0002C2 4770 C224             00224 242          BNE     X4                  IF ANY, BYPASS LOADING LIBRARY      463.
                                   243          LOAD    DE=DRUNLIB,DCB=0    LOAD RUN-TIME LIBRARY MODULE        464.
000212 1810                        248          LR      1,0                   (R0 := ADDR OF LIBR DIRECTORY)    465.
000214 5820 1008             00008 249          L       2,DERROR(,1)        SAVE ERROR ENTRY ADDRESS            466.
000218 5020 CBB4             00BB4 250          ST      2,AERROR                                                467.
00021C 5820 100C             0000C 251          L       2,DDATA(,1)          AND DATA SEGMENT ADDRESS           468.
000220 5020 CBB8             00BB8 252          ST      2,ARUNDSEG                                              469.
000224 58F0 C504             00504 253 X4       L       15,=V(ENDLOAD)      MERGE DIRECTORIES, COMPLETE LOAD    470.
000228 05EF                        254          BALR    14,15                                                   471.
00022A 9500 B416       00416       255          CLI     LOADERR,0           BYPASS EXECUTION IF ERRORS          472.
00022E 4770 C296             00296 256          BNE     X5                                                      473.

000232 9200 C855       00855       258          MVI     TIMEOUT,0           RESET EXECUTION FLAGS               475.
000236 92FF CC95       00C95       259          MVI     OUTSYNCH,X'FF'                                          476.
                                   260          SPIE    SPIEEXIT,((1,15))   SET PROGRAM CHECK EXIT             477.
000248 5010 C820             00820 269          ST      1,OLDSPIE                                               478.
00024C 5800 C828             00828 270          L       0,XLLIM             SET LINE LIMIT                      479.
000250 5000 C85C             0085C 271          ST      0,LINELIM                                               480.
000254 5800 C824             00824 272          L       0,XTLIM             SET TASK TIME LIMIT                 481.
000258 5000 CBB0             00BB0 273          ST      0,TIMEINIT                                              482.
                                   274          STIMER  TASK,TIMEEXIT,TUINTVL=TIMEINIT                          483.
000266 D200 CC91 CC90 00C91 00C90  278          MVC     EOF(1),CONTCARD     SET EXECUTION EOF                   484.
00026C D600 CC91 CC92 00C91 00C92  279          OC      EOF(1),ENDDS                                            485.
000272 D200 C858 CC91 00858 00C91  280          MVC     XEOF(1),EOF         SET RUN FLAGS                       486.
000278 9200 C859       00859       281          MVI     XTRACE,0                                                487.
00027C 4110 C858             00858 282          LA      1,ENTVECT                                               488.
000280 58F0 B410             00410 283          L       15,ENTRYPT          CALL COMPILED PROGRAM               489.
000284 05EF                        284          BALR    14,15                                                   490.
                                   285          TTIMER  CANCEL              CANCEL TIME INTERVAL                491.
00028C 5810 C820             00820 288          L       1,OLDSPIE           RESTORE SPIE CONDITIONS             492.
```

```
 LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                                    
                                   289           SPIE     ,MF=(E,(1))                                        493.
000292 47F0 C2A4              002A4 291           B        X6                                                 494.
                                                                                                            
000296 4100 B418              00418 293 X5        LA       0,ERRBUF          PRINT ERROR MESSAGE              496.
00029A 4110 00F1              000F1 294           LA       1,C'1'                                             497.
00029E 583C C508              00508 295           L        3,=A(PUTLINE)                                      498.
0002A2 0523                         296           BALR     2,3                                                499.
                                                                                                            
0002A4 9500 C854       00854        298 X6        CLI      NOLOAD,0          DELETE LIBR MODULE IF LOADED     501.
0002A8 4780 C2B6              002B6 299           BE       X7                                                 502.
                                    300           TTIMER   CANCEL            CANCEL TIME INTERVAL (COMPILER)  503.
0002B2 47F0 C2BC              002BC 303           B        X8                                                 504.
                                    304 X7        DELETE   DE=DRUNLIB        DELETE LIBRARY MODULE            505.
0002BC 5800 C81C              0081C 308 X8        L        0,SDVLEN          RELEASE ANY STORAGE HELD         506.
0002C0 1200                         309           LTR      0,0                                                507.
0002C2 4780 C2CC              002CC 310           BZ       X9                                                 508.
                                    311           FREEMAIN ,MF=(E,SMGLIST)                                    509.
0002CC 1B00                         314 X9        SR       0,0                                                510.
0002CE 5000 C81C              0081C 315           ST       0,SDVLEN                                           511.
0002D2 47F0 C058              00058 316           B        LOOP              CYCLE                            512.
                                                                                                            
000008                              318 DERROR    EQU      8                 DISPL OF ERROR ENTRY ADDR        514.
00000C                              319 DDATA     EQU      12                DISPL OF DATA SEGMENT ADDR       515.
```

```
 LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                                22 DEC 69
                                   321 ************************************************************************  517.
                                   322 *     THE FOLLOWING CODE RELEASES RESOURCES AND TERMINATES         *  518.
                                   323 ************************************************************************  519.

0002D6                             325 CLEANUP  DS      0H                    TERMINATE SUBSYSTEM              521.

0002D6 5800 C81C          0081C    327          L       0,SDVLEN              DETERMINE IF STORAGE HELD        523.
0002DA 1200                        328          LTR     0,0                                                    524.
0002DC 47D0 C2E6          002E6    329          BNH     T0                                                     525.
                                   330          FREEMAIN ,MF=(E,SMGLIST)      RELEASE REMAINING STORAGE        526.
                                   333 T0       CLOSE   ,MF=(E,DMGLIST)       CLOSE DATA SETS                  527.
0002EC 58D0 D004          00004    336          L       13,4(,13)                                              528.
                                   337          RETURN  (14,12),RC=0                                           529.
```

```
 LCC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                                    22 DEC 69
                                   342 ************************************************************************  531.
                                   343 *     THE FOLLOWING CODE FILLS A CHARACTER STRING WITH TIME          *  532.
                                   344 *     AND DATE FOR SYSTEM ID.  DECIMAL INSTRUCTIONS USED.            *  533.
                                   345 *          R4 = RETURN ADDRESS                                       *  534.
                                   346 ************************************************************************  535.

 0002FA                            348 SETID    DS      0H                    SET 'RUNID' STRING                537.

 0002FA 9240 C834       00834      350          MVI     RUNID,C' '            BLANK STRING                      539.
 0002FE D21E C835 C834  00835 00834 351          MVC     RUNID+1(31),RUNID                                       540.
                                   352          TIME    DEC                   OBTAIN TIME, DATE                 541.
 00030A 8800 000C             0000C 355          SRL     0,12                  REDUCE TIME TO MINUTES            542.
 00030E 5000 C82C             0082C 356          ST      0,TIME                                                  543.
 000312 960F C82F       0082F      357          OI      TIME+3,X'0F'          SET ZONE FOR UNPACKING            544.
 000316 5010 C830             C0830 358          ST      1,DATE                FORMAT IS '00YYDDD&' (DECIMAL)    545.
 00031A 9101 C831       00831      359          TM      DYEAR,X'01'           LEAP YEAR IF                      546.
 00031E 4710 C332             00332 360          BO      NOLEAP                  EVEN &                          547.
 000322 9112 C831       00831      361          TM      DYEAR,X'12'             TWO'S BIT OF YEAR =             548.
 000326 4740 C332             00332 362          BM      NOLEAP                  ONE'S BIT OF DECADE             549.
 00032A 929C C3A5       003A5      363          MVI     MDAY+3,X'9C'          ADJUST ENTRY FOR FEBRUARY         550.
 00032E 47F0 C336             00336 364          B       SETID1                                                  551.
 000332 928C C3A5       003A5      365 NOLEAP   MVI     MDAY+3,X'8C'          SET FOR NORMAL YEAR               552.
 000336 4110 C3A2             003A2 366 SETID1   LA      1,MDAY                INITIALIZE TABLE POINTERS         553.
 00033A 4120 C3D2             003D2 367          LA      2,MSTRING                                               554.
 00033E F911 C832 1000  00832 00000 368 SETID2   CP      DDAY(2),MDAYS(2,1)    CHECK IF REDUCED DATE IN MONTH    555.
 000344 47D0 C35A             0035A 369          BNH     SETID3                IF SO, DDAY IS DAY OF MONTH       556.
 000348 FB11 C832 1000  00832 00000 370          SP      DDAY(2),MDAYS(2,1)    IF NOT, REDUCE BY DAYS IN MONTH   557.
 00034E 4A20 1018             00018 371          AH      2,MLEN(,1)            STEP POINTERS                     558.
 000352 4110 1002             00002 372          LA      1,2(,1)                                                 559.
 000356 47F0 C33E             0033E 373          B       SETID2                                                  560.

 00035A F321 C834 C832  00834 00832 375 SETID3   UNPK    RUNID(3),DDAY(2)      CONVERT TO CHARACTER FORMAT       562.
 000360 96F0 C836       00836      376          OI      RUNID+2,X'F0'                                           563.
 000364 9240 C834       00834      377          MVI     RUNID,C' '                                              564.
 000368 4810 1018             00018 378          LH      1,MLEN(,1)            R1 = LENGTH OF MONTH NAME         565.
 00036C 4131 C838             00838 379          LA      3,RUNID+4(1)          R3 = START OF YEAR FIELD          566.
 000370 0610                        380          BCTR    1,0                                                     567.
 000372 4410 C39C             0039C 381          EX      1,MMOVE               MOVE MONTH NAME                   568.
 000376 D201 3001 C51C  00001 0051C 382          MVC     1(2,3),=C'19'         SET YEAR                          569.
 00037C F321 3003 C831  00003 00831 383          UNPK    3(3,3),DYEAR(2)                                         570.
 000382 9240 3005       00005      384          MVI     5(3),C' '             FORMAT TIME - R3 POINTS TO        571.
 000386 927C 3007       00007      385          MVI     7(3),C'@'              ' 19YY  @  HH:MM'                572.
 00038A F333 300B C82C  0000B 0082C 386          UNPK    11(4,3),TIME(4)                                         573.
 000390 D201 300A 300B  0000A 0000B 387          MVC     10(2,3),11(3)                                           574.
 000396 927A 300C       0000C      388          MVI     12(3),C':'                                              575.
 00039A C7F4                        389          BR      4                     RETURN                            576.

 00039C D200 C838 2000  00838 00000 391 MMOVE    MVC     RUNID+4(0),0(2)       EXECUTED                          578.
 0003A2                             392 MDAY     DS      0H                    MONTH LENGTH AND POINTER TABLES   579.
 000000                             393 MDAYS    EQU     *-MDAY                DAYS PER MONTH (2 BYTE DECIMAL)   580.
 0003A2 031C020C031C030C            394          DC      X'031C',X'020C',X'031C',X'030C',X'031C',X'030C'         581.
 0003AE 031C031C030C031C            395          DC      X'031C',X'031C',X'030C',X'031C',X'030C',X'031C'         582.
 000018                             396 MLEN     EQU     *-MDAY                LENGTH OF MONTH NAME              583.
 0003BA 0007000800050005            397          DC      AL2(7,8,5,5,3,4,4,6,9,7,8,8)                            584.
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                                    22 DEC 69
000000                             398 MSTRING  DS      0H                      MONTH NAME (EBCDIC)              585.
000002 C1C1D5E4C1D9E8C6            399          DC      C'JANUARY',C'FEBRUARY',C'MARCH',C'APRIL',C'MAY'          586.
000012 D1E4D5C5D1E4D3E8            400          DC      C'JUNE',C'JULY',C'AUGUST',C'SEPTEMBER',C'OCTOBER'        587.
000022 D5D6E5C5D4C2C5D9            401          DC      C'NOVEMBER',C'DECEMBER'                                  588.
```

```
 LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                                    22 DEC 69
                                   403 ************************************************************************** 590.
                                   404 *     THE FOLLOWING CODE ANALYZES TIME AND LINE LIMITS ON CONT CARD    * 591.
                                   405 *       R4 = RETURN ADDRESS, XTLIM := TIME, XLLIM := LINES             * 592.
                                   406 ************************************************************************** 593.

 000410                            408 ANALYZE  DS     0H                                                       595.
 00000A                            409 STDTIME  EQU    10              DEFAULT TIME LIMIT (SECONDS)             596.
 0001F4                            410 STDLINES EQU    500             DEFAULT LINE LIMIT (LINES)               597.

 00041C 9200 C856      00856       412          MVI    COLON,0         RESET STATE FLAGS                        599.
 000420 9200 C857      00857       413          MVI    COMMA,0                                                  600.
 000424 1B00               *       414          SR     0,0                                                      601.
 000426 1B55               *       415          SR     5,5             LINES ACCUMULATOR                        602.
 000428 1B66               *       416          SR     6,6             MINUTES ACCUMULATOR                      603.
 00042A 1B77               *       417          SR     7,7             SECONDS ACCUMULATOR                      604.
 00042C 4120 CC48          00C48   418          LA     2,CONBUFF+8                                              605.
 000430 4120 2001          00001   419 AN1      LA     2,1(,2)         STEP SCAN POINTER                        606.
 000434 5920 C50C          0050C   420          C      2,=A(CONBUFF+28)                                         607.
 000438 4720 C4AA          004AA   421          BH     AN6                                                      608.
 00043C 9540 2000      00000       422          CLI    0(2),C' '       SKIP BLANKS                              609.
 000440 4780 C430          00430   423          BE     AN1                                                      610.
 000444 95F0 2000      00000       424          CLI    0(2),C'0'       ACCUMULATE DIGIT STRING                  611.
 000448 4740 C482          00482   425          BL     AN4                                                      612.
 00044C 940F 2000      00000       426          NI     0(2),X'0F'                                               613.
 000450 4300 2000          00000   427          IC     0,0(,2)                                                  614.
 000454 95FF C857      00857       428          CLI    COMMA,X'FF'                                              615.
 000458 4770 C466          00466   429          BNE    AN2                                                      616.
 00045C 4C50 C51E          0051E   430          MH     5,=H'10'        COMMA SEEN - LINES                       617.
 000460 1A50               *       431          AR     5,0                                                      618.
 000462 47F0 C430          00430   432          B      AN1                                                      619.
 000466 95FF C856      00856       433 AN2      CLI    COLON,X'FF'                                              620.
 00046A 4770 C478          00478   434          BNE    AN3                                                      621.
 00046E 4C70 C51E          0051E   435          MH     7,=H'10'        COLON SEEN - SECONDS                     622.
 000472 1A70               *       436          AR     7,0                                                      623.
 000474 47F0 C430          00430   437          B      AN1                                                      624.
 000478 4C60 C51E          0051E   438 AN3      MH     6,=H'10'        OTHERWISE - MINUTES                      625.
 00047C 1A60               *       439          AR     6,0                                                      626.
 00047E 47FC C430          00430   440          B      AN1                                                      627.
 000482 957A 2000      00000       441 AN4      CLI    0(2),C':'                                                628.
 000486 4770 C496          00496   442          BNE    AN5                                                      629.
 00048A 9300 C856      00856       443          TS     COLON           SET COLON SWITCH                         630.
 00048E 4770 C4AA          004AA   444          BNZ    AN6                                                      631.
 000492 47F0 C430          00430   445          B      AN1                                                      632.
 000496 956B 2000      00000       446 AN5      CLI    0(2),C','                                                633.
 00049A 4770 C4AA          004AA   447          BNE    AN6                                                      634.
 00049E 92FF C856      00856       448          MVI    COLON,X'FF'     COMMA IMPLIES COLON                      635.
 0004A2 9300 C857      00857       449          TS     COMMA           SET COMMA SWITCH                         636.
 0004A6 4780 C430          00430   450          BZ     AN1                                                      637.

 0004AA 4C60 C520          00520   452 AN6      MH     6,=H'60'        CONVERT TO SECONDS                       639.
 0004AE 1A76               *       453          AR     7,6                                                      640.
 0004B0 4770 C4B8          004B8   454          BNZ    AN7                                                      641.
 0004B4 5870 C510          00510   455          L      7,=A(STDTIME)   IF ZERO, DEFAULT                         642.
 0004B8 5C60 C514          00514   456 AN7      M      6,=F'38400'     CONVERT TO OS TIMER UNITS                643.
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                                    644.
000C4BC 5070 C824           00824   457          ST      7,XTLIM                                               644.
000C4C0 1255                00000   458          LTR     5,5             SET LINES REGISTER                    645.
000C4C2 4770 C4CA           004CA   459          BNZ     AN8                                                   646.
000C4C6 5850 C518           00518   460          L       5,=A(STDLINES)  IF ZERO, DEFAULT                      647.
000C4CA 5050 C828           00828   461 AN8      ST      5,XLLIM                                               648.
000C4CE C7F4                        462          BR      4                                                     649.

000C4D0                             464          LTORG                                                         651.
000C4D0 4040404040404040            465                  =CL8' '
000C4D8 000CC704                    466                  =A(SAVE)
000C4DC 00000000                    467                  =A(COMMON)
000C4E0 00000898                    468                  =A(GETCARD)
000C4E4 00000800                    469                  =A(SS)
000C4E8 00016000                    470                  =A(L+WMIN+CMIN)
000C4EC 00006000                    471                  =A(CMIN)
000C4F0 00002800                    472                  =A(WMAX-WMIN)
000C4F4 00FFFFF8                    473                  =XL4'00FFFFF8'
000C4F8 00010000                    474                  =A(L+WMIN)
000C4FC 7FFFFFFF                    475                  =XL4'7FFFFFFF'
000C500 FFFFFFF0                    476                  =F'-16'
000C5C4 000C0C00                    477                  =V(ENDLOAD)
000C5C8 000009DC                    478                  =A(PUTLINE)
000C50C 00000C5C                    479                  =A(CONBUFF+28)
000C510 0000000A                    480                  =A(STDTIME)
000C514 00009600                    481                  =F'38400'
000C518 000001F4                    482                  =A(STDLINES)
000C51C F1F9                        483                  =C'19'
000C51E 000A                        484                  =H'10'
000C520 003C                        485                  =H'60'
                                    486          DROP    11                                                    652.
                                    487          DROP    12                                                    653.
```

```
 LCC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                        22 DEC 69
                                   489 ********************************************************************** 655.
                                   490 *     THE FOLLOWING CODE IS ENTERED FOR PROGRAM INTERRUPTIONS        * 656.
                                   491 ********************************************************************** 657.

000004                             493 RETA     EQU     4                   DISPL OF RETURN ADDR IN BLK MARK 659.


000522                             495 SPIEEXIT DS      0H                  INTERRUPTION DECODER             661.

000522                             497          USING   SPIEEXIT,15                                          663.
000000                             498          USING   PIE,1                                                664.
000522 90E2 F04A         0056C     499          STM     14,2,PISAVE         SAVE REGISTERS                   665.
000526 95FF F333  00855            500          CLI     TIMEOUT,X'FF'       TEST FOR TIMER FORCED EXIT       666.
00052A 4780 F036         00558     501          BE      SP1                 IF SO, TERMINATE                 667.
00052E 903C F05E         00580     502          STM     3,12,REGSAVE                                         668.
000532 58E0 100C         0000C     503          L       14,014              RESTORE R14                      669.
000536 1800                        504          SR      0,0                 INDICATE ENTRY FROM DECODER      670.
000538 4170 0004         00004     505          LA      7,4                 INDICATE PROGRAM CHECK           671.
00053C 4180 1004         00004     506          LA      8,OLDPSW                                             672.
000540 58F0 F692         00BB4     507          L       15,AERROR           LINK TO ERROR PROCESSOR          673.
                                   508          DROP    15                                                   674.
000544 052F                        509          BALR    2,15                                                 675.
000546                             510          USING   *,2                                                  676.
000546 58F0 2062         005A8     511          L       15,=A(SPIEEXIT)     REESTABLISH ADDRESSING           677.
000522                             512          USING   SPIEEXIT,15                                          678.
                                   513          DROP    2                                                    679.
00054A 983C F05E         00580     514          LM      3,12,REGSAVE                                         680.
00054E 1200                        515          LTR     0,0                 TEST FOR CONTINUED EXECUTION     681.
000550 4780 F042         00564     516          BZ      SP2                                                  682.
000554 5810 F056         00578     517          L       1,PISAVE+12         TERMINATE                        683.
000558 5800 C004         00004     518 SP1      L       0,RETA(,12)         SET RESUME ADDR, LINK REGISTER   684.
00055C 5000 1008         00008     519          ST      0,OLDPSW+4            RETURN ADDRESS IN              685.
000560 5000 1018         00018     520          ST      0,01                  OUTERMOST BLOCK MARK           686.
000564 98E2 F04A         0056C     521 SP2      LM      14,2,PISAVE                                          687.
000568 07FE                        522          BR      14                  RETURN TO OS                     688.
                                   523          DROP    15                                                   689.
                                   524          DROP    1                                                    690.

00056C                             526 PISAVE   DS      5F                                                   692.
000580                             527 REGSAVE  DS      10F                                                  693.

000000                             529 PIE      DSECT                                                        695.
000000                             530 PICADR   DS      F                   PICA ADDRESS                     696.
000004                             531 OLDPSW   DS      2F                  OLD PSW                          697.
00000C                             532 OLDREG   DS      5F                  OLD REGISTER CONTENTS            698.
00000C                             533 014      EQU     OLDREG+0              R14                            699.
000018                             534 01       EQU     OLDREG+12             R1                             700.

000000                             536 AWXGDR01 CSECT                                                        702.
0005A8                             537          LTORG                                                        703.
0005A8 00000522                    538                  =A(SPIEEXIT)
```

```
 LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                        22 DEC 69
                                   540 ************************************************************************  705.
                                   541 *    THE FOLLOWING CODE IS ENTERED ON EXPIRATION OF EXECUTION TIME    *  706.
                                   542 ************************************************************************  707.

0005AC                             544 TIMEEXIT DS      0H                    STIMER EXIT ROUTINE               709.

0005AC                             546          USING   TIMEEXIT,15                                             711.
0005AC 9300 F6E9      00C55        547          TS      OUTSYNCH              DEFER PROCESSING IF IN PUTCARD    712.
0005B0 4780 F0C0            0066C  548          BZ      TT9                     (NEEDED BY ERROR PROCESSOR)     713.
0005B4 900E F0D4            00680  549          STM     0,14,TSAVE            SAVE REGISTERS                    714.
0005B8 58D0 F60C            00BB8  550          L       13,ARUNDSEG           LOAD RUN-TIME FIXED DATA BASE     715.
0005BC D247 F110 D000 006BC 00000  551          MVC     TXSAVE(18*4),0(13)    PROTECT SAVE AREA CONTENTS        716.

0005C2 5810 0010            00010  553          L       1,ACVT                TRACK POINTERS TO RB CHAIN        718.
0005C6 5810 1000            00000  554          L       1,CVTTCBP(,1)                                           719.
0005CA 5810 1004            00004  555          L       1,CURRTCBP(,1)                                          720.
0005CE 5810 1000            00000  556          L       1,TCBRBP(,1)                                            721.

0005D2 5820 101C            0001C  558 TT1       L       2,RBLINK-1(,1)        R1 = RB ADDR, R2 := PREV RB ADDR  723.
0005D6 91C0 200A      0000A        559          TM      RBSTAB(2),PRBMASK     SEARCH FOR FIRST PRB IN CHAIN     724.
0005DA 4780 F038            005E4  560          BZ      TT2                     (BITS 0,1 = 00)                 725.
0005DE 1812                        561          LR      1,2                   CHAIN TO NEXT RB                  726.
0005E0 47F0 F026            005D2  562          B       TT1                                                     727.

0005E4 D2C7 F68C 2010 00C38 00010  564 TT2       MVC     TIMEPSW(8),RBOPSW(2)  SAVE OLD PSW                      729.
0005EA 59D0 1054            00054  565          C       13,RBGRSAVE+13*4(,1)  TEST FOR CORRECT REGISTER SET     730.
0005EE 4770 F04E            005FA  566          BNE     TT3                                                     731.
0005F2 986E 1038            00038  567          LM      6,14,RBGRSAVE+6*4(1)  MATCH - RESTORE ADDR REGISTERS    732.
0005F6 47F0 F052            005FE  568          B       TT4                                                     733.
0005FA 58E0 F0CC            00678  569 TT3       L       14,=F'-1'             NO MATCH - INHIBIT CARD SEARCH    734.
0005FE 1B00                        570 TT4       SR      0,0                   CALL ERROR PROCESSOR              735.
000600 4170 0002            00002  571          LA      7,2                                                     736.
000604 4180 F68C            00C38  572          LA      8,TIMEPSW                                               737.
000608 58F0 F608            00BB4  573          L       15,AERROR                                               738.
                                   574          DROP    15                                                      739.
00060C 052F                        575          BALR    2,15                                                    740.
00060E                             576          USING   *,2                   RESTORE ADDRESSING                741.
00060E 58F0 206E            0067C  577          L       15,=A(TIMEEXIT)                                         742.
0005AC                             578          USING   TIMEEXIT,15                                             743.
                                   579          DROP    2                                                       744.
000612 92FF F2A9      00855        580          MVI     TIMEOUT,X'FF'         SET SWITCH FOR SPIE EXIT          745.
000616 9812 F26C            00818  581          LM      1,2,SDVLIST           (R1,R2) := COMPILED CODE BOUNDS   746.
00061A 1A21                        582          AR      2,1                                                     747.
00061C 5830 F690            00C3C  583          L       3,TIMEPSW+4           RECOVER INTERRUPTION ADDRESS      748.
000620 4130 3000            00000  584          LA      3,0(,3)                                                 749.
000624 1931                        585          CR      3,1                   TEST IF WITHIN COMPILED CODE      750.
000626 4740 F08C            00638  586          BL      TT5                                                     751.
00062A 1932                        587          CR      3,2                                                     752.
00062C 4720 F08C            00638  588          BH      TT5                                                     753.
000630 9200 3000      00000        589          MVI     0(3),X'00'            IF SO, REPLACE WITH ILLEGAL OP    754.
000634 47F0 F0B6            00662  590          B       TT8                                                     755.

000638 D7C7 1000 1000 00000 00000  592 TT5       XC      0(8,1),0(1)           OTHERWISE, SET ALL COMPILED CODE  757.
00063E 41C0 1008            00008  593          LA      0,8(,1)                 TO ILLEGAL OP, FORCE CHECK      758.
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                                    22 DEC 69
000642 1B20                         594           SR      2,0                                                      759.
000644 4100 0100              00100 595           LA      0,256                                                    760.
000648 1920                         596 TT6       CR      2,0                                                      761.
00064A 47DO F0B0              0065C 597           BNH     TT7                                                      762.
00064E D2FF 1008 1000 00008 00000 598           MVC     8(256,1),0(1)                                            763.
000654 1A10                         599           AR      1,0                                                      764.
000656 1B20                         600           SR      2,0                                                      765.
000658 47F0 F09C              00648 601           B       TT6                                                      766.
00065C 0620                         602 TT7       BCTR    2,0                                                      767.
00065E 4420 F0C2              0066E 603           EX      2,MOVEZERO                                               768.

000662 D247 D000 F110 00000 006BC 605 TT8       MVC     0(18*4,13),TXSAVE     RESTORE                            770.
000668 980E F0D4              00680 606           LM      0,14,TSAVE                                               771.
00066C C7FE                         607 TT9       BR      14                    RETURN TO OS                       772.

00066E D200 1008 1000 00008 00000 609 MOVEZERO  MVC     8(0,1),0(1)           EXECUTED                           774.
000678                              610           LTORG                                                          775.
000678 FFFFFFFF                     611                 =F'-1'
00067C 000005AC                     612                 =A(TIMEEXIT)

000680                              614 TSAVE     DS      15F                                                      777.
0006BC                              615 TXSAVE    DS      18F                                                      778.

                                    617 * THE FOLLOWING DESCRIBE OS/360 CONTROL BLOCKS (CF. C28-6628)          *  780.
000010                              618 ACVT      EQU     16                    POINTER TO CVT (ABSOLUTE)          781.
000000                              619 CVTTCBP   EQU     0                     DISPL OF TCB QUEUE PTR IN CVT      782.
000004                              620 CURRTCBP  EQU     4                     DISPL OF CURR TCB PTR IN QUEUE     783.
000000                              621 TCBRBP    EQU     0                     DISPL OF CURR RB PTR IN TCB        784.
00001D                              622 RBLINK    EQU     29                    DISPL OF RB LINK IN RB             785.
00000A                              623 RBSTAB    EQU     10                    DISPL OF RB TYPE BITS IN RB        786.
000000                              624 PRBMASK   EQU     X'C0'                 MASK FOR TYPE BITS IN RBSTAB       787.
000010                              625 RBOPSW    EQU     16                    DISPL OF OLD PSW IN RB             788.
000020                              626 RBGRSAVE  EQU     32                    DISPL OF OLD GPR IN RB (IRB,ETC)   789.
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                                    22 DEC 69
000704                           628 SAVE     DS      18F                    SAVE AREA PROVIDED BY MONITOR      791.
                                 629 DMGLIST  OPEN    (SYSIN,(INPUT),        DCB LIST FOR OPEN AND CLOSE    1 792.
                                                      SYSOUT,(OUTPUT)),                                     2 793.
                                                      MF=L                                                    794.
                                 635 SMGLIST  FREEMAIN V,A=SDVLIST,SP=0,MF=L                                  795.
000750                           640 LIBDIR   DS      0H                     BLDL LIST FOR LINK AND LOAD        796.
000750 0003003A                  641          DC      AL2(3,58)              LIST DESCRIPTOR                    797.
000762 C1E6E7C3D4D7C1F1          642 DPHASEA  DC      CL8'AWXCMPA1',25H'0'   PHASE A                            798.
00079C C1E6E7C3D4D7C2F1          643 DPHASEB  DC      CL8'AWXCMPB1',25H'0'   PHASE B                            799.
0007D6 C1E6E7D3C9C2D9F1          644 DRUNLIB  DC      CL8'AWXLIBR1',25H'0'   RUN-TIME LIBRARY                   800.
000810 0001680000093000          645 ALLCORE  DC      A(COREMIN,COREMAX)     LIMITS TO OBTAIN ALL FREE CORE     801.
000818                           646 SDVLIST  DS      2F                     STORAGE DOPE VECTOR (GET/FREE)     802.
000818                           647 SDVBASE  EQU     SDVLIST+0              STORAGE AREA BASE ADDRESS          803.
00081C                           648 SDVLEN   EQU     SDVLIST+4              STORAGE AREA LENGTH                804.
000820                           649 OLDSPIE  DS      F                      OLD SPIE INFORMATION               805.

000824                           651 XTLIM    DS      F                      EXECUTION TIME LIMIT               807.
000828                           652 XLLIM    DS      F                      EXECUTION LINE LIMIT               808.
00082C                           653 TIME     DS      F                      SYSTEM TIME '000HHMM&'             809.
000830                           654 DATE     DS      F                      SYSTEM DATE '00YYDDD&'             810.
000831                           655 DYEAR    EQU     DATE+1                                                    811.
000832                           656 DDAY     EQU     DATE+2                                                    812.
000834                           657 RUNID    DS      CL32                   SYSTEM IDENTIFICATION              813.
000854                           658 NOLOAD   DS      X                      NON-ZERO IFF LIBRARY NOT LOADED    814.
000855                           659 TIMEOUT  DS      X                      SET IF TIME INTERVAL EXPIRED       815.
000856                           660 COLON    DS      X                      SCAN SWITCHES                      816.
000857                           661 COMMA    DS      X                                                         817.

000858                           663 ENTVECT  DS      0F                     ENTRY VECTOR PASSED TO COMPILER    819.
000858                           664 COMMLIM  DS      2F                        COMMON LIMITS                   820.
000858                           665 XEOF     EQU     COMMLIM+0                 EXECUTION END-OF-FILE           821.
000859                           666 XTRACE   EQU     COMMLIM+1                 EXECUTION TRACE OPTION          822.
00085C                           667 LINELIM  EQU     COMMLIM+4                 EXECUTION LINE LIMIT            823.
000860 000009DC                  668          DC      A(PUTLINE)                SERVICE ROUTINE ENTRY POINTS    824.
000864 00000898                  669          DC      A(GETCARD)                                                825.
000868 00000A3A                  670          DC      A(PUTCARD)                                                826.
00086C 00000A4A                  671          DC      A(GETMAIN)                                                827.
000870 00000A72                  672          DC      A(FREEMAIN)                                               828.
000874 00000A96                  673          DC      A(GETTIME)                                                829.
000878 00000AAE                  674          DC      A(GETCLOCK)                                               830.
00087C 0000000000000000          675          DC      6A(0)                  RESERVED FOR EXPANSION             831.
000894 00000834                  676          DC      A(RUNID)               SYSTEM IDENTIFICATION              832.

                                 678 *    THE FOLLOWING SECTION CONTAINS THE VARIABLES SHARED BY THE       *  834.
                                 679 *    MONITOR AND THE PL360 LOADER PROCEDURES                          *  835.

000000                           681          COM                                                              837.

000000                           683 COMMON   DS      0D                     COMMON DATA SEGMENT                839.
000000                           684 ESDTABLE DS      256F                   GLOBAL SYMBOL DICTIONARY           840.
000400                           685 ESDINDEX DS      F                      ESD TABLE INDEX                    841.
000404                           686 LOCORE   DS      F                      LOAD AREA BOUNDS                   842.
000408                           687 HICORE   DS      F                                                         843.
00040C                           688 LOADBASE DS      F                      LOWEST LOADED ADDRESS              844.
```

```
 LCC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT
000410                            689 ENTRYPT  DS      F       LOADED PROGRAM ENTRY POINT          845.
000414                            690 ENTRYSET DS      X       SET IFF ENTRY ADDR VALID            846.
000415                            691 INIT     DS      X       SET TO INITIALIZE LOADER TABLES     847.
000416                            692 LOADERR  DS      X       NON-ZERO IFF LOADING ERROR          848.
000417                            693 BLANK    DS      C                                           849.
000418                            694 ERRBUF   DS      CL132   ASSEMBLY AREA FOR ERROR MESSAGES    850.

000000                            696 AWXGDR01 CSECT                                               852.
```

```
 LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                  22 DEC 69
                                   698 ********************************************************************************  854.
                                   699 *                              GETCARD                                         *  855.
                                   700 *  SUPPLY 80 CHARACTER INPUT RECORD, DETECT SYSTEM CONTROL CARDS               *  856.
                                   701 *   R0 = ADDRESS OF RECORD DESTINATION, CC := 1 FOR END-OF-FILE                *  857.
                                   702 ********************************************************************************  858.

 000898                            704 GETCARD  DS      0H                    GET CARD IMAGE                          860.

 000898                            706          USING   GETCARD,3                                                     862.
 000898 90E6 3324        00BBC     707          STM     14,6,SSAVE            SAVE REGISTERS                          863.
 00089C 1820                       708          LR      2,0                   SAVE DESTINATION ADDRESS                864.
 00089E 95FF 33FA  00C92           709 READ0    CLI     ENDDS,X'FF'           TEST FOR PREVIOUS END OF DS             865.
 0008A2 4770 3016        008AE     710          BNE     READ1                                                         866.
 0008A6 58C0 3228        00AC0     711          L       12,=A(AWXGDR01)       REESTABLISH ADDRESSING                  867.
 000000                            712          USING   AWXGDR01,12                                                  868.
 0008AA 47F0 C2D6        002D6     713          B       CLEANUP               TERMINATE SUBSYSTEM                     869.
                                   714          DROP    12                                                            870.

 0008AE 4860 3314        00BAC     716 READ1    LH      6,ISTATE              FETCH STATE OF INPUT SYSTEM             872.
 0008B2 1266                       717          LTR     6,6                   TEST FOR STRAIGHT READ                  873.
 0008B4 4780 3028        008C0     718          BZ      RDR1                                                          874.
 0008B8 4960 3234        00ACC     719          CH      6,=AL2(SB)            TEST FOR STREAM B                       875.
 0008BC 4770 3044        008DC     720          BNE     RDR2                                                          876.

 000800 1802                       722 RDR1     LR      0,2                   RESTORE R0                              878.
                                   723          GET     SYSIN,(0)             GET NEXT RECORD                         879.
 0008CC 95FF 33FC  00C94           727          CLI     REREAD,X'FF'          TEST FOR SUCCESSFUL TRANSFER            880.
 000800 4770 30CC        00964     728          BNE     READ2                   (SET BY EXIT ROUTINE)                 881.
 0008D4 9200 33FC  00C94           729          MVI     REREAD,0                                                      882.
 0008D8 47F0 3028        008C0     730          B       RDR1                  REISSUE GET                             883.

 0008DC 1842                       732 RDR2     LR      4,2                   UNPACK CARD IMAGE FROM MODULE           885.
 0008DE 5850 3310        00BA8     733          L       5,INEXT               R4 = DESTINATION, R5 = SOURCE           886.
 0008E2 9240 4000  00000           734          MVI     0(4),C' '             BLANK DESTINATION                       887.
 0008E6 D24E 4001 4000 00001 00000 735          MVC     1(79,4),0(4)                                                  888.
 0008EC 1B00                       736          SR      0,0                                                           889.
 0008EE 4300 5000        00000     737          IC      0,0(,5)               R0 = PACKED CHAR COUNT FOR LINE         890.
 0008F2 1200                       738          LTR     0,0                   TERMINATED BY 0                         891.
 0008F4 4770 3072        00C0A     739          BNZ     RDR3                                                          892.
 0008F8 92FF 33FA  00C92           740          MVI     ENDDS,X'FF'           SET FLAGS                               893.
 0008FC 92FF 33F9  00C91           741          MVI     EOF,X'FF'                                                     894.
                                   742          DELETE  EPLOC=EPNAME                                                  895.
 000906 47F0 3102        0099A     745          B       READ5                 EXIT                                    896.
 00090A 1A05                       746 RDR3     AR      0,5                   STEP BUFFER LINE POINTER                897.
 00090C 5000 3310        00BA8     747          ST      0,INEXT                                                       898.
 000910 1B11                       748          SR      1,1                                                           899.
 000912 4150 5001        00001     749          LA      5,1(,5)               STEP PAST CHAR COUNT                    900.

 000916 1950                       751 RDR4     CR      5,0                   TEST FOR COMPLETION OF LINE             902.
 000918 4780 30B2        00A4A     752          BE      RDR7                                                          903.
 00091C 4310 5000        00000     753          IC      1,0(,5)                                                       904.
 000920 5910 322C        00AC4     754          C       1,=XL4'80'            BLANK COUNT IF BIT 0 = 0                905.
 000924 47B0 3098        00930     755          BNL     RDR5                                                          906.
 000928 4150 5001        00001     756          LA      5,1(,5)               STEP SOURCE POINTER                     907.
```

```
  LCC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT
000920 47F0 30A4             0093C   757          B       RDR6                                                    908.
000930 5B10 322C             00AC4   758 RDR5     S       1,=XL4'80'          CHAR COUNT IF BIT 0 = 1             909.
000934 4410 30AC             00944   759          EX      1,RDRMOVE           MOVE CHAR STRING                    910.
000938 4151 5002             00002   760          LA      5,2(1,5)            STEP SOURCE POINTER                 911.
00093C 4141 4001             00001   761 RDR6     LA      4,1(1,4)            STEP DESTINATION POINTER            912.
000940 47F0 307E             00916   762          B       RDR4                                                    913.

000944 D200 4000 5001 00000  00001   764 RDRMOVE   MVC     0(0,4),1(5)         EXECUTED                            915.

00094A 4960 3236             00ACE   766 RDR7     CH      6,=AL2(SA1)         TEST FOR STATE A1                   917.
00094E 4770 30CC             00964   767          BNE     READ2                                                   918.
000952 D506 2000 3238 00000  00AD0   768          CLC     0(7,2),=C'$INSERT'     AND '$INSERT' CARD               919.
000958 4770 30CC             00964   769          BNE     READ2                                                   920.
00095C 4160 0002             00002   770          LA      6,SB                SET STATE TO 6                      921.
000960 47F0 3028             008C0   771          B       RDR1                   AND READ CARD RECORD             922.

000964 4060 3314             00BAC   773 READ2     STH     6,ISTATE            SAVE STATE                          924.
000968 956C 2000       00000         774          CLI     0(2),C'%'           TEST FOR POSSIBLE CONTROL CARD      925.
00096C 4770 3102             0099A   775          BNE     READ5                                                   926.
000970 D504 2001 323F 00001  00AD7   776          CLC     1(5,2),=C'ALGOL'    CAN BE  "%ALGOL",                   927.
000976 4770 30F4             0098C   777          BNE     READ4                                                   928.
00097A 92FF 33F8       00C90         778 READ3     MVI     CONTCARD,X'FF'      INDICATE CONTROL CARD FOUND         929.
00097E D24F 33A8 2000 00C40  00000   779          MVC     CONBUFF(80),0(2)    SAVE CARD IMAGE                     930.
000984 92FF 33F9       00C91         780          MVI     EOF,X'FF'           SET JOB END-OF-FILE                 931.
000988 47FC 3102             0099A   781          B       READ5                                                   932.
00098C D502 2001 3244 00001  00ADC   782 READ4     CLC     1(3,2),=C'EOF'      TEST FOR "%EOF"                     933.
000992 4770 3102             0099A   783          BNE     READ5                                                   934.
000996 92FF 33F9       00C91         784          MVI     EOF,X'FF'           SET JOB END-OF-FILE                 935.

00099A 9500 33F9       00C91         786 READ5     CLI     EOF,0               SET CONDITION CODE                  937.
00099E 98E6 3324             00BBC   787          LM      14,6,SSAVE          RESTORE REGISTERS                   938.
0009A2 07F2                  00000   788          BR      2                                                       939.

0009A4 4960 3234             00ACC   790 ENDRDR    CH      6,=AL2(SB)          OS END OF DS EXIT ROUTINE           941.
0009A8 4770 311C             009B4   791          BNE     ENDRDR1                                                 942.
0009AC 4160 0003             00003   792          LA      6,SA2               IF STATE = B, SET STATE = A2        943.
0009B0 47F0 3044             00C8DC  793          B       RDR2                   AND UNPACK CARD FROM BUFFER      944.
0009B4 92FF 33F9       00C91         794 ENDRDR1   MVI     EOF,X'FF'           OTHERWISE, SET FLAGS AND EXIT       945.
0009B8 92FF 33FA       00C92         795          MVI     ENDDS,X'FF'         SET JOB AND SYSTEM EOF              946.
0009BC 47F0 3102             0099A   796          B       READ5                                                   947.
                                     797          DROP    3                                                       948.

0009C0                               799 ADCBEXIT  DS      0F                  DCB EXIT LIST (SYSIN)               950.
0009C0 850009C4                      800          DC      X'85',AL3(DCBEXIT)  EXIT ROUTINE FOR CONCATENATION      951.
0009C4                               801          USING   DCBEXIT,15                                              952.
0009C4 9608 F14C       00B10         802 DCBEXIT   OI      SYSIN+OFLGS,X'08'   SET UNLIKE ATTRIBUTES BIT         953.
0009C8 9500 F2CF       00C93         803          CLI     RDRSTAT,0           TEST FOR INITIAL ENTRY              954.
0009CC 4770 F012             009D6   804          BNE     SETRETRY                                                955.
0009D0 92FF F2CF       00C93         805          MVI     RDRSTAT,X'FF'       INITIAL OPEN - SET STATUS           956.
0009D4 07FE                  00000   806          BR      14                                                      957.
0009D6 92FF F2D0       00C94         807 SETRETRY  MVI     REREAD,X'FF'        NOT INITIAL OPEN - SET REREAD       958.
0009DA 07FE                  00000   808          BR      14                   (CONCATENATION - NO TRANSFER)      959.
                                     809          DROP    15                                                      960.
```

```
 LCC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT                                                22 DEC 69
                                    811 ********************************************************************************  962.
                                    812 *                              PUTLINE                           *  963.
                                    813 *  ACCEPT 132 CHARACTER OUTPUT STRING AND USASI CONTROL CODE       *  964.
                                    814 *   R0 = ADDRESS OF STRING,  R1 = CONTROL CHARACTER                *  965.
                                    815 ********************************************************************************  966.


 CCC9DC                             817 PUTLINE  DS      0H                  PRINT LINE IMAGE                  968.

 0009DC                             819          USING   PUTLINE,3                                             970.
 CCC9DC 9200 3289       00C95       820          MVI     OUTSYNCH,0          INDICATE ENTRY TO WRITE           971.
 0009E0 90E2 31E0             00BBC 821          STM     14,2,SSAVE                                            972.
 0009E4 1820                        822          LR      2,0                 SAVE ADDRESS                      973.
                                    823          PUT     SYSOUT              LOCATE NEXT BUFFER, R1 := ADDR    974.
 0009F0 D283 1001 2000 00001 00000  827          MVC     1(132,1),0(2)       MOVE LINE IMAGE                   975.
 0009F6 5820 31EC             00BC8 828          L       2,SSAVE+12          PREFIX CONTROL CHARACTER          976.
 0009FA 4220 1000             00000 829          STC     2,0(,1)                                               977.
 0009FE 9300 32BA       00C96       830          TS      XMIT                                                  978.
 000AC2 4770 3034             00A10 831          BNZ     WRITE0              INSERT MODULE NAME IN HEADER      979.
 000AC6 9260 1027       00027       832          MVI     39(1),C'-'                                            980.
 000A0A D207 102A 31C4 0002A 00BA0  833          MVC     42(8,1),EPNAME                                        981.
 000A10 98E2 31E0             00BBC 834 WRITE0   LM      14,2,SSAVE                                            982.
 000A14 9300 3289       00C95       835          TS      OUTSYNCH            INDICATE EXIT                     983.
 000A18 4770 3042             00A1E 836          BNZ     WRITE1              IF SET, TIME-OUT DURING WRITE     984.
 000A1C 07F2                        837          BR      2                                                     985.

 000A1E 1B00                        839 WRITE1   SR      0,0                 CALL ERROR PROCESSOR              987.
 000A20 4170 0002             00002 840          LA      7,2                                                   988.
 000A24 D203 3260 31F0 00C3C 00BCC  841          MVC     TIMEPSW+4(4),SSAVE+16 CONSTRUCT PSEUDO-PSW            989.
 000A2A 4180 325C             00C38 842          LA      8,TIMEPSW                                             990.
 000A2E 58F0 31D8             00BB4 843          L       15,AERROR                                             991.
 000A32 052F                        844          BALR    2,15                                                  992.
 000A34 5810 C004             00004 845          L       1,RETA(,12)         GET RETURN ADDR FROM OUTER BLOCK  993.
 000A38 07F1                        846          BR      1                   TERMINATE EXECUTION               994.
                                    847          DROP    3                                                     995.


                                    849 ********************************************************************************  997.
                                    850 *                              PUTCARD                           *  998.
                                    851 *  ACCEPT 80 CHARACTER OBJECT OUTPUT STRING                       *  999.
                                    852 *   R0 = ADDRESS OF STRING, R1 = ADDRESS OF HIGHEST COMMON IN USE  * 1000.
                                    853 ********************************************************************************  1001.


 000A3A                             855 PUTCARD  DS      0H                  PUNCH CARD IMAGE                  1003.

 000A3A                             857          USING   PUTCARD,3                                             1005.
 000A3A 90BF 3182             00BBC 858          STM     11,15,SSAVE         SAVE REGISTERS                    1006.
 000A3E 58F0 308E             00AC8 859          L       15,=V(LOADCARD)     PROCESS CARD IMAGE                1007.
 000A42 05EF                        860          BALR    14,15                 (R11, R14, R15 DESTROYED)       1008.
 000A44 98BF 3182             00BBC 861          LM      11,15,SSAVE                                           1009.
 000A48 07F2                        862          BR      2                                                     1010.
                                    863          DROP    3                                                     1011.
```

```
                                865 ********************************************************************************* 1013.
                                866 *                              GETMAIN                                          * 1014.
                                867 *  SUPPLY VARIABLE-LENGTH FREE STORAGE AREA                                     * 1015.
                                868 *   R0,R1 = MINIMUM, MAXIMUM LENGTH, R0 := ADDRESS, R1 := LENGTH                * 1016.
                                869 ********************************************************************************* 1017.

000A4A                          871 GETMAIN  DS      0H                      OBTAIN FREE STORAGE                  1019.

000A4A                          873          USING   GETMAIN,3                                                    1021.
000A4A 90DF 3172        00BBC   874          STM     13,15,SSAVE                                                  1022.
000A4E 41D0 3196        00BE0   875          LA      13,XSAVE                SUPPLY SAVE AREA                     1023.
000A52 9001 31DE        00C28   876          STM     0,1,SREQUEST                                                 1024.
                                877          GETMAIN VU,LA=SREQUEST,A=SDESCRIP,SP=0   ISSUE GETMAIN               1025.
000A68 9801 31E6        00C30   885          LM      0,1,SDESCRIP            RETURN STORAGE DESCRIPTION           1026.
000A6C 98DF 3172        00BBC   886          LM      13,15,SSAVE                                                  1027.
000A70 07F2                     887          BR      2                                                            1028.
                                888          DROP    3                                                            1029.


                                890 ********************************************************************************* 1031.
                                891 *                             FREEMAIN                                          * 1032.
                                892 *  RETURN FREE STORAGE AREA TO SYSTEM                                           * 1033.
                                893 *    R0 = ADDRESS, R1 = LENGTH                                                  * 1034.
                                894 ********************************************************************************* 1035.

000A72                          896 FREEMAIN DS      0H                      RELEASE FREE STORAGE                 1037.

000A72                          898          USING   FREEMAIN,3                                                   1039.
000A72 90D1 314A        00BBC   899          STM     13,1,SSAVE                                                   1040.
000A76 41D0 316E        00BE0   900          LA      13,XSAVE                SUPPLY SAVE AREA                     1041.
000A7A 9001 31BE        00C30   901          STM     0,1,SDESCRIP            SET UP                               1042.
                                902          FREEMAIN V,A=SDESCRIP,SP=0        AND ISSUE FREEMAIN                 1043.
000A9C 98D1 314A        00BBC   910          LM      13,1,SSAVE                                                   1044.
000A94 07F2                     911          BR      2                                                            1045.
                                912          DROP    3                                                            1046.


                                914 ********************************************************************************* 1048.
                                915 *                              GETTIME                                          * 1049.
                                916 *  OBTAIN ELAPSED TIME IN OS TIMER UNITS (1 OS TU = 2 MACHINE TU)               * 1050.
                                917 *   R0 := ELAPSED TIME                                                          * 1051.
                                918 ********************************************************************************* 1052.

000A96                          920 GETTIME  DS      0H                      GET ELAPSED COMPILATION TIME         1054.

000A96                          922          USING   GETTIME,3                                                    1056.
000A96 90D1 3126        00BBC   923          STM     13,1,SSAVE                                                   1057.
                                924          TTIMER                                                               1058.
000A9E 5F00 311A        00BB0   927          SL      0,TIMEINIT              COMPUTE ELAPSED TIME                 1059.
000AA2 1300                     928          LCR     0,0                                                          1060.
```

```
 LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                                    
000AA4 98DF 3126           00BBC   929          LM      13,15,SSAVE                                          1061.
000AA8 5810 3136           00BCC   930          L       1,SSAVE+16                                           1062.
000AAC C7F2                        931          BR      2                                                    1063.
                                   932          DROP    3                                                    1064.


                                   934 ************************************************************************ 1066.
                                   935 *                              GETCLOCK                                * 1067.
                                   936 *  OBTAIN TIME OF DAY IN OS TIMER UNITS (1 OS TU = 2 MACHINE TU)       * 1068.
                                   937 *    R0 := TIME OF DAY                                                 * 1069.
                                   938 ************************************************************************ 1070.


000AAE                             940 GETCLOCK DS      0H                    GET TIME OF DAY                1072.

000AAE                             942          USING   GETCLOCK,3                                           1074.
000AAE 90D1 310E           00BBC   943          STM     13,1,SSAVE                                           1075.
                                   944          TIME    TU                    OBTAIN TIME OF DAY             1076.
000AB6 98DF 310E           00BBC   947          LM      13,15,SSAVE                                          1077.
000ABA 5810 311E           00BCC   948          L       1,SSAVE+16                                           1078.
000ABE C7F2                        949          BR      2                                                    1079.
                                   950          DROP    3                                                    1080.


000AC0                             952          LTORG                                                        1082.
000AC0 00000000                    953                  =A(AWXGDR01)
000AC4 00000080                    954                  =XL4'80'
000AC8 00000000                    955                  =V(LOADCARD)
000ACC 0002                        956                  =AL2(SB)
000ACE 0001                        957                  =AL2(SA1)
000AD0 5BC9D5E2C5D9E3              958                  =C'$INSERT'
000AD7 C1D3C7D6D3                  959                  =C'ALGOL'
000ADC C5D6C6                      960                  =C'EOF'
```

```
 LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT
                                   962 SYSIN    DCB     DSORG=PS,MACRF=GM,DDNAME=SYSIN,DEVD=DA,RECFM=FB,      11084.
                                                        LRECL=80,BFTEK=S,EROPT=ABE,EODAD=ENDRDR,              21085.
                                                        EXLST=ADCBEXIT                                         1086.
                                   963                *,***  IHB063  DDNAME SHORT-PADDED TO 8 CHAR

                                  1020 SYSOUT   DCB     DSORG=PS,MACRF=PL,DDNAME=SYSPRINT,DEVD=DA,RECFM=FBA,  11088.
                                                        LRECL=133,BFTEK=S,EROPT=ABE                            1089.

 000EA0                           1077 EPNAME   DS      2F                    SOURCE MODULE NAME               1091.
 000EA8                           1078 INEXT    DS      A                     SOURCE BUFFER POINTER            1092.
 000EAC                           1079 ISTATE   DS      H                     INPUT SYSTEM STATE               1093.
 000EB0                           1080 TIMEINIT DS      F                     INITIAL TIMER SETTING            1094.
 000EB4                           1081 AERROR   DS      A                     ADDR OF ALGOLRUN ERROR PROCESSOR 1095.
 000EB8                           1082 ARUNDSEG DS      A                     ADDR OF ALGOLRUN FIXED DATA SEG  1096.
 000EBC                           1083 SSAVE    DS      9F                    LOCAL SAVE AREA                  1097.
 000EE0                           1084 XSAVE    DS      18F                   LOCAL SAVE AREA (GET/FREEMAIN)   1098.
 000F28                           1085 SREQUEST DS      2F                    STORAGE REQUEST BOUNDS           1099.
 000F30                           1086 SDESCRIP DS      2F                    STORAGE ALLOCATION DOPE VECTOR   1100.
 000F38                           1087 TIMEPSW  DS      2F                    PSEUDO-PSW FOR TIMER TRAP        1101.
 000F40                           1088 CONBUFF  DS      CL80                  CONTROL CARD BUFFER              1102.
 000F90                           1089 CONTCARD DS      X                     ON IF CONBUFF FILLED             1103.
 000F91                           1090 EOF      DS      X                     READER END-OF-FILE FOR JOB       1104.
 000F92                           1091 ENDDS    DS      X                     READER END-OF-FILE FOR SYSTEM    1105.
 000F93                           1092 RDRSTAT  DS      X                                                      1106.
 000F94                           1093 REREAD   DS      X                                                      1107.
 000F95                           1094 OUTSYNCH DS      X                     SET IF PUTCARD FREE OR TIME-OUT  1108.
 000F96                           1095 XMIT     DS      X                     RESET TO DISPLAY MODULE NAME     1109.

                                  1097          DCBD    DSORG=PS,DEVD=DA      SYMBOLIC DCB DESCRIPTION         1111.

 000000                           1204 AWXGDR01 CSECT                                                          1113.
 000030                           1205 OFLGS    EQU     DCBOFLGS-IHADCB       DCB OFLGS DISPLACEMENT           1114.
 000000                           1206          END     AWXGDR01                                               1115.


NO STATEMENTS FLAGGED IN THIS ASSEMBLY
```

```
IEF285I   T123.PLLIB                                      PASSED
IEF285I   VOL SER NOS= SYS11 .
IEF285I   SYS1.MACLIB                                     KEPT
IEF285I   VOL SER NOS= SYS00 .
IEF285I   SYS2.DCLIB                                      KEPT
IEF285I   VOL SER NOS= SYS01 .
IEF285I   SYS2.DUMMYMAC                                   KEPT
IEF285I   VOL SER NOS= SYS01 .
IEF285I   SYS1.UT1                                        KEPT
IEF285I   VOL SER NOS= CAMP09.
IEF285I   SYS1.UT2                                        KEPT
IEF285I   VOL SER NOS= SYS02 .
IEF285I   SYS1.UT3                                        KEPT
IEF285I   VOL SER NOS= SYS03 .
IEF285I   SYSOUT                                          SYSOUT
IEF285I   VOL SER NOS=       .
IEF285I   LOADSET.XALGOL                                  PASSED
IEF285I   VOL SER NOS= SYS02 .
//LKED   EXEC  PGM=IEWL,PARM=(LET,LIST,MAP),COND=(5,LT,ASM)          00001200
//SYSLIN DD    DSNAME=&LOADSET,DISP=(OLD,DELETE)                     00001300
//       DD    DDNAME=SYSIN                                          00001400
//LKED.SYSLMOD DD DSNAME=SYS2.HISPEED,DISP=(OLD,KEEP)                1117.
//SYSLMOD DD  DSNAME=&GOSET(GO),UNIT=2314,SPACE=(CYL,(12,1,1)),     *00001500
//            DISP=(MOD,PASS)                                       00001600
//SYSUT1   DD  DSNAME=SYS1.UT1,DISP=OLD,DCB=(KEYLEN=0)              00001700
//SYSPRINT DD  SYSOUT=A                                             00001800
//SYSLIB   DD   DSNAME=SYS1.FORTLIB,DISP=OLD                        00001900
//         DD  DSNAME=SYS2.SSPLIB,DISP=OLD                          00002000
//         DD  DSNAME=SYS2.SUBLIB1,DISP=OLD                         00002100
//LKED.SYSIN DD *                                                   1120.
IEF236I ALLOC. FOR XALGOL   LKED     ASM
IEF237I JOBLIB    ON 434
IEF237I SYSLIN    ON 330
IEF237I           ON 00C
IEF237I SYSLMOD   ON 430
IEF237I SYSUT1    ON 1C1
IEF237I SYSLIB    ON 331
IEF237I           ON 430
IEF237I           ON 430
```

```
F128-LEVEL LINKAGE EDITOR OPTIONS SPECIFIED LET,LIST,MAP
          VARIABLE OPTIONS USED - SIZE=(256000,51200)                    DEFAULT OPTION(S) USED
IEW0000              NAME  XALGOLW(R)                                          1121.


                                                MODULE MAP


  CONTROL SECTION                          ENTRY

    NAME     ORIGIN  LENGTH                  NAME   LOCATION     NAME   LOCATION     NAME   LOCATION     NAME   LOCATION

  LOADCARD       00     3D8
  ENDLCAD       3D8     258
  AWXGCR01      630     C97
 $BLANKCOM     12C8     558


 ENTRY ADDRESS      630
 TOTAL LENGTH      1820

****XALGOLW   DOES NOT EXIST BUT HAS BEEN ADDED TO DATA SET
```

```
IEF285I   T123.PLLIB                                PASSED
IEF285I   VOL SER NOS= SYS11 .
IEF285I   LOADSET.XALGOL                            DELETED
IEF285I   VOL SER NOS= SYS02 .
IEF285I   SYS2.HISPEED                              KEPT
IEF285I   VOL SER NOS= SYS01 .
IEF285I   SYS1.UT1                                  KEPT
IEF285I   VOL SER NOS= CAMP09.
IEF285I   SYSOUT                                    SYSOUT
IEF285I   VOL SER NOS=       .
IEF285I   SYS1.FORTLIB                              KEPT
IEF285I   VOL SER NOS= SYS00 .
IEF285I   SYS2.SSPLIB                               KEPT
IEF285I   VOL SER NOS= SYS01 .
IEF285I   SYS2.SUBLIB1                              KEPT
IEF285I   VOL SER NOS= SYS01 .
IEF285I   T123.PLLIB                                KEPT
IEF285I   VOL SER NOS= SYS11 .



H A S P  JOB STATISTICS --  1,120 CARDS READ --    1,210 LINES PRINTED --       0 CARDS PUNCHED --   0.58 MINUTES EXECUTION TIME
         167 I/O CALLS              390 SVC CALLS         0.31 MINUTES CPU TM    TIME=  18:04:41 DATE=  12/22/69
```

BIN 914	SEQ	18	ED SATTERTHWAITE	END	18:04	12/22	END	ED SATTERTHWAITE	SEQ	18	BIN 914

BIN 914 *PUNCH* SEQ 951 SATTERTHWAITE

START 17:51 02/07 START SATTERTHWAITE

SEQ 951 *PUNCH* BIN 914

ALGOLW SUPER

H A S P S Y S T E M L O G

NO MESSAGES LOGGED FOR THIS JOB

```
//
//MONASM JOB (T123***,914,1,2,5CC),SATTERTHWAITE,MSGLEVEL=1                    JOB 951
//STEP  EXEC ASMGCL,PARM.ASM='NOLOAD,DECK,ESD,FULLXREF'
//ASM     EXEC  PGM=ASMGASM,PARM='LOAD,NODECK,FULLXREF,ESD'                   00000000
//SYSLIB DD     DSNAME=SYS1.MACLIB,DISP=OLD                                   00000100
//          DD  DSNAME=SYS2.DCLIB,DISP=OLD                                    00000200
//    DD    DDNAME=MACLIB                                                     00000300
//MACLIB  DD  DSNAME=SYS2.DUMMYMAC,DISP=OLD                                   00000400
//SYSUT1    DD  DSNAME=SYS1.UT1,DISP=OLD,DCB=(KEYLEN=0,BLKSIZE=3520)          00000500
//SYSUT2    DD  DSNAME=SYS1.UT2,DISP=OLD,DCB=(KEYLEN=0,BLKSIZE=3520)          00000600
//SYSUT3    DD  DSNAME=SYS1.UT3,DISP=OLD,DCB=(KEYLEN=0,BLKSIZE=3520)          00000700
//SYSPRINT DD   SYSOUT=A                                                      00000800
//SYSLIN    DD DSNAME=&LOADSET,UNIT=2314,SPACE=(CYL,(3,1)),                  *00000900
//             DISP=(MOD,PASS),DCB=(LRECL=80,BLKSIZE=3200,RECFM=FB)           00001000
//SYSPUNCH  DD UNIT=SYSCP                                                     00001100
//ASM.SYSIN  DD  *
IEF236I ALLOC. FOR MONASM   ASM      STEP
IEF237I SYSLIB   ON 235
IEF237I          ON 331
IEF237I          ON 331
IEF237I SYSUT1   ON 1C0
IEF237I SYSUT2   ON 230
IEF237I SYSUT3   ON 330
IEF237I SYSLIN   ON 230
IEF237I SYSPUNCH ON 01D
IEF237I SYSIN    ON 00C
```

OS/360 ASSEMBLER

LEVEL=G RELEASE=12JUL68 SYSTEM=MFT TIME=17:18 DAY=FRIDAY DATE= 7 FEB 69

```
ASSEMBLER OPTICNS=ESD,DECK,LIST,NORLD,NOLOAD,NORENT,NOTEST,EXTIME=5,NOBATCH,OVERLAY,UTBUFF=3,FULLXREF,
                  INSTSET=1,LINECNT=55,NOEXECUTE.
```

7 FEB 69

SYMBOL	TYPE	ID	ADDR	LENGTH	LD ID
ALGOLW02	SD	01	000000	0012BC	

```
LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                                    7 FEB 69

                                   1          ICTL    1,71,18                FORMAT CONTROL
                                   2 ********************************************************************************
                                   3 *                                                                              *
                                   4 *                              ALGOL W SUBMONITOR                              *
                                   5 *                  SPECIAL OS INTERFACE FOR STUDENT ALGOL W JOBS                *
                                   6 *                  STANFORD COMPUTATION CENTER, CAMPUS FACILITY                 *
                                   7 *                              E. SATTERTHWAITE                                *
                                   8 *                              JULY 1968 VERSION                               *
                                   9 *                          LAST MODIFIED 1 DECEMBER 1968                       *
                                  10 *                                                                              *
                                  11 ********************************************************************************

                                  13 *                               ASSEMBLY CONTROL                               *
                                  14          LCLB    &SWAP                    0 - RESIDENT,  1 - OVERLAY

000000                            16 ALGOLW02 CSECT

                                  18 *                               SYMBOLIC REGISTERS                             *
00000A                            19 I        EQU     10
000009                            20 S        EQU     9
000008                            21 T        EQU     8

                                  23 *                               MONITOR PARAMETERS                             *
019000                            24 COREMIN  EQU     100*1024             MINIMUM STORAGE REQUEST
080000                            25 COREMAX  EQU     512*1024             MAXIMUM STORAGE REQUEST
008000                            26 SYSFREE  EQU     2*1024               STORAGE RELEASED TO SYSTEM
000084                            27 LINELEN  EQU     132                  RECORD LENGTH FOR FT06F001
00003C                            28 LINESMAX EQU     60                   MAX. LINES/PAGE FOR FT06F001
000004                            29 BUFMOD   EQU     1*4                  1 PROGRAM BUFFER
00003F                            30 MAXUSEG  EQU     63                   MAXIMUM USER SEGMENT NUMBER
000035                            31 MAXASEG  EQU     53                   MAXIMUM ALGOL SEGMENT NUMBER
007FF8                            32 RECLEN   EQU     32760                MAXIMUM COMPILER RECORD LENGTH
00000A                            33 DEFTIME  EQU     10                   DEFAULT TIME LIMIT (SECONDS)
0001F4                            34 DEFLINES EQU     500                  DEFAULT LINE LIMIT
                                  35 &SWAP    SETB    1                    SET ASSEMBLY OPTION


                                  37          PRINT   NOGEN
00000C                            38          USING   IPL,12
                                  39 IPL      SAVE    (14,12)              COMPLETE OS LINKAGE
000004 18CF                       42          LR      12,15
000006 58E0 C0D8          0000D8  43          L       14,=A(SAVE)
00000A 50E0 D008          000008  44          ST      14,8(,13)
00000E 50D0 E004          000004  45          ST      13,4(,14)
000012 18DE                       46          LR      13,14
000014 5880 C0DC          0000DC  47          L       11,=A(JSDSEG)        ADDRESS DATA
000D4C                            48          USING   JSDSEG,11
000018 D701 B216 B216 00F56 00F56 49          XC      RDRSTATS(2),RDRSTATS CLEAR STATUS FLAGS
                                  50          OPEN    (PRINTER,(OUTPUT),READER,(INPUT),SYSTEM,(INPUT))
                                  60          SPIE    PROGINT,((1,15))     CAPTURE PROGRAM INTERRUPTS
000040 5010 B1E8          00F28   69          ST      1,PIRESET            SAVE PICA ADDRESS
                                  70          GETMAIN VU,LA=ASKLIST,A=GOTLIST  OBTAIN ALL FREE STORAGE
000054 9834 B3C0          01040   78          LM      3,4,GOTLIST
000058 D203 B304 C0E0 01044 000E0 79          MVC     GOTLIST+4(4),=A(SYSFREE)
                                  80          FREEMAIN V,A=GOTLIST         RELEASE STORAGE FOR SYSTEM
00007C 5A30 C0E0          000E0   88          A       3,=A(SYSFREE)
```

```
  LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT
000074 5840 C0E0              000E0    89           S       4,=A(SYSFREE)
000078 9034 B3CC              01040    90           STM     3,4,GOTLIST             SAVE STORAGE LIMITS

00007C 4130 3007              00007    92           LA      3,7(,3)                 ALIGN BOUNDARY
000080 5430 C0E4              000E4    93           N       3,=F'-8'
000084 1A43                            94           AR      4,3
                                       95           AIF     (&SWAP).L1
000086 4130 3004              00004    96 .L1       LA      3,4(,3)
00008A 5030 B1D0              00F10    97           ST      3,BUFADDR+0
00008E 5A30 C0E8              000E8    98           A       3,=F'65524'
000092 9034 B2C4              01004    99 .L2       STM     3,4,COMMLIM             SAVE COMMON BOUNDS
000096 1854                           100           LR      5,4
000098 4100 00C4              00004   101           LA      0,4
00009C 1850                           102           SR      5,0
00009E 5050 B2DC              01010   103           ST      5,ORGP
0000A2 5050 B2D4              01014   104           ST      5,AP
                                      105           TIME    TU                      GET TIME AND DATE
0000AA 5010 B2E8              01028   108           ST      1,WORK
0000AE F343 B2F3 B2E8 01033   01028   109           UNPK    DATE+3(5),WORK(4)
0000B4 D201 B2F2 B2F3 01032   01033   110           MVC     DATE+2(2),DATE+3
0000BA 924B B2F4      01034           111           MVI     DATE+4,C'.'
0000BE 58E0 C0EC              000EC   112           L       14,=A(SUPVPRT)
000000                                113           USING   PRT,14
0000C2 58B0 E128              00128   114           L       11,JSCBASE
0000C6 58C0 E12C              0012C   115           L       12,JSPBASE
000186                                116           USING   JOBSEQ,12
0000CA D71F B3EC B3EC 0112C 0112C     117           XC      PCHKTBL(32),PCHKTBL     ZERO COUNT TABLE
                                      118           AIF     (&SWAP).L3
0000D0 1BAA                           119 .L3       SR      I,I
0000D2 47F0 C020              001A6   120 .L4       B       CYCLE
0000D8                                121           LTORG
0000D8 00000F60                       122                   =A(SAVE)
0000DC CC000D40                       123                   =A(JSDSEG)
0000E0 000008CC                       124                   =A(SYSFREE)
0000E4 FFFFFFF8                       125                   =F'-8'
0000E8 C000FFF4                       126                   =F'65524'
0000EC 0000114C                       127                   =A(SUPVPRT)
                                      128           DROP    11
                                      129           DROP    12
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT
000000                            131          USING   PRT,14
                                  132 *                        GETTIME ROUTINE                    *
0000F0                            134          USING   GETTIME,15
0000F0 90CF FE40          00F30   135 GETTIME  STM     12,15,TSAVE          SAVE WORKING REGISTERS
0000F4 18CF                       136          LR      12,15
0000F0                            137          USING   GETTIME,12
                                  138          DROP    15
0000F6 9610 CE68  00F58           139          CI      TMASK,TR13           INDICATE PRT POINTER STORED
0000FA 41D0 CE70          00F60   140          LA      13,SAVE
0000FE 1BCC                       141          SR      0,0
000100 9120 CE68  00F58           142          TM      TMASK,TIMEOUT        TEST FOR EXPIRED LIMIT
000104 4710 C01C          0010C   143          BO      CONVERT
                                  144          TTIMER
00010C 5810 CE58          00F48   147 CONVERT  L       1,TIMELIM            ASSUME EXECUTION
000110 9180 CE68  00F58           148          TM      TMASK,TIMESET        TEST FOR EXECUTION STAGE
000114 4710 C02C          C011C   149          BO      TDIFF
000118 5810 CE5C          00F4C   150          L       1,COMPLIM            IF NOT, USE COMP INTERVAL
00011C 1F10                       151 TDIFF    SLR     1,0                  COMPUTE INTERVAL USED
00011E 1B00                       152          SR      0,0
000120 5D00 CB68          00C58   153          D       0,=F'640'            CONVERT TO SECONDS/60
000124 98CF CE40          00F30   154          LM      12,15,TSAVE
0000F0                            155          USING   GETTIME,15
                                  156          DROP    12
000128 94EF FE68  00F58           157          NI      TMASK,X'FF'-TR13     INDICATE PRT POINTER LOADED
00012C C7F2                       158          BR      2
                                  159          DROP    15


                                  161 *                        WRITETIME ROUTINE                  *
00012E                            163          USING   WRITTIME,15
00012E 5020 FDFE          0CF2C   164 WRITTIME ST      2,TSAVE2
000132 58F0 E114          00114   165          L       15,GTIMBASE
000136 052F                       166          BALR    2,15                 GET ELAPSED TIME
000138 58C0 E118          00118   167          L       12,WTIMBASE          REGISTERS SAVED BY GETTIME
00012E                            168          USING   WRITTIME,12
                                  169          DROP    15
00013C D283 CF1E CF1D 0104C 0104B 170          MVC     BUFFER(132),BLANK    SKIP A LINE
000142 4100 CF1E          0104C   171          LA      0,BUFFER
000146 45F0 C5F6          CC724   172          BAL     15,WRITE
00014A D215 CF1E CB86 0104C 00CB4 173          MVC     BUFFER(22),=C'ELAPSED TIME IS   :  :'
000150 4120 CF34          01062   174          LA      2,BUFFER+22
000154 41F0 0003          00003   175          LA      15,3
000158 1B00                       176 TLOOP    SR      0,0                  CONVERT UNITS AND UNPACK
00015A 5D00 CB2E          0CC5C   177          D       0,=F'60'
00015E 4E00 CEFA          01028   178          CVD     0,CONWORK
000162 F317 2CC0 CEFA 00000 01028 179          UNPK    0(2,2),CONWORK(8)
000168 96F0 2001          00001   180          OI      1(2),C'0'
00016C 4B20 CB9C          00CCA   181          SH      2,=H'3'
000170 46F0 C02A          C0158   182          BCT     15,TLOOP
000174 4100 CF1E          0104C   183          LA      0,BUFFER             WRITE TIME
C00178 45F0 C5F6          00724   184          BAL     15,WRITE
00017C 5820 CDFE          0CF2C   185          L       2,TSAVE2
000180 98CF CE02          00F30   186          LM      12,15,TSAVE
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT    SOURCE STATEMENT
000184 C7F2                        187           BR      2
                                   188           DROP    12
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT
                                    190 *                          MASTER JOB CONTROL                          *
000186                              191          USING   JOBSEQ,12          PROGRAM BASE REGISTER
000D40                              192          USING   JSDSEG,11          JOB SEQ DATA BASE REGISTER
CC000C                              193          USING   PRT,14             PRT BASE REGISTER

000186                              195 JOBSEQ   CS      0H
000186 58B0 E128            00128   196 PURGE    L       11,JSDBASE
00018A 58C0 E12C            0012C   197          L       12,JSPBASE
                                    198          AIF     (NOT &SWAP).L6
00018E 41D0 B220            00F60   199          LA      13,SAVE
                                    200          CLOSE   (SYSTEM,REREAD),TYPE=T  CLEAR PENDING I/O REQUESTS
00019E 58F0 E118            00118   206 .L6      L       15,WTIMBASE         WRITE TIME
0001A2 C52F                         207          BALR    2,15
0001A4 1BAA                         208          SR      I,I


                                    211 *                          INTERPRETIVE DRIVER                         *
0001A6 4CA0 CB46            00CCC   212 CYCLE    MH      I,=H'12'            COMPUTE POINTER TO STATE RECORD
0001AA 41AA B160            00EA0   213          LA      I,PROCREC(I)
0001AE 92FF B308     01048          214          MVI     SUCCESS,X'FF'       ASSUME SUCCESSFUL EXECUTION
0001B2 1B22                         215 TEST1    SR      2,2                 TEST FOR LOAD PROCESS
0001B4 4320 A000            00000   216          IC      2,LOAD(,I)             FETCH RECORD COUNT
0001B8 8B20 00C2            00002   217          SLA     2,2                    MULTIPLY BY 4
0001BC 4780 CC6A            0C1FC   218          BZ      TEST2
                                    219          AIF     (&SWAP).L7
0001C0 5E80 B1CC            00F0C   220 .L7      L       T,BUFCBPTR          DETERMINE NEXT FREE BUFFER
0001C4 4180 8004            00004   221          LA      T,4(,T)
0001C8 5480 CADA            00C60   222          N       T,=A(BUFMOD-1)
0001CC 5080 B1CC            00F0C   223          ST      T,BUFCBPTR
0001D0 5808 B1D0            00F10   224          L       0,BUFADDR(T)        POINT TO BUFFER
                                    225 .L8      ANOP
0001D4 5812 B114            00E54   226 LDLOOP   L       1,DECBTAB-4(2)      SELECT DECB
0001D8 5C00 10CC            0000C   227          ST      0,12(,1)
0001DC 45F0 C6CA            0C790   228          BAL     15,SYSREAD          INITIATE READ
0001E0 4B20 CB48            00CCE   229          SH      2,=H'4'
0001E4 4780 CC6A            001F0   230          BZ      TEST2
0001E8 5A00 CADE            00C64   231          A       0,=A(RECLEN)        ADJUST DESTINATION ADDRESS
0001EC 47F0 C04E            001D4   232          B       LDLOOP
0001FC 9500 ACC1     00C01          233 TEST2    CLI     EXEC(I),0           TEST FOR EXECUTION PROCESS
0001F4 4780 CC88            0020E   234          BE      TEST3
0001F8 4880 A004            00004   235          LH      T,EXECID(,I)        DETERMINE PROGRAM ID
0001FC 58E8 B148            00E88   236          L       14,PRTBASE(T)       SELECT CORRECT PRT
000200 4880 A006            00006   237          LH      T,INDEX(,I)         SELECT SUBROUTINE FOR ACTIVATION
C00204 5818 CA5E            00C24   238          L       1,SUBRSW(T)
000208 0521                         239          BALR    2,1                 LINK
00020A 58E0 CAE2            00C68   240          L       14,=A(SUPVPRT)      REESTABLISH MONITOR PRT
C0020E 1B22                         241 TEST3    SR      2,2                 TEST FOR LOAD PROCESS
000210 4320 A000            00000   242          IC      2,LOAD(,I)
000214 8B20 00C2            000C2   243          SLA     2,2
000218 4780 CCEE            00274   244          BZ      SELECT
00021C 5812 B114            00E54   245 CHKLOOP  L       1,DECBTAB-4(2)      SELECT DECB
000220 45F0 C616            0079C   246          BAL     15,CHECK            ISSUE CHECK
000224 4B20 CB48            00CCE   247          SH      2,=H'4'
000228 4770 CC96            0021C   248          BNZ     CHKLOOP
```

```
 LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT
                                   250 *                        ADJUST PRT ENTRIES                    *
                                   251          AIF     (&SWAP).L9
00022C 5880 B1CC             00F0C 252 .L9      L       T,BUFCBPTR
000230 5818 B1D0             00F10 253          L       1,BUFADDR(T)          (R1) = BUFFER ADDRESS
000234 5820 1000             00000 254 .L10     L       2,0(,1)               (R2) = PRT DISPLACEMENT
000238 4121 2CC0             00000 255          LA      2,0(1,2)              (R2) = PRT ADDRESS
00023C 4130 0000             00CC0 256          LA      3,0
000240 4930 CB4A             00CD0 257 MAPLOOP  CH      3,=AL2(MAXUSEG*4+8)
000244 4720 C0DC             00262 258          BH      ENDMAP
000248 5843 2000             00000 259          L       4,0(3,2)              (R4) = SEGMENT DISPLACEMENT
00024C 1244                        260          LTR     4,4                   0 IFF NULL SEGMENT
00024E 4780 CCD4             0025A 261          BZ      MAPINCR
000252 4144 1000             00000 262          LA      4,0(4,1)              (R4) = SEGMENT ADDRESS
000256 5043 2000             00000 263          ST      4,0(3,2)
00025A 4130 3004             00004 264 MAPINCR  LA      3,4(,3)
00025E 47F0 C0EA             00240 265          B       MAPLOOP
000262 D237 210C B518 0010C  01258 266 ENDMAP   MVC     268(56,2),SUPVPRT+268 COMPLETE PRT WITH SUPV ROUTINES
000268 4880 A002             00002 267          LH      T,LOADID(,I)          DETERMINE ID OF LOADED PROGRAM
00026C 5800 2000             00000 268          L       0,0(,2)               MAKE ENTRY IN PRT DIRECTORY
000270 5008 B148             00E88 269          ST      0,PRTBASE(T)
                                   270          AIF     (&SWAP).L11
                                   271 .L11     ANOP

000274 95FF B308       01048       273 SELECT  CLI     SUCCESS,X'FF'         TEST FOR SUCCESSFUL EXECUTION
000278 4770 C0FE             00284 274          BNE     PRGFAIL
00027C 48AA CCC8             CCCC8 275          LH      I,NORMNEXT(I)         SELECT NORMAL NEXT STATE
000280 47F0 C020             001A6 276          B       CYCLE
000284 48AA CCCA             0000A 277 PRGFAIL  LH      I,ERRNEXT(I)          SELECT EXCEPTIONAL NEXT STATE
000288 47F0 C020             001A6 278          B       CYCLE
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT
                                    280 *                        JOB INITIALIZATION                          *
00028C 5020 B1CC                 00F1C  281 INITIAL  ST      2,R2SAVE
00029C D7C7 B218 E218 00F58 00F58  282          XC      STATS(8),STATS
000296 92FE C4E5      0066B      283          MVI     INTEXIT+1,X'FE'       RESTORE TRAPS
00029A 92FF C93F      00AC5      284          MVI     IOEXIT+1,X'FF'
00029E 4560 C3A2                 00528  285          BAL     6,SHOWCHK             WRITE CHECK COUNTS
                                    286          AIF     (NOT &SWAP).L14
0002A2 1B00                      287          SR      0,0
0002A4 5000 B1CC                 C0FCC  288          ST      0,BUFCBPTR
                                    289 .L14     ANOP
0002A8 D283 B3CC B30B 0104C 0104B  290 IO       MVC     BUFFER(132),BLANK
0002AE 92F1 B3E2      01122      291          MVI     CARRCONT,C'1'         SKIP TO NEW PAGE
0002B2 C207 B384 B2F0 010C4 01030  292          MVC     BUFFER+120(8),DATE
                                    293          AIF     (&SWAP).L15
0002B8 D221 B360 CB4C 010A0 00CD2  294 .L15     MVC     BUFFER+84(34),=C'STANFORD ALGOL W (OVERLAY VERSION)'
0002BE 410C B3CC                 0104C  295 .L16     LA      0,BUFFER
0002C2 95FF B3E0      01120      296          CLI     EOF,X'FF'             TEST FOR PRIOR JOB CARD
0002C6 4770 C14E                 002D4  297          BNE     I1
0002CA D24F B3CC B390 0104C 010D0  298          MVC     BUFFER(80),CARDBUF    MOVE FROM SAVE AREA
0002D0 47F0 C15A                 002E0  299          B       I2
0002D4 45F0 C592                 00718  300 I1       BAL     15,READ               READ TO JOB CARD
0002D8 95FF B3E0      01120      301          CLI     EOF,X'FF'
0002DC 4770 C14E                 002D4  302          BNE     I1
0002EC 9200 B3E0      01120      303 I2       MVI     EOF,0                 RESET EOF
0002E4 45F0 C59E                 00724  304          BAL     15,WRITE              WRITE HEADER
0002E8 92F0 B3E2      01122      305          MVI     CARRCONT,C'0'         SKIP LINE
0002EC 9202 B3E3      01123      306          MVI     LINECNT,2             ADJUST LINE COUNT
0002F0 D507 B300 CAB2 0104D 00C38  307          CLC     BUFFER+1(8),=CL8'ALGOL' TEST FOR ALGOL
0002F6 4780 C190                 00316  308          BE      I3
0002FA 925C B3CC      0104C      309          MVI     BUFFER,C'*'           INDICATE UNAVAILABILITY
0002FE D20E B315 CB88 01055 00D0E  310          MVC     BUFFER+9(15),=C' NOT AVAILABLE '
000304 D26A B324 B323 01064 01063  311          MVC     BUFFER+24(107),BUFFER+23
00030A 4100 B30C                 0104C  312          LA      0,BUFFER
00030E 45F0 C59E                 00724  313          BAL     15,WRITE
000312 47F0 C122                 002A8  314          B       IO                    READ TO NEXT JOB CARD

000316 41D0 B220                 00F60  316 I3       LA      13,SAVE               ESTABLISH OS CONVENTIONS
                                    317          AIF     (NOT &SWAP).L18
00031A 189E                      318          LR      S,14
00031C 4100 CAE6                 00C6C  319          LA      0,=F'1'               REPOSITION COMPILER FILE
                                    320          POINT   SYSTEM,(0)
00032C 18E9                      324          LR      14,S
00032E 1B00                      325 .L18     SR      0,0                   ANALYZE LIMIT FIELDS
000330 1810                      326          LR      1,0
000332 1820                      327          LR      2,0
000334 1880                      328          LR      T,0
000336 D701 B309 B309 01049 01049  329          XC      CHARSW(2),CHARSW      RESET STATE SWITCHES
00033C 4130 B314                 01054  330          LA      3,BUFFER+8
000340 4130 3001                 00001  331 L1       LA      3,1(,3)               POINT TO NEXT CHARACTER
000344 5930 CAEA                 00C70  332          C       3,=A(BUFFER+28)       TEST FOR END OF FIELD
000348 4720 C24C                 003D2  333          BH      L7
00034C 9540 3000      00000      334          CLI     0(3),C' '             IGNORE BLANKS
000350 4780 C1BA                 00340  335          BE      L1
000354 95F0 3000      00000      336          CLI     0(3),C'0'             TEST FOR DIGIT
000358 4740 C20C                 00392  337          BL      L4
```

```
  LCC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT
00035C 94CF 3CCC       00000         338          NI      0(3),X'0F'          CLEAR ZONE
000360 4380 3000              00CC0  339          IC      T,0(,3)
000364 95FF B30A       0104A         340          CLI     COMMA,X'FF'
000368 4770 C1F0              00376  341          BNE     L2
00036C 4C20 CB6E              00CF4  342          MH      2,=H'10'            ACCUMULATE LINES
000370 1A28                          343          AR      2,T
000372 47F0 C1BA              0034C  344          B       L1
000376 95FF B3C9       01049         345 L2       CLI     COLON,X'FF'
00037A 4770 C202              00388  346          BNE     L3
00037E 4C10 CB6E              00CF4  347          MH      1,=H'10'            ACCUMULATE SECONDS
000382 1A18                          348          AR      1,T
000384 47F0 C1BA              0034C  349          B       L1
000388 4C00 CB6E              00CF4  350 L3       MH      0,=H'10'            ACCUMULATE MINUTES
00038C 1AC8                          351          AR      0,T
00038E 47F0 C1BA              0034C  352          B       L1
000392 957A 3000       00000         353 L4       CLI     0(3),C':'           CHANGE STATE TO SCAN SECONDS
000396 4770 C220              003A6  354          BNE     L5
00039A 9300 B3C9       01C49         355          TS      COLON
00039E 4780 C1BA              00340  356          BZ      L1
0003A2 47F0 C230              003B6  357          B       L6
0003A6 956B 3000       00000         358 L5       CLI     0(3),C','           CHANGE STATE TO SCAN LINES
0003AA 4770 C23C              003B6  359          BNE     L6
0003AE 9300 B3CA       01C4A         360          TS      COMMA
0003B2 4780 C1BA              00340  361          BZ      L1
0003B6 410C B3CC              0104C  362 L6       LA      0,BUFFER            LIMIT FIELD FORMAT INVALID
0003BA D283 B30C B30B 0104C 0104B    363          MVC     BUFFER(132),BLANK
0003C0 D21C 30CC CB97 00000 00D1D    364          MVC     0(29,3),=C'| INVALID LIMIT SPECIFICATION'
0003C6 925C B30C       0104C         365          MVI     BUFFER,C'*'
0003CA 45F0 C59E              00724  366          BAL     15,WRITE
0003CE 47F0 C122              002A8  367          B       IO
0003D2 189E                          368 L7       LR      S,14
0003D4 1222                          369          LTR     2,2                 TEST FOR LINE LIMIT
0003D6 4770 C258              0C3DE  370          BNZ     L8
0003DA 5820 CAEE              00C74  371          L       2,=A(DEFLINES)      IF NONE, USE DEFAULT LIMIT
0003DE 5C20 B3E4              01124  372 L8       ST      2,LINELIM
0003E2 4C00 CB7C              00CF6  373          MH      0,=H'60'            CONVERT TO SECONDS
0003E6 1A10                          374          AR      1,0                 COMBINE
0003E8 4770 C26A              003F0  375          BNZ     L9                  TEST FOR TIME LIMIT
0003EC 5810 CAF2              00C78  376          L       1,=A(DEFTIME)       IF NONE, USE DEFAULT LIMIT
0003F0 4C10 CB7C              00CF6  377 L9       MH      1,=H'60'            CONVERT TO SIXTIETHS
0003F4 5C00 CAD2              00C58  378          M       0,=F'640'           CONVERT TO TIMER UNITS
0003F8 5010 B208              00F48  379          ST      1,TIMELIM
                                     380          STIMER  TASK,TUINTVL=COMPLIM START CLOCK
000404 18E9                          384          LR      14,S
000406 5820 B1DC              00F1C  385          L       2,R2SAVE
00040A C7F2                          386          BR      2                   RETURN
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                    7 FEB 69

                                    388 *                        COMPILER PASS EXECUTION          *
0007F9                              390 MONFLAG  EQU     X'7F9'              MONITOR IDENTIFICATION FLAG

00040C 5020 B1CC          00F1C     392 PASSEXEC ST      2,R2SAVE
000410 50A0 B1D4          00F14     393          ST      I,ISAVE             SAVE STATE INDEX
                                    394          AIF     (&SWAP).L19
000414 9200 B3E9    01129         395 .L19     MVI     PMASK,X'00'         DISABLE PROGRAM CHECKS
000418 5810 CAF6          00C7C     396          L       1,=XL4'0E000000'    SUPPRESS SIGNIFICANCE CHECK
00041C C410                       397          SPM     1
00041E 9801 B2C4          01004     398          LM      0,1,COMMLIM         SUPPLY COMMON LIMITS
000422 C610                       399          BCTR    1,0
000424 1820                       400          LR      2,0
000426 92FF 27F9    007F9         401          MVI     MONFLAG(2),X'FF'    SET MONITOR FLAG
00042A 5860 CADE          00C64     402          L       6,=A(RECLEN)           AND RECORD LENGTH
00042E 58F0 E004          00004     403          L       15,4(,14)
000432 C52F                       404          BALR    2,15
000434 58C0 E12C          0012C     405          L       12,JSPBASE
000438 58B0 E128          00128     406          L       11,JSDBASE
00043C 5010 B1EC          00F20     407          ST      1,HICORE            SAVE HIGH COMMON LOCATION
000440 1200                       408          LTR     0,0                 TEST RETURN CODE
000442 4780 C2CA          00450     409          BZ      P1
000446 9200 B308    01048         410          MVI     SUCCESS,0           INDICATE FAILURE
00044A 58F0 E118          00118     411          L       15,WTIMBASE         WRITE COMPILATION TIME
00044E C52F                       412          BALR    2,15
000450 58A0 B1C4          00F14     413 P1       L       I,ISAVE
000454 5820 B1CC          00F1C     414          L       2,R2SAVE
000458 C7F2                       415          BR      2
```

```
 LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT

                                  417 *                          COMPILED PROGRAM EXECUTION                *

000170                            419 MP       EQU     X'170'
000174                            420 LIM      EQU     X'174'
0001AC                            421 RECTAB   EQU     X'1AC'
0002AF                            422 EOFFLAG  EQU     X'2AF'

00045A 5020 B1DC          00F1C   424 RUNEXEC  ST      2,R2SAVE
00045E 50A0 B1D4          00F14   425          ST      I,ISAVE
                                  426          AIF     (&SWAP).L21
000462 953C B3E3    01123         427 .L21     CLI     LINECNT,LINESMAX      TEST FOR FULL PAGE
000466 47B0 C2E8          0046E   428          BNL     R3
00046A 9240 B3E2    01122         429          MVI     CARRCONT,C' '         CANCEL PAGE SKIP
00046E 58F0 E118          00118   430 R3       L       15,WTIMBASE           WRITE COMPILATION TIME
000472 052F                       431          BALR    2,15
000474 4560 C3A2          00528   432          BAL     6,SHOWCHK             WRITE CHECK COUNTS

000478 4100 E1AC          001AC   434          LA      0,RECTAB(,14)         READ RECORD TABLE
00047C 4110 0100          00100   435          LA      1,256
000480 4120 CCC5          00CC5   436          LA      2,5
000484 45F0 C5DA          00760   437          BAL     15,RDTAPE
000488 D2D3 ECC4 8410 00004 01150 438          MVC     4(MAXASEG*4,14),SUPVPRT+4 MERGE PRT SEGMENT ADDRESSES

00048E 4100 B3CC          01C4C   440 R4       LA      0,BUFFER              READ TO NEXT CONTROL CARD
000492 45F0 C592          00718   441          BAL     15,READ
000496 4780 C3C8          0048E   442          BE      R4
00049A D200 E2AF B3E0 002AF 0112C 443          MVC     EOFFLAG(1,14),EOF     PASS EOF INDICATOR
0004AC 5810 CAFA          00C80   444          L       1,=XL4'08000000'      SUPPRESS SIGNIFICANCE AND UNFL
0004A4 0410                       445          SPM     1
0004A6 92C9 B3E9    01129         446          MVI     PMASK,X'C9'
0004AA 92FF B21D    00F5D         447          MVI     ARUN,X'FF'
0004AE 92F1 B3E2    01122         448          MVI     CARRCONT,C'1'         SKIP TO NEW PAGE
0004B2 5830 E100          00100   449          L       3,PROGLEN             FETCH STORAGE BOUNDS
0004B6 45F0 C622          007A8   450          BAL     15,FSPTM              POSITION POINTERS
0004BA 45F0 C622          007A8   451          BAL     15,FSPTM
0004BE 189E                       452          LR      S,14
0004C0 4100 B220          C0F60   453          LA      13,SAVE               SET TIME LIMIT
                                  454          STIMER  TASK,TIMEINT,TUINTVL=TIMELIM
0004CE 92C0 B218    00F58         458          MVI     TMASK,TIMESET+TIMEON  ENABLE TIMER TRAP
0004D2 92FF B21C    00F5C         459          MVI     LINECHK,X'FF'          AND LINE COUNTING
0004D6 18E9                       460          LR      14,S
0004D8 5840 B2D4          01014   461          L       4,AP
0004DC 5440 CAFE          00C84   462          N       4,=F'-8'
0004E0 5030 E17C          00170   463          ST      3,MP(,14)             SET UP ALGOL STORAGE RECORDS
0004E4 5040 E174          00174   464          ST      4,LIM(,14)
0004E8 4100 3018          00018   465          LA      0,24(,3)                 AND FIRST BLOCK MARK
0004EC 1B11                       466          SR      1,1
0004EE 1B22                       467          SR      2,2
0004FC 9C02 3000          00000   468          STM     0,2,0(3)
0004F4 58F0 E0C4          00CC4   469          L       15,4(,14)             LINK TO PROGRAM
0004F8 051F                       470          BALR    1,15
0004FA 58C0 E12C          0012C   471          L       12,JSPBASE            RESTORE ADDRESSING
0004FE 58B0 E128          00128   472          L       11,JSDBASE
000502 94BF B218    00F58         473          NI      TMASK,X'FF'-TIMEON    DISABLE TIMER
000506 58F0 E118          00118   474          L       15,WTIMBASE           WRITE TOTAL TIME
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT
00050A 052F                       475          BALR    2,15
00050C 9120 B218    00F58         476          TM      TMASK,TIMEOUT     TEST FOR RUNNING CLOCK
000510 4710 C398          0051E   477          BO      R5                SKIP IF EXPIRED
000514 41D0 B220          00F60   478          LA      13,SAVE
                                  479          TTIMER  CANCEL            CANCEL STIMER
00051E 58A0 B1C4          00F14   482 R5       L       I,ISAVE
000522 5820 B1CC          00F1C   483          L       2,R2SAVE
000526 07F2                       484          BR      2
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT
000528 D50F B3FC B3EC 0113C 0112C   486 SHOWCHK  CLC     PCHKTBL+16(16),PCHKTBL
00052E C786                         487          BCR     8,6
000530 D283 B3CC B30B 0104C 0104B   488          MVC     BUFFER(132),BLANK      WRITE PROGRAM CHECK COUNTS
000536 D20F B311 CABA 01051 00C40   489          MVC     BUFFER+5(16),=C'PROGRAM CHECK NO'
00053C 4100 B30C              0104C 490          LA      0,BUFFER
000540 1B11                         491          SR      1,1
000542 4120 000E              0000E 492          LA      2,14
000546 4120 2CC2              00002 493 SC1      LA      2,2(,2)
00054A 4920 CB72              00CF8 494          CH      2,=H'30'
00054E 4720 C402              00588 495          BH      SC2
000552 4912 B3EC              0112C 496          CH      1,PCHKTBL(2)
000556 4780 C3C0              00546 497          BE      SC1
00055A 4832 B3EC              0112C 498          LH      3,PCHKTBL(2)
00055E 4E30 B2E8              01028 499          CVD     3,CONWORK
000562 F337 B3CC B2E8 0104C 01028   500          UNPK    BUFFER(4),CONWORK(8)
000568 96F0 B30F       0104F        501          OI      BUFFER+3,C'0'
00056C 1832                         502          LR      3,2
00056E 8830 0001              00001 503          SRL     3,1
000572 4E30 B2E8              01028 504          CVD     3,CONWORK
000576 F317 B322 B2E8 01062 01028   505          UNPK    BUFFER+22(2),CONWORK(8)
00057C 96F0 B323       01063        506          OI      BUFFER+23,C'0'
000580 45F0 C59E              00724 507          BAL     15,WRITE
000584 47F0 C3C0              00546 508          B       SC1
000588 D71F B3EC B3EC 0112C 0112C   509 SC2      XC      PCHKTBL(32),PCHKTBL
00058E 07F6                         510          BR      6
                                    511          AIF     (&SWAP).L23
                                    512 .L23     ANOP

                                    514          DROP    11
                                    515          DROP    12
                                    516          DROP    14
```

```
 LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT

                                  518 *                    TIMER INTERRUPT EXIT                    *
000590                            520          USING   TIMEINT,15
000594 9620 F9C8       00F58      521 TIMEINT  OI      TMASK,TIMEOUT        SET TO TRAP
000598 9140 F9C8       00F58      522          TM      TMASK,TIMEON         TEST FOR SOFTWARE ENABLE
00059C 078E                       523          BCR     8,14                 IF NOT, IGNORE
00059E 90E1 DOOC             0000C 524          STM     14,1,12(13)
0005A2 95FF F9CA       00F5A      525          CLI     INTFLAG,X'FF'        TEST FOR INTERRUPT EXIT
0005A6 4780 F05C             005EC 526          BE      SETINT
0005AA 95FF F9C8       00F5B      527          CLI     IOFLAG,X'FF'         TEST FOR I/O EXIT
0005AE 4780 FC6C             005FC 528          BE      SETIO
                                  529 *                 LOCATE PSW TO BE RELOADED                  *
0005B2 5810 0010             00010 530          L       1,CVTPTR             POINT TO CVT
0005B6 5810 1000             00000 531          L       1,CVTTCBP(,1)        POINT TO TCB POINTER BLOCK
0005BA 5010 F994             00F24 532          L       1,TCBCURR(,1)        POINT AT CURRENT TCB
0005BE 5810 1000             00000 533          ST      1,TCBADR             SAVE ADDRESS
0005C2 D502 101D F995 0001D  00F25 534          L       1,TCBRBP(,1)         ENTER RB POINTER CHAIN
0005C8 4780 F044             005D4 535 CHAIN    CLC     XRBLINK+1(3,1),TCBADR+1 CHAIN TO LAST RB
0005CC 5810 101C             0001C 536          BE      PRBEND
0005D0 47F0 F032             005C2 537          L       1,XRBLINK(,1)
0005D4 5810 1014             00014 538          B       CHAIN
0005D8 4110 1000             000CC 539 PRBEND   L       1,XRBPSW+4(,1)       GET ADDRESS OF RETURN POINT
0005DC 5510 F6F8             00C88 540          LA      1,0(,1)
0005E0 4740 F074             00604 541          CL      1,=A(PROGINT)        TEST FOR I/O OR INT LINK
0005E4 5510 F6FC             00C8C 542          BL      SETBYTE              BELOW INT LINK
0005E8 4780 F064             005F4 543          CL      1,=A(SERVICE)
0005EC 9200 F0DB       0066B      544          BNL     TTEST1               ABOVE INT LINK
0005F0 47F0 F082             00612 545 SETINT   MVI     INTEXIT+1,X'00'      CONVERT INT EXIT TO NO-OP
0005F4 5510 F700             00C90 546          B       TEXIT
0005F8 4720 F074             00604 547 TTEST1   CL      1,=A(LINKEND)
0005FC 9200 F535       00AC5      548          BH      SETBYTE              ABOVE I/O LINK
000600 47F0 FC82             00612 549 SETIO    MVI     IOEXIT+1,X'00'       CONVERT I/O EXIT TO NO-OP
000604 5010 F9CC             00F50 550          B       TEXIT
000608 D200 F9C4 1000 00F54  00000 551 SETBYTE  ST      1,TINTLOC            SAVE INT ADDRESS
00060E 9200 1000       00000      552          MVC     TINTBYTE(1),0(1)        AND INSTRUCTION BYTE
000612 98E1 D00C             0000C 553          MVI     0(1),0               ZERO OPCODE
000616 07FE                       554 TEXIT    LM      14,1,12(13)          RETURN
                                  555          BR      14
                                  556          DROP    15

000010                            558 CVTPTR   EQU     16                   CVT ADDRESS
000000                            559 CVTTCBP  EQU     0
000004                            560 TCBCURR  EQU     4
000000                            561 TCBRBP   EQU     0
00001C                            562 XRBLINK  EQU     28
000010                            563 XRBPSW   EQU     16
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                                    7 FEB 69

                                    565 *                    PROGRAM INTERRUPT RECOVERY                      *
000618                              567          USING   PROGINT,15
000618 92FF F942       00F5A        568 PROGINT  MVI     INTFLAG,X'FF'        INDICATE INT PROCESSING
00061C 90E2 F9D8             00FF0  569          STM     14,2,PISAVE          SAVE OS LINK
000620 9120 F940       00F58        570          TM      TMASK,TIMEOUT        TEST FOR TIMER FORCED INTERRUPT
000624 4780 F022             0063A  571          BZ      NOTIMER
000628 58E0 F938             00F50  572          L       14,TINTLOC           RESTORE INSTRUCTION BYTE
00062C D200 E000 F93C 00000  00F54  573          MVC     0(1,14),TINTBYTE
000632 4120 0006             00006  574          LA      2,6                  SET TIMER INTERRUPT CODE
000636 47F0 FCA4             006BC  575          B       PROGINT3
00063A 48E0 1006             00006  576 NOTIMER   LH      14,PIEPSW+2(,1)      SELECT PMASK BIT FROM CODE
00063E 54E0 F67C             00C94  577          N       14,=F'15'            CLEAR 360/91 BIT MASK
000642 5800 FB10             01128  578          L       0,PMASK-1
000646 8900 E000             00000  579          SLL     0,0(14)
00064A 1AEE                         580          AR      14,14                INCREMENT COUNT TABLE
00064C 482E FB14             0112C  581          LH      2,PCHKTBL(14)
000650 4120 2001             00001  582          LA      2,1(,2)
000654 402E FB14             0112C  583          STH     2,PCHKTBL(14)
000658 1200                         584          LTR     0,0                  TEST BIT
00065A 4740 FC6C             00678  585          BL      PROGINT2             PROCESS IF ONE
00065E 9200 F942       00F5A        586 PROGINT1 MVI     INTFLAG,0            RESET FLAG
000662 98E2 F9D8             00FF0  587          LM      14,2,PISAVE
000666 9630 1008       00008        588          OI      PIEPSW+4(1),X'30'    SET CC = 3
00066A 07FE                         589 INTEXIT  BR      14                   CHANGED TO NO-OP BY TIMER TRAP
00066C 92FE FC53       0066B        590          MVI     INTEXIT+1,X'FE'      RESTORE BRANCH
000670 4120 0006             00006  591          LA      2,6
000674 47F0 F0A4             006BC  592          B       PROGINT3

000678 4120 0004             00004  594 PROGINT2 LA      2,4                  SET PROGRAM CHECK CODE
00067C 95FF F945       00F5D        595          CLI     ARUN,X'FF'           TEST FOR ALGOLRUN
00068C 4780 FCA4             006BC  596          BE      PROGINT3
000684 58E0 F650             00C68  597 INTMSG    L       14,=A(SUPVPRT)
000688 9200 F944       00F5C        598          MVI     LINECHK,0            DISABLE LINE COUNT
00068C 4100 FA34             01C4C  599          LA      0,BUFFER
000690 18CF                         600          LR      12,15
000618                              601          USING   PROGINT,12
                                    602          DROP    15
000692 925C CA34       0104C        603          MVI     BUFFER,C'*'
000696 D282 CA35 CA34 0104D 0104C   604          MVC     BUFFER+1(131),BUFFER
00069C 45F0 C10C             00724  605          BAL     15,WRITE
0006AC D20D CA34 C6E2 0104C 00CFA   606          MVC     BUFFER(14),=C'COMPILER ERROR'
0006A6 D213 CA42 C68C 0105A 00C98   607          MVC     BUFFER+14(20),=C' DURING COMPILATION '
0006AC D261 CA56 CA55 0106E 0106D   608          MVC     BUFFER+34(98),BUFFER+33
0006B2 45F0 C1CC             00724  609          BAL     15,WRITE
0006B6 18FC                         610          LR      15,12
000618                              611          USING   PROGINT,15
                                    612          DROP    12
0006B8 47F0 F0EE             007C6  613          B       PROGINT5             TERMINATE
0006BC 58E0 100C             0000C  614 PROGINT3 L       14,PIERE(,1)
0006C0 D20B F994 1014 00FAC 00014   615          MVC     REGSAVE(12),PIERO(1) ASSEMBLE REGISTERS
0006C6 50D0 F9D4             00FEC  616          ST      13,SSAVE13           SAVE R13
0006CA 9110 F940       00F58        617          TM      TMASK,TR13           LOCATE PRT POINTER
0006CE 4780 FCBE             006D6  618          BZ      PROGINT4             IF OFF, IN R13
0006D2 58D0 F91C             00F34  619          L       13,TSAVE+4           OTHERWISE IN SAVE AREA
```

```
  LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT
0006D6 90C3D F9A0              00FB8   620 PROGINT4 STM     3,13,REGSAVE+12
0006DA D207 F9CC 100C 00FE4 00CCC   621          MVC     REGSAVE+56(8),PIERE(1)
0006E0 1B00                         622          SR      0,0                     SET UP CALL
0006E2 4160 F994              00FAC 623          LA      6,REGSAVE
0006E6 1872                         624          LR      7,2
0006E8 4180 1CC4              00004 625          LA      8,PIEPSW(,1)
0006EC 0700                         626          CNOP    2,4
0006EE 58F0 D0FC              000FC 627          L       15,63*4(,13)            CALL ALGOL RUNERROR
0006F2 052F                         628          BALR    2,15
0006F4 58F0 2900              00900 629          L       15,PISAVE+4-*(,2)       RESTORE REGISTERS
0006F8 983C F9A0              00FB8 630          LM      3,12,REGSAVE+12
0006FC 58D0 F9D4              00FEC 631          L       13,SSAVE13
000700 1200                         632          LTR     0,0                     TEST FOR CONTINUATION
000702 4780 FC46              0065E 633          BZ      PROGINT1
000706 98E2 F9D8              00FF0 634 PROGINT5 LM      14,2,PISAVE             TERMINATE JOB AND PURGE
00070A D203 1CCC F650 0000C 00C68   635          MVC     PIERE(4,1),=A(SUPVPRT)
000710 D202 1CC9 F722 00CC9 00D3A   636          MVC     PIEPSW+5(3,1),=AL3(PURGE)
000716 07FE                         637          BR      14

000004                              639 PIEPSW   EQU     4
00000C                              640 PIERE    EQU     12
000014                              641 PIERO    EQU     20
```

```
  LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT                                                    7 FEB 69
                                     643 *        ALGOL W SYSTEM SUPERVISOR STANDARD FUNCTIONS                 *
                                     644 *        MINIMAL SET FOR ALGOL W COMPILATION AND EXCUTION             *
                                     645 *           (1) DEVICE 5 ASSUMED IN MAIN STORAGE                      *
                                     646 *           (2) DEVICE 7 NOT SUPPORTED                                *

000718                               648          USING   SERVICE,12
000718                               649 SERVICE  DS      OF
000718 90C3 E144           00144     650 READ     STM     12,3,RCSAVE(14)       SAVE REGISTERS
00071C 58C0 E140           00140     651          L       12,DSERV(,14)         ESTABLISH ADDRESSING
000720 47F0 C0B4           007CC     652          B       READ1                 BRANCH TO SERVICE ROUTINE
000724 90C3 E144           00144     653 WRITE    STM     12,3,RCSAVE(14)
000728 58C0 E140           00140     654          L       12,DSERV(,14)
00072C 47F0 C0FE           00816     655          B       WRITE1
000730 90C3 E144           00144     656 PAGE     STM     12,3,RCSAVE(14)
000734 58C0 E140           00140     657          L       12,DSERV(,14)
000738 47F0 C16A           00882     658          B       PAGE1
00073C C000000000000000              659          DC      9F'0'                 PUNCH & TYPEWRITER NOT SUPPORTED
00076C 90C3 E144           00144     660 RDTAPE   STM     12,3,RCSAVE(14)
000764 58C0 E140           00140     661          L       12,DSERV(,14)
000768 47F0 C1AA           008C2     662          B       RDTAPE1
00076C 90C3 E144           00144     663 WRTAPE   STM     12,3,RCSAVE(14)
00077C 58C0 E140           0014C     664          L       12,DSERV(,14)
000774 47F0 C200           00918     665          B       WRTAPE1
000778 90C3 E144           00144     666 MKTAPE   STM     12,3,RCSAVE(14)
00077C 58C0 E140           00140     667          L       12,DSERV(,14)
000780 47F0 C308           00A20     668          B       MKTAPE1
000784 90C3 E144           00144     669 REWIND   STM     12,3,RCSAVE(14)
000788 58C0 E140           00140     670          L       12,DSERV(,14)
00078C 47F0 C342           00A5A     671          B       REWIND1
000790 90C3 E144           00144     672 SYSREAD  STM     12,3,RCSAVE(14)
000794 58C0 E140           00140     673          L       12,DSERV(,14)
000798 47FC C182           00C8A     674          B       SYSREAD1
00079C 90C3 E144           00144     675 CHECK    STM     12,3,RCSAVE(14)
0007AC 58C0 E140           00140     676          L       12,DSERV(,14)
0007A4 47F0 C198           00CB0     677          B       CHECK1
0007A8 90C3 E144           00144     678 FSPTM    STM     12,3,RCSAVE(14)
0007AC 58C0 E14C           00140     679          L       12,DSERV(,14)
0007B0 47F0 C362           00A7A     680          B       FSPTM1
0007B4 CC0000CC00000000              681          DC      3F'0'                 BSPTM NOT SUPPORTED
0007C0 90C3 E144           00144     682 SPMASK   STM     12,3,RCSAVE(14)
0007C4 58C0 E140           00140     683          L       12,DSERV(,14)
0007C8 47F0 C176           0088E     684          B       SPMASK1
```

```
 LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT

                                  686 *                    I/O SERVICE ROUTINES                    *

0007CC 4530 C390              00AA8   688 READ1    BAL     3,SETUP              SET UP REGISTERS
0007DC 9200 CAC9      01121           689          MVI     CCFLAG,0
                                      690 READ2    GET     READER,(0)           GET NEXT CARD IMAGE
0007DE 95FF C83F      00F57           694          CLI     REREAD,X'FF'         TEST FOR DATA SET SWITCH
0007E2 4770 CCD6              007EE   695          BNE     READ3
0007E6 9200 C83F      00F57           696          MVI     REREAD,0             RESET SWITCH
0007EA 47F0 CCBC              007D4   697          B       READ2                REISSUE GET
0007EE 956C 1000      00000           698 READ3    CLI     0(1),C'%'            TEST FOR CONTROL CARD
0007F2 4770 C0F6              0080E   699          BNE     NOTEOF
0007F6 92FF CAC9      01121           700          MVI     CCFLAG,X'FF'         PREPARE TO SET CC = 1
0007FA D507 1001 C538 00001 00C50     701          CLC     1(8,1),=CL8'EOF'     TEST FOR 'EOF' CARD
000800 4780 C0F6              0080E   702          BE      NOTEOF
000804 92FF CA08      01120           703          MVI     EOF,X'FF'            INDICATE SYSTEM EOF
000808 D24F C988 1000 010D0 00000     704          MVC     CARDBUF(80),0(1)     SAVE CARD IMAGE
00080E 9500 CAC9      01121           705 NOTEOF   CLI     CCFLAG,0             SET CONDITION CODE
000812 47F0 C3A0              00AB8   706          B       RETURN1

000816 4530 C39C              00AA8   708 WRITE1   BAL     3,SETUP
00081A 1820                           709          LR      2,0
                                      710          PUT     PRINTER              FETCH NEXT FREE BUFFER
000826 D200 10CC CA0A 00000 01122     714          MVC     0(1,1),CARRCONT      ASSEMBLE PRINT LINE
00082C 95F1 CA0A      01122           715          CLI     CARRCONT,C'1'        TEST FOR SKIP
000830 4770 C12C              00838   716          BNE     NOSKIP
000834 9200 CA0B      01123           717          MVI     LINECNT,0            RESET LINE COUNT
000838 9240 CA0A      01122           718 NOSKIP   MVI     CARRCONT,C' '           AND CODE
00083C D283 1001 2000 00CC1 00000     719          MVC     1(LINELEN,1),0(2)
000842 4320 CA0B              01123   720          IC      2,LINECNT            INCREMENT LINE COUNT
000846 4120 2001              00001   721          LA      2,1(,2)
00084A 4220 CA0B              01123   722          STC     2,LINECNT
00084E 953C CA0B      01123           723          CLI     LINECNT,LINESMAX     TEST FOR PAGE OVERFLOW
000852 4740 C142              0085A   724          BL      WRITE2
000856 92F1 CA0A      01122           725          MVI     CARRCONT,C'1'        SET FOR SKIP
00085A 9500 C844      00F5C           726 WRITE2   CLI     LINECHK,0            TEST FOR LINE COUNTING
00085E 4780 C162              0087A   727          BE      PTRTN
000862 5800 CA0C              01124   728          L       0,LINELIM            DECREMENT LINE COUNT
000866 5B00 C554              00C6C   729          S       0,=F'1'
00086A 5000 CA0C              01124   730          ST      0,LINELIM
00086E 4780 C162              0087A   731          BNL     PTRTN                TERMINATE IF NEGATIVE
000872 41C0 CCC1              00CC1   732          LA      0,1                  INDICATE END OF DS
000876 47F0 C470              00B88   733          B       ERRPROC
00087A 9540 CACA      01122           734 PTRTN    CLI     CARRCONT,C' '        SET CONDITION CODE
00087E 47F0 C3A0              00AB8   735          B       RETURN1

000882 4530 C390              00AA8   737 PAGE1    BAL     3,SETUP
000886 92F1 CACA      01122           738          MVI     CARRCONT,C'1'        SET PAGE SKIP
00088A 47F0 C39E              00AB6   739          B       RETURN

00088E 4530 C390              00AA8   741 SPMASK1  BAL     3,SETUP
000892 4200 CA11              01129   742          STC     0,PMASK              SET PROGRAM CHECK MASK
000896 47F0 C39E              00AB6   743          B       RETURN

00089A 4530 C39C              00AA8   745 SYSREAD1 BAL     3,SETUP
```

```
  LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT
                                     746           READ    (1),SF,MF=E          ISSUE READ
0008AC 47F0 C39E                00AB6  751           B       RETURN
0008B0 4530 C390                00AA8  753 CHECK1    BAL     3,SETUP
                                     754           CHECK   (1)                  ISSUE CHECK
0008BE 47F0 C39E                00AB6  758           B       RETURN
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT
0008C2 4530 C3SO            00AA8   760 RDTAPE1  BAL     3,SETUP          SET UP REGISTERS
0008C6 4530 C3BE            00AD6   761          BAL     3,TCHECK         REJECT ALL BUT DEVICE 5
0008CA 9200 C83D      00F55         762          MVI     TMK,0            ASSUME NO TAPE MARK
0008CE 58D0 C8FC            01014   763          L       13,AP            LOAD CORE DESCRIPTOR POINTER
0008D2 58E0 D000            00000   764          L       14,0(,13)        LOAD LENGTH OF BLOCK
0008D6 49E0 C5B6            00CCE   765          CH      14,=H'4'
0008DA 4740 C414            00B2C   766          BL      DEVERR           END OF DATA SET IFF LENGTH < 4
0008DE 4770 C1CE            008E6   767          BNE     NOTTMK
0008E2 92FF C83D      00F55         768          MVI     TMK,X'FF'        TAPE MARK IFF LENGTH = 4
0008E6 15E1                         769 NOTTMK   CLR     14,1             FIND MINIMUM LENGTH FOR TRANSFER
0008E8 47B0 C1D6            003EE   770          BNL     RLENOK
0008EC 181E                         771          LR      1,14
0008EE 4120 E007            00007   772 RLENOK   LA      2,7(,14)         POINT TO NEXT CORE DESCRIPTOR
0008F2 5420 C594            00CAC   773          N       2,=F'-4'
0008F6 1BD2                         774          SR      13,2
0008F8 50D0 C8FC            01014   775          ST      13,AP
0008FC 41E0 D004            00004   776          LA      14,4(,13)        SET UP REGISTERS
000900 18F0                         777          LR      15,0
000902 18C1                         778          LR      0,1
000904 45D0 C3E4            00AEC   779          BAL     13,COREMOVE      TRANSFER DATA
000908 58E0 C800            00F18   780          L       14,S14
00090C 5000 E158            00158   781          ST      0,R1SAVE(,14)    RETURN RECORD LENGTH
000910 9500 C83D      00F55         782          CLI     TMK,0            SET CONDITION CODE
000914 47F0 C3AC            00AB8   783          B       RETURN1

000918 4530 C390            00AA8   785 WRTAPE1  BAL     3,SETUP          REJECT ALL BUT DEVICE 5
00091C 4530 C3BE            00AD6   786          BAL     3,TCHECK
000920 1B33                         787          SR      3,3              DETERMINE I/O STATE
000922 4330 C847            00F5F   788          IC      3,IOSTATE
000926 4323 C9C8            01020   789          IC      2,IONEXT(3)      SELECT NEXT STATE FROM
00092A 4220 C847            00F5F   790          STC     2,IOSTATE           TRANSITION TABLE
00092E 8930 0002            00002   791          SLL     3,2
000932 47F3 C21E            00936   792          B       WRSELECT(3)      EXECUTE APPROPRIATE ROUTINE
000936 47F0 C22E            00946   793 WRSELECT B       TSTATE0
00093A 47F0 C252            0096A   794          B       TSTATE1
00093E 47F0 C2B2            009CA   795          B       TSTATE2
000942 47F0 C2C0            009D8   796          B       TSTATE3

000946 583C C8FC            01014   798 TSTATE0  L       3,AP             STATE 0 -
00094A 4120 1007            00007   799          LA      2,7(,1)             ACCEPT RECTABLE
00094E 5420 C594            00CAC   800          N       2,=F'-4'            PREPARE TO PROCESS PROG SEG
000952 1B32                         801          SR      3,2
000954 5030 C900            01018   802          ST      3,APLINK         SAVE AP FOR LATER FIXUP
000958 5430 C56C            00C84   803          N       3,=F'-8'
00095C 5030 C9C4            0101C   804          ST      3,IOP            INITIALIZE SEGMENT I/O POINTER
000960 D7FF CA34 CA34 0114C 0114C   805          XC      SUPVPRT(256),SUPVPRT CLEAR LOCAL REFERENCE TABLE
000966 47FC C2C0            C09D8   806          B       TSTATE3          COMPLETE DATA TRANSFER

00096A 1810                         808 TSTATE1  LR      1,0              STATE 1 - ACCEPT SEGMENT HEADER
00096C D203 C910 1002 01028 00002   809          MVC     WORK(4),2(1)     RECOVER SEGMENT LENGTH
000972 5800 C9C4            0101C   810          L       0,IOP            ADJUST I/O POINTER
000976 5800 C910            01028   811          S       0,WORK
00097A 5400 C56C            00C84   812          N       0,=F'-8'
00097E 5500 C8F4            0100C   813          CL      0,LIMITP         TEST FOR ADEQUATE SPACE
000982 47B0 C286            C099E   814          BNL     TLOAD1
```

```
  LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT
000986 4520 C42C             0CB44   815          BAL     2,SRVMSG
00098A C9D5E2E4C6C6C9C3              816          DC      CL20'INSUFFICIENT STORAGE'
00099E 5C00 C9C4             0101C   817 TLOAD1   ST      0,IOP
0009A2 1B22                          818          SR      2,2
0009A4 4320 1001             00001   819          IC      2,1(,1)              RECOVER SEGMENT NUMBER
0009A8 8920 0CC2             0CCC2   820          SLL     2,2
0009AC 5810 C910             01028   821          L       1,WORK                  AND SEGMENT LENGTH
0009B0 1211                          822 TLOAD2   LTR     1,1
0009B2 4740 C39E             00AB6   823          BL      RETURN
0009B6 5002 CA34             0114C   824          ST      0,SUPVPRT(2)          MAKE PRT ENTRY
0009BA 4120 2CC4             00004   825          LA      2,4(,2)              STEP SEGMENT NUMBER
0009BE 4A00 C5F0             00DC8   826          AH      0,=H'4096'           INCREMENT BASE ADDRESS
0009C2 4B10 C5F0             00DC8   827          SH      1,=H'4096'           DECREMENT LENGTH
0009C6 47F0 C298             C09B0   828          B       TLOAD2

0009CA 18E0                          830 TSTATE2  LR      14,0                 STATE 2 - ACCEPT SEGMENT
0009CC 58F0 C9C4             0101C   831          L       15,IOP               SET UP MOVE SUBROUTINE
0009D0 45D0 C3E4             00AFC   832          BAL     13,COREMOVE
0009D4 47F0 C39E             00AB6   833          B       RETURN

0009D8 5830 C8FC             01014   835 TSTATE3  L       3,AP                 STATE 3 - STANDARD WRITE
0009DC 5010 3000             00000   836          ST      1,0(,3)              SAVE RECORD LENGTH
0009EC 4120 1CC7             00007   837          LA      2,7(,1)              LOCATE NEXT DESCRIPTOR
0009E4 5420 C594             00CAC   838          N       2,=F'-4'
0009E8 1B32                          839          SR      3,2
0009EA 5530 C8F4             0100C   840          CL      3,LIMITP             TEST FOR AREA OVERFLOW
0009EE 47B0 C2F2             00A0A   841          BNL     TWR1
0009F2 4520 C42C             00844   842          BAL     2,SRVMSG
0009F6 C9D5E2E4C6C6C9C3              843          DC      CL20'INSUFFICIENT STORAGE'
000A0A 92FF 3000       00000         844 TWR1     MVI     0(3),X'FF'           SET NEW LENGTH NEGATIVE
C00A0E 5030 C8FC             01014   845          ST      3,AP
000A12 41F0 3004             00004   846          LA      15,4(,3)             SET UP MOVE ROUTINE
000A16 18E0                          847          LR      14,0
000A18 45D0 C3E4             00AFC   848          BAL     13,COREMOVE
000A1C 47F0 C39E             00AB6   849          B       RETURN

000A20 4530 C390             00AA8   851 MKTAPE1  BAL     3,SETUP
000A24 4530 C3BE             00AD6   852          BAL     3,TCHECK
000A28 9501 C847       00F5F         853          CLI     IOSTATE,1            TEST FOR END OF SEGMENT OUTPUT
000A2C 4770 C336             00A4E   854          BNE     TMK1
000A30 9203 C847       00F5F         855          MVI     IOSTATE,3            SWITCH STATE FOR NORMAL WRITE
000A34 5810 C900             01018   856          L       1,APLINK             COMPUTE DESCRIPTOR FOR PROGRAM
000A38 1801                          857          LR      0,1
000A3A 5B00 C904             0101C   858          S       0,IOP
000A3E 5000 1CC0             00CC0   859          ST      0,0(,1)              STORE IT
000A42 5810 C904             0101C   860          L       1,IOP                UPDATE AP
000A46 5A10 C594             00CAC   861          A       1,=F'-4'
000A4A 5010 C8FC             01014   862          ST      1,AP
000A4E 4100 C598             00CB0   863 TMK1     LA      0,=F'-1'             SPECIFY MARK RECORD
000A52 4110 0CC4             00004   864          LA      1,4
000A56 47F0 C2C0             009D8   865          B       TSTATE3              WRITE

000A5A 4530 C390             00AA8   867 REWIND1  BAL     3,SETUP
000A5E 4530 C3BE             0CAD6   868          BAL     3,TCHECK
000A62 9300 C846       00F5E         869          TS      LIMITSET             TEST FOR CORRECT DEVICE 5 ORIGIN
```

```
  LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT
000A66 4770 C358             00A70   870          BNZ     RW2
000A6A D203 C8F4 C808 01C0C 00F2C   871          MVC     LIMITP(4),HICORE      INITIALIZE DEVICE 5 ORIGIN
000A7C D203 C8FC C8F8 01014 01010   872 RW2      MVC     AP(4),ORGP            RESET CORE DESCRIPTOR POINTER
000A76 47F0 C39E             00AB6   873          B       RETURN

000A7A 4530 C390             00AA8   875 FSPTM1   BAL     3,SETUP
000A7E 4530 C3BE             00AD6   876          BAL     3,TCHECK
000A82 58D0 C8FC             C1C14   877          L       13,AP                 LOAD CORE DESCRIPTOR POINTER
000A86 58E0 D000             00000   878 LOOK     L       14,0(,13)             LOAD LENGTH
000A8A 4120 ECC7             00007   879          LA      2,7(,14)              LOCATE NEXT DESCRIPTOR
000A8E 5420 C594             00CAC   880          N       2,=F'-4'
000A92 1BD2                  00000   881          SR      13,2                  POINT TO IT
000A94 49E0 C5B6             00CCE   882          CH      14,=H'4'              TEST LENGTH
000A98 4740 C414             00B2C   883          BL      DEVERR                END OF DATA SET IFF LENGTH < 4
000A9C 4770 C36E             00A86   884          BNE     LOOK                  TAPE MARK IFF LENGTH = 4
000AA0 50D0 C8FC             01014   885          ST      13,AP                 SAVE POINTER
000AA4 47F0 C39E             00AB6   886          B       RETURN
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT
000AA8 92FF C843      00F5B          888 SETUP    MVI     IOFLAG,X'FF'        INDICATE I/O EXIT
000AAC 50E0 C800              00F18  889          ST      14,S14              SAVE R14
000AB0 41D0 C848              00F6C  890          LA      13,SAVE             SUPPLY OS SAVE AREA
000AB4 07F3                          891          BR      3

000AB6 1B00                          893 RETURN   SR      0,0                 RETURN POINT - SET CC = 0
000AB8 58E0 C800              00F18  894 RETURN1  L       14,S14              RETURN POINT - CC PREVIOUSLY SET
000ABC 9200 C843      00F5B          895          MVI     IOFLAG,0            RESET I/O EXIT FLAG
000AC0 98C3 E144              00144  896          LM      12,3,RCSAVE(14)
000AC4 07FF                          897 IOEXIT   BR      15                  CHANGED TO NO-OP BY TIMER TRAP
000AC6 58C0 E140              00140  898          L       12,DSERV(,14)       RESTORE ADDRESSING
000ACA 92FF C3AD      00AC5          899          MVI     IOEXIT+1,X'FF'      RESTORE BRANCH
000ACE 4100 0006              00006  900          LA      0,6                 INDICATE TIMER INTERRUPT
000AD2 47F0 C470              00B88  901          B       ERRPROC
```

```
  LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT
000AD6 4920 C5F2             00D0A   903 TCHECK   CH      2,=H'5'                 TEST FOR DEVICE 5
000ADA 0783                          904          BCR     8,3
000ADC 4920 C5F4             00D0C   905          CH      2,=H'7'                 TEST FOR DEVICE 7
000AE0 4770 C414             00B2C   906          BNE     DEVERR
000AE4 4520 C42C             00B44   907          BAL     2,SRVMSG
000AE8 C9D5E2E4C6C6C9C3              908          DC      CL20'INSUFFICIENT STORAGE'

000AFC 5000 C91C             01028   910 COREMOVE ST      0,WORK                  SAVE R0
000B00 4100 0100             00100   911          LA      0,256
000B04 1910                          912 CORELOOP CR      1,0                     MOVE 256 BYTE BLOCKS
000B06 47D0 C402             00B1A   913          BNH     LASTMOVE
000B0A D2FF F000 E000 00000  00000   914          MVC     0(256,15),0(14)
000B10 1B10                          915          SR      1,0
000B12 1AE0                          916          AR      14,0
000B14 1AF0                          917          AR      15,0
000B16 47F0 C3EC             00BC4   918          B       CORELOOP
000B1A 1211                          919 LASTMOVE LTR     1,1                     MOVE REMAINING BLOCK
000B1C 47D0 C4CE             00B26   920          BNH     CORERTN
000B20 0610                          921          BCTR    1,0
000B22 4410 C504             00C1C   922          EX      1,COREMVC
000B26 5800 C91C             01028   923 CORERTN  L       0,WORK
000B2A 07FD                          924          BR      13                      RETURN


000B2C 4520 C42C             00B44   926 DEVERR   BAL     2,SRVMSG
000B30 C961D640C5D9D9D6              927          DC      CL20'I/O ERROR'
000B44 94BF C840       00F58         928 SRVMSG   NI      TMASK,X'FF'-TIMEON      CANCEL LIMIT CHECKS
000B48 9200 C844       00F5C         929          MVI     LINECHK,0
000B4C 58E0 C550             00C68   930          L       14,=A(SUPVPRT)
000B50 4100 C934             0104C   931          LA      0,BUFFER                WRITE MESSAGE
000B54 925C C934       01C4C         932          MVI     BUFFER,C'*'
000B58 D282 C935 C934 0104D  0104C   933          MVC     BUFFER+1(131),BUFFER
000B5E 45F0 CCCC             00724   934          BAL     15,WRITE
000B62 D283 C934 C933 0104C  0104B   935          MVC     BUFFER(132),BLANK
000B68 D213 C934 2000 0104C  00000   936          MVC     BUFFER(20),0(2)
000B6E 95FF C845       00F5D         937          CLI     ARUN,X'FF'              CHECK FOR EXECUTION
000B72 4780 C464             00B7C   938          BE      SRVMSG1
000B76 D213 C94E C580 01060  00C98   939          MVC     BUFFER+20(20),=C' DURING COMPILATION '
000B7C 45F0 C00C             00724   940 SRVMSG1  BAL     15,WRITE
000B80 58C0 C860             01278   941          L       12,JSPBASE+SUPVPRT-PRT
000B84 47F0 C000             00CC0   942          B       PURGE-JOBSEQ(,12)


000B88 94BF C840       00F58         944 ERRPROC  NI      TMASK,X'FF'-TIMEON      CANCEL LIMITS
000B8C 9200 C844       00F5C         945          MVI     LINECHK,0
000B90 58E0 C800             00F18   946          L       14,S14
000B94 D203 C890 E150 00FA8  00150   947          MVC     ERRADDR(4),RFSAVE(14)   SAVE LINK ADDRESS
000B9A D20F C894 E154 00FAC  00154   948          MVC     REGSAVE(16),R0SAVE(14)  ASSEMBLE REGISTERS
000BA0 9048 C8A4             00FBC   949          STM     4,11,REGSAVE+16
000BA4 D20F C8C4 E144 00FCC  00144   950          MVC     REGSAVE+48(16),RCSAVE(14)
000BAA 4160 C894             00FAC   951          LA      6,REGSAVE               SET UP CALL
000BAE 1870                          952          LR      7,0
000BB0 4180 C88C             00FA4   953          LA      8,ERRADDR-4
000BB4 1B00                          954          SR      0,0
000BB6 58F0 E0FC             000FC   955          L       15,63*4(,14)            CALL ALGOL RUNERROR
```

```
  LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT
000BBA 052F                          956          BALR    2,15
000BBC 58E0 C550                00C68 957          L       14,=A(SUPVPRT)
C0000C                               958          USING   PRT,14
000BC0 58C0 E12C                CC12C 959          L       12,JSPBASE
000BC4 47F0 C000                00000 960          B       PURGE-JOBSEQ(,12)     TERMINATE JOB AND PURGE
                                      961          DROP    14
```

```
  LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT
000BC8 58E0 C8CC             00F18   963 OSRETURN L       14,S14
000BCC 41D0 C848             00F60   964          LA      13,SAVE
                                     965          FREEMAIN V,A=GOTLIST          FREE SYSTEM RESOURCES
000BE0 5810 C810             00F28   973          L       1,PIRESET             RESTORE PICA ADDRESS
                                     974          SPIE    MF=(E,(1))
                                     976          CLOSE   (PRINTER,,READER,,SYSTEM)
000BFA 58D0 D004             00004   986          L       13,4(,13)             RETURN
                                     987          RETURN  (14,12)

                                     991          DROP    12
000C04                               992          USING   DCBEXIT,15
000C04 96C8 F1CC       00DD0         993 DCBEXIT  OI      READER+DCBOFLGS,X'08' SET UNLIKE-ATTRIBUTES BIT
000C08 9500 F352       00F56         994          CLI     RDRSTAT,0             TEST FOR PREVIOUS ENTRY
000C0C 4770 F012             00C16   995          BNE     SETRETRY
000C10 92FF F352       00F56         996          MVI     RDRSTAT,X'FF'          INDICATE PREVIOUS ENTRY
000C14 07FE                          997          BR      14
000C16 92FF F353       00F57         998 SETRETRY MVI     REREAD,X'FF'          REISSUE GET
000C1A 07FE                          999          BR      14
                                    1000          DROP    15
00003C                              1001 DCBOFLGS EQU     X'30'

000C1C D200 F000 E000 00000 000C0   1003 COREMVC  MVC     0(0,15),0(14)
000C22 C7F2                         1004 UDUMP    BR      2                     DUMMY
000C24                              1005 SUBRSW   DS      0F                    SUBROUTINE SELECTION TABLE
000C24 C000028C                     1006          DC      A(INITIAL)
000C28 CCC0C4CC                     1007          DC      A(PASSEXEC)
000C2C 0000045A                     1008          DC      A(RUNEXEC)
000C30 8500CC04                     1009 ADCBEXIT DC      X'85',AL3(DCBEXIT)    DCB EXIT CODE AND ADDRESS

000C34                              1011 LINKEND  DS      0H
000C38                              1012          LTORG
000C38 C1D3C7D6D34C4040             1013                =CL8'ALGOL'
000C40 D7D9D6C7D9C1D440             1014                =C'PROGRAM CHECK NO'
000C50 C5D6C64040404040             1015                =CL8'EOF'
000C58 CC000280                     1016                =F'640'
000C5C 0000003C                     1017                =F'60'
000C60 00000003                     1018                =A(BUFMOD-1)
000C64 00007FF8                     1019                =A(RECLEN)
000C68 0000114C                     1020                =A(SUPVPRT)
000C6C CC000001                     1021                =F'1'
000C70 00001068                     1022                =A(BUFFER+28)
000C74 000001F4                     1023                =A(DEFLINES)
000C78 CCC0C00A                     1024                =A(DEFTIME)
000C7C 0E000000                     1025                =XL4'0E000000'
000C80 08000000                     1026                =XL4'08000000'
000C84 FFFFFFF8                     1027                =F'-8'
000C88 00000618                     1028                =A(PROGINT)
000C8C 00000718                     1029                =A(SERVICE)
000C90 00000C34                     1030                =A(LINKEND)
000C94 0000000F                     1031                =F'15'
000C98 40C4E4D9C5D5C740             1032                =C' DURING COMPILATION '
000CAC FFFFFFFC                     1033                =F'-4'
000CB0 FFFFFFFF                     1034                =F'-1'
000CB4 C5D3C1D7E2C5C440             1035                =C'ELAPSED TIME IS   :  :'
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT
000CCA 0003                       1036                =H'3'
000CCC CCCC                       1037                =H'12'
000CCE 0004                       1038                =H'4'
000CDC C104                       1039                =AL2(MAXUSEG*4+8)
000CD2 E2E3C1D5C6D6D9C4           1040                =C'STANFORD ALGOL W (OVERLAY VERSION)'
000CF4 000A                       1041                =H'10'
000CF6 003C                       1042                =H'60'
000CF8 001E                       1043                =H'30'
000CFA C3C6D4C7C9C3C5D9           1044                =C'COMPILER ERROR'
000D08 1000                       1045                =H'4096'
000D0A 0005                       1046                =H'5'
000D0C 0007                       1047                =H'7'
000D0E 4CD5D6E340C1E5C1           1048                =C' NOT AVAILABLE '
000D1C 4F40C9D5E5C1D3C9           1049                =C'| INVALID LIMIT SPECIFICATION'
000D3A C00186                     1050                =AL3(PURGE)
```

```
 LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT

                                  1052 *                          JOB CONTROL DATA                                *

000D40                            1054 JSDSEG   DS      0D
                                  1055          PRINT   NOGEN
                                  1056 PRINTER  DCB     DSORG=PS,MACRF=PL,DDNAME=FT06F001,DEVD=DA,RECFM=FBA,   X
                                                        LRECL=LINELEN+1,BFTEK=S,EROPT=ABE
                                  1112 READER   DCB     DSORG=PS,MACRF=GM,DDNAME=FT05F001,DEVD=DA,RECFM=FB,    X
                                                        LRECL=80,BFTEK=S,SYNAD=DEVERR,EODAD=OSRETURN,          X
                                                        EROPT=ACC,EXLST=ADCBEXIT
                                  1168 SYSTEM   DCB     DSORG=PS,MACRF=RP,DDNAME=COMPILER,DEVD=DA,RECFM=UT,    X
                                                        BLKSIZE=32760,NCP=2,SYNAD=DEVERR,EODAD=DEVERR

000E58 CC000E7400000E60           1222 DECBTAB  DC      A(DECB2),A(DECB1)

                                  1224 SYS1     READ    DECB1,SF,SYSTEM,,32760,MF=L
                                  1233 SYS2     READ    DECB2,SF,SYSTEM,,32760,MF=L

000E88                            1243 PRTBASE  DS      0F                      PRT POINTER TABLE
000E88 0000114C                   1244          DC      A(SUPVPRT)              SUPERVISOR PRT
000E8C                            1245          DS      4F                      COMPILER PRT POINTERS

000EA0                            1247 PROCREC  DS      0D                      PROCESS CONTROL RECORDS
000000                            1248 LOAD     EQU     0                          LOAD PROCESS ACTIVATION
000001                            1249 EXEC     EQU     1                          EXECUTION PROCESS ACTIVATION
000002                            1250 LOADID   EQU     2                       ID # OF PROGRAM TO BE LOADED
000004                            1251 EXECID   EQU     4                       ID # OF PROGRAM TO BE EXECUTED
000006                            1252 INDEX    EQU     6                       INDEX TO SUBROUTINE TABLE
000008                            1253 NORMNEXT EQU     8                       NORMAL NEXT STATE
00000A                            1254 ERRNEXT  EQU     10                       ERROR NEXT STATE

                                  1256          AIF     (&SWAP).L25
                                  1257 .L25     ANOP
000EA0 00FFCCCC00C00000           1258 STATE0   DC      X'00',X'FF',H'0',H'0',H'0',H'1',H'0'  INITIALIZE
000EAC C10000C4CCCC0000           1259 STATE1   DC      X'01',X'00',H'4',H'0',H'0',H'2',H'0'  LOAD PASS 1
000EB8 00FF000C00040004           1260 STATE2   DC      X'00',X'FF',H'0',H'4',H'4',H'3',H'0'  EXEC PASS 1
000EC4 C2CCCCC8CC000000           1261 STATE3   DC      X'02',X'00',H'8',H'0',H'0',H'4',H'0'  LOAD PASS 2
000ED0 00FF000000080004           1262 STATE4   DC      X'00',X'FF',H'0',H'8',H'4',H'5',H'0'  EXEC PASS 2
000EDC C20CC00CCCC00000           1263 STATE5   DC      X'02',X'00',H'12',H'0',H'0',H'6',H'0'  LOAD PASS 3
000EE8 00FFCCC00CCC0004           1264 STATE6   DC      X'00',X'FF',H'0',H'12',H'4',H'7',H'0'  EXEC PASS 3
000EF4 C100001000000000           1265 STATE7   DC      X'01',X'00',H'16',H'0',H'0',H'8',H'0'
000F00 00FFCCCCCC100008           1266 STATE8   DC      X'00',X'FF',H'0',H'16',H'8',H'0',H'0'  ALGOLRUN

000F0C 00000000                   1268 BUFCBPTR DC      A(BUFMOD-4)             BUFFER CONTROL BLOCK INDEX
000F10                            1269 BUFADDR  DS      A                       BUFFER CONTROL BLOCK
                                  1270 .L26     ANOP
000F14                            1271 ISAVE    DS      A                       CONTROL RECORD POINTER
000F18                            1272 S14      DS      A                       R14 SAVE WORD
000F1C                            1273 R2SAVE   DS      F
000F20                            1274 HICORE   DS      A                       HIGHEST COMMON LOCATION USED
000F24                            1275 TCBADR   DS      A                       TCB ADDRESS
000F28                            1276 PIRESET  DS      F                       PICA ADDRESS SAVE AREA
000F2C                            1277 TSAVE2   DS      F
000F30                            1278 TSAVE    DS      6F                      TIME ROUTINE SAVE AREA
000F48                            1279 TIMELIM  DS      F
000F4C C15F9000                   1280 COMPLIM  DC      F'23040000'             10 MINUTES (IN TUNITS)
```

```
  LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT
000F50                             1281 TINTLOC  DS      F                 TIMER INTERRUPT LOCATION
000F54                             1282 TINTBYTE DS      X                 SAVED INSTRUCTION BYTE
000F55 CC                          1283 TMK      DC      X'00'             TAPE MARK FLAG

000F56 0000                        1285 RDRSTATS DC      2X'00'            READER STATUS FLAG
000F56                             1286 RDRSTAT  EQU     RDRSTATS+0        SET AFTER FIRST OPEN
000F57                             1287 REREAD   EQU     RDRSTATS+1        SET IF GET MUST BE REISSUED

000F58 CCCCCCCCCCCCCCCC            1289 STATS    DC      8X'00'            STATUS FLAGS
000F58                             1290 TMASK    EQU     STATS+0           TIMER STATUS BITS
000080                             1291 TIMESET  EQU     X'80'             TIMER STARTED
000040                             1292 TIMEON   EQU     X'40'             TIMER ENABLED
000020                             1293 TIMEOUT  EQU     X'20'             TIMER EXPIRED
000010                             1294 TR13     EQU     X'10'             PRT POINTER IN SAVE AREA
000F5A                             1295 INTFLAG  EQU     STATS+2           INTERRUPT PROCESSING
000F5B                             1296 IOFLAG   EQU     STATS+3           I/O PROCESSING
000F5C                             1297 LINECHK  EQU     STATS+4           LINE COUNTING
000F5D                             1298 ARUN     EQU     STATS+5           ALGOLRUN LOADED
000F5E                             1299 LIMITSET EQU     STATS+6           DEVICE 5 LIMIT SET
000F5F                             1300 IOSTATE  EQU     STATS+7           DEVICE 5 STATUS SETTING

000F60                             1302 SAVE     DS      18F               OS SAVE AREA
000FA8                             1303 ERRADDR  DS      A
000FAC                             1304 REGSAVE  DS      16F               REGISTER ASSEMBLY AREA
000FEC                             1305 SSAVE13  DS      F                 SPECIAL R13 SAVE AREA
000FFC                             1306 PISAVE   DS      5F
001004                             1307 COMMLIM  DS      2F
00100C                             1308 IOLIM    DS      2F
001010                             1309 CRGP     EQU     IOLIM+4           UPPER I/O BUFFER BOUND
00100C                             1310 LIMITP   EQU     IOLIM+0           LOWER I/O BUFFER BOUND
001014                             1311 AP       DS      F                 I/O POINTER
001018                             1312 APLINK   DS      A                 LINK FOR ADJUSTING AP
00101C                             1313 IOP      DS      A                 SEGMENT LOAD POINTER
001020 01020103                    1314 IONEXT   DC      X'01020103'       DEVICE 5 STATE TRANSITION VECT
001028                             1315 CONWORK  DS      D                    WORK AREA
001028                             1316 WORK     EQU     CONWORK
001030 4C404C4C4C404040            1317 DATE     DC      CL8' '            CURRENT DATE
001038 00019C0000080000            1318 ASKLIST  DC      A(COREMIN),A(COREMAX)
001040                             1319 GOTLIST  DS      2F
001048                             1320 SUCCESS  DS      X           PROGRAM EXECUTION RETURN CODE
001049 0000                        1321 CHARSW   DC      2X'00'            LIMIT FIELD STATE SWITCHES
001049                             1322 COLON    EQU     CHARSW+0
00104A                             1323 COMMA    EQU     CHARSW+1
00104B 40                          1324 BLANK    DC      C' '
00104C                             1325 BUFFER   DS      CL132             MONITOR PRINT BUFFER
0010D0                             1326 CARDBUF  DS      CL80              JOB CARD BUFFER

                                   1328 *                          UNIT RECORD VARIABLES                          *
001120 00                          1329 EOF      DC      X'00'             READER END-OF-FILE
001121                             1330 CCFLAG   DS      X                 CONDITION CODE FLAG
001122 F1                          1331 CARRCONT DC      C'1'              USASA CARRIAGE CONTROL CODE
001123 00                          1332 LINECNT  DC      X'00'             PRINTER LINE COUNT
001124                             1333 LINELIM  DS      F                 JOB LINE LIMIT

001128                             1335 FLAGS    DS      0F
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT
001128 FFFF0000                   1336           DC       X'FFFF0000'
001129                            1337 PMASK     EQU      FLAGS+1              PROGRAM CHECK MASK

00112C                            1339 PCHKTBL   DS       16H                  PROGRAM CHECK COUNT TABLE
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                                    7 FEB 69

                                   1341 *                          PROGRAM REFERENCE TABLE                 *

CC0000                             1343 PRT      DSECT
                                   1344 *         EACH PROGRAM HAS A PRT OF THE FOLLOWING FORM

000000                             1346          DS      64F                 0-63 COMPILER/USER SEGMENTS
00010C CCCCCCCC                    1347 PROGLEN  DC      A(0)                64  HIGH PROGRAM ADDRESS
000104 CCCCCCCC                    1348 DATALOW  DC      A(0)                65  LOW NON-REUSABLE DATA BASE
000108 00000000                    1349 DATALNG  DC      A(0)                66  NON-REUSABLE DATA LNGGTH
00010C CCCCCCCCCCCC0000            1350          DC      2F'0'               67-68 RESERVED
000114 C00000F0                    1351 GTIMBASE DC      A(GETTIME)          69  GET ELAPSED TIME
000118 0000012E                    1352 WTIMBASE DC      A(WRITTIME)         70  WRITE ELAPSED TIME
00011C CC000C22                    1353          DC      A(UDUMP)            71  USER DUMP ROUTINE
000120 00000CCCCCC00000            1354          DC      2F'0'               72-73 RESERVED
000128 C0000D40                    1355 JSDBASE  DC      A(JSDSEG)           74  JOB SEQUENCE DATA SEGMENT
00012C 00000186                    1356 JSPBASE  DC      A(JOBSEQ)           75  JOB SEQUENCE PROG SEGMENT
000130 000000C0CCC00000            1357          DC      2F'0'               76-77 RESERVED
000138 CCCCCCCC                    1358          DC      A(0)                78  TAPE LOAD ROUTINE
00013C 00CCCCCC                    1359          DC      F'0'                79  RESERVED
000140 C0000718                    1360          DC      A(SERVICE)          80  SUPERVISOR ENTRY POINT
000144                             1361          DS      8F                  REGISTER SAVE AREA
000164                             1362          DS      3F                  SAVE AREA
C00000                             1363 ALGOLW02 CSECT

00114C                             1365 SUPVPRT  DS      0F                  SUPERVISOR PRT
00114C                             1366          DS      64F
00124C CCCCCCCC                    1367          DC      A(0)                64  HIGH PROGRAM ADDRESS
001250 00000000                    1368          DC      A(0)                65  LOW NON-REUSABLE DATA BASE
001254 CC000C00                    1369          DC      A(0)                66  NON-REUSABLE DATA LENGTH
001258 0C00C0CCCCCC0000            1370          DC      2F'0'               67-68 RESERVED
001260 000000F0                    1371          DC      A(GETTIME)          69  GET ELAPSED TIME
001264 CC00C12E                    1372          DC      A(WRITTIME)         70  WRITE ELAPSED TIME
001268 C0000C22                    1373          DC      A(UDUMP)            71  USER DUMP ROUTINE
00126C CCC0C0CC00000000            1374          DC      2F'0'               72-73 RESERVED
001274 CC00CD40                    1375          DC      A(JSDSEG)           74  JOB SEQUENCE DATA SEGMENT
001278 00000186                    1376          DC      A(JOBSEQ)           75  JOB SEQUENCE PROG SEGMENT
00127C 0C000C0CC00000000           1377          DC      2F'0'               76-77 RESERVED
001284 CC000C00                    1378          DC      A(0)                78  TAPE LOAD ROUTINE
001288 00000C000                   1379          DC      F'0'                79  RESERVED
00128C CC00C718                    1380          DC      A(SERVICE)          80  SUPERVISOR ENTRY POINT
001290                             1381          DS      8F                  REGISTER SAVE AREA
001280                             1382          DS      3F                  SAVE AREA

000144                             1384 RCSAVE   EQU     324
000150                             1385 RFSAVE   EQU     336
000154                             1386 ROSAVE   EQU     340
000158                             1387 RISAVE   EQU     344
000140                             1388 DSERV    EQU     4*80
                                   1389          END
```

SYMBOL	LEN	VALUE	DEFN	REFERENCES													
ADCBEXIT	1	000C30	1009	1138													
ALGOLW02	1	CCC000	16	1363													
AP	4	CC1014	1311	104	461	763	775	798	835	845	862	872	877	885			
APLINK	4	001018	1312	802	856												
ARUN	1	000F5D	1298	447	595	937											
ASKLIST	4	001038	1318	73													
BLANK	1	00104B	1324	170	290	363	488	935									
BUFADDR	4	000F10	1269	97	224	253											
BUFCBPTR	4	000F0C	1268	220	223	252	288										
BUFFER	132	00104C	1325	170	171	173	174	183	290	292	294	295	298	307	309	310	311
				311	312	330	332	362	363	365	440	488	489	490	500	501	505
				506	599	603	604	604	606	607	608	608	931	932	933	933	935
				936	939	1022											
BUFMOD	1	000004	29	222	1018	1268											
CARDBUF	80	001000	1326	298	704												
CARRCONT	1	CC1122	1331	291	305	429	448	714	715	718	725	734	738				
CCFLAG	1	001121	1330	689	700	705											
CHAIN	6	CCC5C2	535	538													
CHARSW	1	001049	1321	329	329	1322	1323										
CHECK	4	00079C	675	246													
CHECK1	4	CCC8B0	753	677													
CHKLOOP	4	00021C	245	248													
COLON	1	CC1049	1322	345	355												
COMMA	1	CC104A	1323	340	360												
COMMLIM	4	001004	1307	99	398												
COMPLIM	4	CCCF4C	1280	150	381												
CONVERT	4	C0010C	147	143													
CONWORK	8	001028	1315	178	179	499	500	504	505	1316							
CORELOOP	2	CC0B04	912	918													
COREMAX	1	C80000	25	1318													
COREMIN	1	C19000	24	1318													
COREMOVE	4	CCCAFC	910	779	832	848											
COREMVC	6	000C1C	1003	922													
CORERTN	4	CCCB26	923	920													
CVTPTR	1	000010	558	530													
CVTTCBP	1	000000	559	531													
CYCLE	4	CCC1A6	212	120	276	278											
DATALNG	4	000108	1349														
DATALOW	4	0CC104	1348														
DATE	8	CC1030	1317	109	110	110	111	292									
DCBEXIT	4	000C04	993	992	1009												
DCBOFLGS	1	CCC030	1001	993													
DECBTAB	4	000E58	1222	226	245												
DECB1	4	000E60	1226	1222													
DECB2	4	CCCE74	1235	1222													
DEFLINES	1	0001F4	34	371	1023												
DEFTIME	1	CCC00A	33	376	1024												
DEVERR	4	CCCB2C	926	766	883	906	1151	1192	1207								
DSERV	1	000140	1388	651	654	657	661	664	667	670	673	676	679	683	898		
ENDMAP	6	CC0262	266	258													
EOF	1	001120	1329	296	301	303	443	703									
EOFFLAG	1	0002AF	422	443													
ERRADDR	4	0CCFA8	1303	947	953												
ERRNEXT	1	00000A	1254	277													
ERRPROC	4	CCCB88	944	733	901												

SYMBOL	LEN	VALUE	DEFN	REFERENCES												
EXEC	1	CCCCC1	1249	233												
EXECID	1	000004	1251	235												
FLAGS	4	CC1128	1335	1337												
FSPTM	4	0007A8	678	450	451											
FSPTM1	4	000A7A	875	680												
GETTIME	4	CCC0F0	135	134	137	155	1351	1371								
GOTLIST	4	001040	1319	74	78	79	84	90	969							
GTIMBASE	4	000114	1351	165												
HICORE	4	0C0F20	1274	407	871											
I	1	00000A	19	119	119	208	208	212	213	213	216	233	235	237	242	267	275
				275	277	277	393	413	425	482							
INDEX	1	CCCC06	1252	237												
INITIAL	4	00028C	281	1006												
INTEXIT	2	CCC66A	589	283	545	590										
INTFLAG	1	CCCF5A	1295	525	568	586										
INTMSG	4	000684	597													
IOEXIT	2	000AC4	897	284	549	899										
IOFLAG	1	000F5B	1296	527	888	895										
IOLIM	4	00100C	1308	1309	1310											
IONEXT	4	0C1020	1314	789												
IOP	4	00101C	1313	804	810	817	831	858	860							
IOSTATE	1	000F5F	1300	788	790	853	855									
IPL	2	CCC000	40	38												
ISAVE	4	000F14	1271	393	413	425	482									
IO	6	CCC2A8	290	314	367											
I1	4	0002D4	300	297	302											
I2	4	0002E0	303	299												
I3	4	0CC316	316	308												
JOBSEG	2	000186	195	116	191	942	960	1356	1376							
JSDBASE	4	CCC128	1355	114	196	406	472									
JSDSEG	8	CC0D40	1054	47	48	123	192	1355	1375							
JSPBASE	4	00012C	1356	115	197	405	471	941	959							
LASTMOVE	2	000B1A	919	913												
LDLOOP	4	0001D4	226	232												
LIM	1	000174	420	464												
LIMITP	4	00100C	1310	813	840	871										
LIMITSET	1	000F5E	1299	869												
LINECHK	1	CCCF5C	1297	459	598	726	929	945								
LINECNT	1	CC1123	1332	306	427	717	720	722	723							
LINELEN	1	000084	27	719	1107											
LINELIM	4	CC1124	1333	372	728	730										
LINESMAX	1	00003C	28	427	723											
LINKEND	2	000C34	1011	547	1030											
LOAD	1	CCCCCC	1248	216	242											
LOADID	1	000002	1250	267												
LOOK	4	CCCA86	878	884												
L1	4	CCC340	331	335	344	349	352	356	361							
L2	4	000376	345	341												
L3	4	CC0388	350	346												
L4	4	000392	353	337												
L5	4	0003A6	358	354												
L6	4	0CC3B6	362	357	359											
L7	2	0003D2	368	333												
L8	4	0003DE	372	370												
L9	4	CCC3EC	377	375												

SYMBOL	LEN	VALUE	DEFN	REFERENCES											
MAPINCR	4	00025A	264	261											
MAPLCOP	4	000240	257	265											
MAXASEG	1	000035	31	438											
MAXUSEG	1	00003F	30	257	1039										
MKTAPE	4	000778	666												
MKTAPE1	4	000A20	851	668											
MONFLAG	1	0007F9	390	401											
MP	1	000170	419	463											
NORMNEXT	1	000008	1253	275											
NOSKIP	4	000838	718	716											
NOTEOF	4	00080E	705	699	702										
NOTIMER	4	00063A	576	571											
NOTTMK	2	000886	769	767											
ORGP	4	001010	1309	103	872										
OSRETURN	4	0008C8	963	1136											
PAGE	4	000730	656												
PAGE1	4	000882	737	658											
PASSEXEC	4	00040C	392	1007											
PCHKTBL	2	00112C	1339	117	117	486	486	496	498	509	509	581	583		
PIEPSW	1	000004	639	576	588	625	636								
PIERE	1	00000C	640	614	621	635									
PIERO	1	000014	641	615											
PIRESET	4	000F28	1276	69	973										
PISAVE	4	000FF0	1306	569	587	629	634								
PMASK	4	001129	1337	395	446	578	742								
PRBEND	4	000508	539	536											
PRGFAIL	4	000284	277	274											
PRINTER	4	000D40	1061	54	711	980									
PROCREC	8	000EA0	1247	213											
PROGINT	4	000618	568	65	541	567	601	611	1028						
PROGINT1	4	00065E	586	633											
PROGINT2	4	000678	594	585											
PROGINT3	4	0006BC	614	575	592	596									
PROGINT4	4	0006D6	620	618											
PROGINT5	4	000706	634	613											
PROGLEN	4	000100	1347	449											
PRT	1	000000	1343	113	131	193	941	958							
PRTBASE	4	000E88	1243	236	269										
PTRTN	4	00087A	734	727	731										
PURGE	4	000186	196	636	942	960	1050								
P1	4	000450	413	409											
RCSAVE	1	000144	1384	650	653	656	660	663	666	669	672	675	678	682	896	950
RDRSTAT	1	000F56	1286	994	996											
RDRSTATS	1	000F56	1285	49	49	1286	1287									
RDTAPE	4	000760	660	437											
RDTAPE1	4	0008C2	760	662											
READ	4	000718	650	300	441										
READER	4	000DA0	1117	56	691	982	993								
READ1	4	0007CC	688	652											
READ2	4	0007D4	691	697											
READ3	4	0007EE	698	695											
RECLEN	1	007FF8	32	231	402	1019									
RECTAB	1	0001AC	421	434											
REGSAVE	4	000FAC	1304	615	620	621	623	630	948	949	950	951			
REREAD	1	000F57	1287	694	696	998									

SYMBOL	LEN	VALUE	DEFN	REFERENCES
RETURN	2	0C0AB6	893	739 743 751 758 823 833 849 873 886
RETURN1	4	000AB8	894	706 735 783
REWIND	4	0C0784	669	
REWIND1	4	00CA5A	867	671
RFSAVE	1	000150	1385	947
RLENOK	4	0CC8EE	772	77C
RUNEXEC	4	00045A	424	1008
RW2	6	CCCA70	872	87C
R0SAVE	1	CCC154	1386	948
R1SAVE	1	000158	1387	781
R2SAVE	4	0C0F1C	1273	281 385 392 414 424 483
R3	4	00046E	43C	428
R4	4	00048E	440	442
R5	4	CCC51E	482	477
S	1	C00009	20	318 324 368 384 452 460
SAVE	4	CCCF60	1302	43 122 140 199 316 453 478 890 964
SC1	4	C0C546	493	497 508
SC2	6	000588	509	495
SELECT	4	000274	273	244
SERVICE	4	000718	649	543 648 1029 1360 1380
SETBYTE	4	C00604	551	542 548
SETINT	4	0C05EC	545	526
SETIO	4	0005FC	549	528
SETRETRY	4	000C16	998	995
SETUP	4	CCCAA8	888	688 708 737 741 745 753 760 785 851 867 875
SHOWCHK	6	000528	486	285 432
SPMASK	4	CCC7C0	682	
SPMASK1	4	00088E	741	684
SRVMSG	4	000B44	928	815 842 907 926
SRVMSG1	4	CCCB7C	94C	938
SSAVE13	4	000FEC	1305	616 631
STATEC	1	CCCEA0	1258	
STATE1	1	000EAC	1259	
STATE2	1	000EB8	1260	
STATE3	1	CCCEC4	1261	
STATE4	1	000ED0	1262	
STATE5	1	CCCEDC	1263	
STATE6	1	0CCEE8	1264	
STATE7	1	000EF4	1265	
STATE8	1	CCCF00	1266	
STATS	1	000F58	1289	282 282 1290 1295 1296 1297 1298 1299 1300
SUBRSW	4	000C24	1005	238
SUCCESS	1	001048	1320	214 273 410
SUPVPRT	4	00114C	1365	112 127 240 266 438 597 635 805 805 824 930 941 957 1020 1244
SYSFREE	1	CCC8CC	26	79 88 89 124
SYSREAD	4	000790	672	228
SYSREAD1	4	CCC89A	745	674
SYSTEM	4	C00E00	1173	58 204 321 984 1230 1239
SYS1	4	000E60	1225	
SYS2	4	000E74	1234	
S14	4	000F18	1272	780 889 894 946 963
T	1	000008	21	220 221 221 222 223 224 235 236 237 238 252 253 267 269 328 339 343 348 351
TCBADR	4	000F24	1275	533 535

SYMBOL	LEN	VALUE	DEFN	REFERENCES											
TCBCURR	1	CCC004	560	532											
TCBRBP	1	000000	561	534											
TCHECK	4	000AD6	903	761	786	852	868	876							
TDIFF	2	CCC11C	151	149											
TEST1	2	000182	215												
TEST2	4	0CC1F0	233	218	230										
TEST3	2	00C20E	241	234											
TEXIT	4	000612	554	546	550										
TIMEINT	4	CCC590	521	456	520										
TIMELIM	4	000F48	1279	147	379	455									
TIMEON	1	000040	1292	458	473	522	928	944							
TIMEOUT	1	CCC020	1293	142	476	521	570								
TIMESET	1	000080	1291	148	458										
TINTBYTE	1	000F54	1282	552	573										
TINTLOC	4	CCCF50	1281	551	572										
TLOAD1	4	00099E	817	814											
TLOAD2	2	CCC9B0	822	828											
TLOOP	2	000158	176	182											
TMASK	1	000F58	1290	139	142	148	157	458	473	476	521	522	570	617	928	944
TMK	1	CCCF55	1283	762	768	782									
TMK1	4	000A4E	863	854											
TR13	1	CCC010	1294	139	157	617									
TSAVE	4	CCCF30	1278	135	154	186	619								
TSAVE2	4	000F2C	1277	164	185										
TSTATE0	4	0CC946	798	793											
TSTATE1	2	CCC96A	808	794											
TSTATE2	2	0009CA	830	795											
TSTATE3	4	CCC9D8	835	796	806	865									
TTEST1	4	0005F4	547	544											
TWR1	4	000A0A	844	841											
UDUMP	2	CCCC22	1004	1353	1373										
WORK	8	001028	1316	108	109	809	811	821	910	923					
WRITE	4	CCC724	653	172	184	304	313	366	507	605	609	934	940		
WRITE1	4	CCC816	708	655											
WRITE2	4	00085A	726	724											
WRITTIME	4	CCC12E	164	163	168	1352	1372								
WRSELECT	4	000936	793	792											
WRTAPE	4	00C76C	663												
WRTAPE1	4	0CC918	785	665											
WTIMBASE	4	000118	1352	167	206	411	430	474							
XRBLINK	1	0CC01C	562	535	537										
XRBPSW	1	000010	563	539											

NO STATEMENTS FLAGGED IN THIS ASSEMBLY

```
IEF285I   SYS1.MACLIB                                    KEPT
IEF285I   VOL SER NOS= SYSCO .
IEF285I   SYS2.CCLIB                                     KEPT
IEF285I   VOL SER NOS= SYSO1 .
IEF285I   SYS2.DUMMYMAC                                  KEPT
IEF285I   VOL SER NOS= SYSO1 .
IEF285I   SYS1.UT1                                       KEPT
IEF285I   VOL SER NOS= CAMP08.
IEF285I   SYS1.UT2                                       KEPT
IEF285I   VOL SER NOS= SYSO2 .
IEF285I   SYS1.UT3                                       KEPT
IEF285I   VOL SER NOS= SYSO3 .
IEF285I   SYSOUT                                         SYSOUT
IEF285I   VOL SER NOS=       .
IEF285I   LOADSET.MONASM                                 PASSED
IEF285I   VOL SER NOS= SYSO2 .
IEF285I   AAAAAAAA.AAAAAAAA.AAAAAAAA.AAAAAAAA.00000806 DELETED
IEF285I   VOL SER NOS=       .
//LKED   EXEC  PGM=IEWL,PARM=(LET,LIST,MAP),COND=(5,LT,ASM)          00001200
//SYSLIN DD    DSNAME=&LOADSET,DISP=(OLD,DELETE)                     00001300
//       DD    DDNAME=SYSIN                                          00001400
//SYSLMOD DD  DSNAME=&GOSET(GO),UNIT=2314,SPACE=(CYL,(12,1,1)),      *00001500
//             DISP=(MOD,PASS)                                       00001600
//SYSUT1   DD  DSNAME=SYS1.UT1,DISP=OLD,DCB=(KEYLEN=0)               00001700
//SYSPRINT DD  SYSOUT=A                                              00001800
//SYSLIB   DD   DSNAME=SYS1.FORTLIB,DISP=OLD                         00001900
//         DD  DSNAME=SYS2.SSPLIB,DISP=OLD                           00002000
//         DD  DSNAME=SYS2.SUBLIB1,DISP=OLD                          00002100
//
IEF236I ALLOC. FOR MONASM   LKED     STEP
IEF237I SYSLIN   ON 230
IEF237I SYSLMOD  ON 230
IEF237I SYSUT1   ON 1C0
IEF237I SYSLIB   ON 235
IEF237I          ON 331
IEF237I          ON 331
```

```
   E-LEVEL LINKAGE EDITOR OPTIONS SPECIFIED LET,LIST,MAP
IEW0132  IHESACA
IEW0132  IHESACB
IEW0132  IHESAPC
IEW0132  IHEDMAA
IEW0132  IHEDNCA
IEW0132  IHECSMF
IEW0132  IHEBSMZ
IEW0132  IHEBSKA
IEW0132  IHEICXA
IEW0132  IHEICXB
IEW0132  IHEIOPB
IEW0132  IHEICPA
IEW0132  IHEDICA
IEW0132  IHEDCBB
IEW0132  IHEDIBA
IEW0132  IHEDOBA
IEW0132  IHECIAA
IEW0132  IHEDOAA
IEW0132  IHEOCLB
IEW0132  IHEICBC
IEW0132  IHEIOAT
IEW0132  IHEICAA
IEW0132  IHEICBT
IEW0132  IHEIOBA
IEW0132  IHECCLA
IEW0132  IHEIOGA
IEW0132  IHEOSEA
IEW0132  IHESAFA
IEW0132  IHESAFB
IEW0132  ALPHAUP
IEW0132  IHEVPB
IEW0132  IHEVFA
IEW0132  IHEVFC
IEW0132  IHEDMA
IEW0132  IHECDA
IEW0132  IHEDCN
IEW0132  IHEVFB
IEW0132  IHEVPA
IEW0132  IHEUPA
IEW0132  IHEVPE
IEW0132  IHEVPG
IEW0132  IHEDIA
IEW0132  IHEVPC
IEW0132  IHEVPF
IEW0132  IHEVSC
IEW0132  IHEDNC
IEW0132  IHEVFE
IEW0132  IHEVSC
IEW0132  IHEVCA
IEW0132  IHEVQB
IEW0132  IHEVQC
****GO        DOES NOT EXIST BUT HAS BEEN ADDED TO DATA SET
```

DIAGNOSTIC MESSAGE DIRECTORY

IEW0132 ERROR - SYMBOL PRINTED IS AN UNRESOLVED EXTERNAL REFERENCE.

MODULE MAP

CONTROL SECTION

NAME	ORIGIN	LENGTH
ALPHA	C0	1D1E
**ALPHAA	1D2C	2E57F
IHEMAIN	302A0	4
IHENTRY	3C2A8	C
SYSIN	3C2B8	38
IHESPRT	3C2F0	38
DATAIN	3C328	38
SYSPNCH	3C360	38
LINEFIL	3C398	38

ENTRY

NAME	LOCATION	NAME	LOCATION	NAME	LOCATION	NAME	LOCATION

PSEUDO REGISTERS

NAME	ORIGIN	LENGTH	NAME	ORIGIN	LENGTH	NAME	ORIGIN	LENGTH	NAME	ORIGIN	LENGTH
IHEQINV	C0	4	IHEQERR	4	4	IHEQLWF	8	4	IHEQSLA	C	4
**ALPHAB	1C	4	**ALPHAC	14	4	**ALPHAD	18	4	**ALPHAE	1C	4
SYSIN	20	4	IHEQSPR	24	4	DATAIN	28	4	SYSPNCH	2C	4
LINEFIL	3C	4									

TOTAL LENGTH OF PSEUDO REGISTERS 34
ENTRY ADDRESS 302A8
TOTAL LENGTH 303D0

```
IEF285I   LCACSET.MCNASM                          DELETED
IEF285I   VOL SER NOS= SYSC2 .
IEF285I   GCSET.MCNASM                            PASSED
IEF285I   VOL SER NOS= SYSC2 .
IEF285I   SYS1.UT1                                KEPT
IEF285I   VOL SER NOS= CAMP08.
IEF285I   SYSCLT                                  SYSOUT
IEF285I   VOL SER NOS=       .
IEF285I   SYS1.FORTLIB                            KEPT
IEF285I   VOL SER NOS= SYS00 .
IEF285I   SYS2.SSPLIB                             KEPT
IEF285I   VOL SER NOS= SYS01 .
IEF285I   SYS2.SUBLIB1                            KEPT
IEF285I   VOL SER NOS= SYS01 .
IEF285I   GOSET.MCNASM                            DELETED
IEF285I   VOL SER NOS= SYS02 .


H A S P  JOB STATISTICS --  1,160 CARDS READ --    1,557 LINES PRINTED --     91 CARDS PUNCHED --   0.92 MINUTES EXECUTION TIME
         480 I/O CALLS              535 SVC CALLS        0.42 MINUTES CPU TM    TIME=  17:53:26 DATE=  02/07/69
```

BIN	914	*PUNCH*	SEQ	951	SATTERTHWAITE	END	17:53	02/07	END	SATTERTHWAITE	SEQ	951	*PUNCH*	BIN	914

```
//EHSPL360 JOB EHS$CG,107,CLASS=E,PRTY=8,REGION=300K                    JOB 273
//JOBLIB   DD  DSNAME=PUB.EHS.PL360,DISP=(SHR,PASS)
//PL360  EXEC  PGM=PL360,PARM='LOAD,NODECK'
//SYSPRINT DD  SYSOUT=A,DCB=BLKSIZE=2660
//SYSGO    DD  DSNAME=&&LOADSET,UNIT=SYSDA,SPACE=(TRK,(10,5)),        X
//             DISP=(NEW,PASS),DCB=(BLKSIZE=3200)
//SYSIN    DD  *
ACCOUNT(INITIATION)                EHSPL360 PL360                      DATE=69338     TIME=14.17.226
IEF236I ALLOC. FOR EHSPL360 PL360
IEF237I JOBLIB   ON 233
IEF237I SYSPRINT ON 563
IEF237I SYSGO    ON 437
IEF237I SYSIN    ON 585
IEF285I   PUB.EHS.PL360                                  PASSED
IEF285I   VOL SER NOS= PUB001.
IEF285I   SYS69338.T124933.RV000.EHSPL360.R0000827       DELETED
IEF285I   VOL SER NOS=        .
IEF285I   SYS69338.T124933.RV000.EHSPL360.LOADSET        PASSED
IEF285I   VOL SER NOS= SCF0V3.
IEF285I   SYS69338.T124933.RV000.EHSPL360.R0000828       DELETED
IEF285I   VOL SER NOS=        .
ACCOUNT(STEP) 107                  EHSPL360 PL360    RUN TIME=(    ,01.47) DATE=69338 END TIME=14.17.284 CODE=00000000 DEC
//GO     EXEC  PGM=MONITOR,PARM=200K,COND=(0,NE,PL360)
//SYSPRINT DD  SYSOUT=A,DCB=BLKSIZE=2660
//SYSPUNCH DD  SYSOUT=A,DCB=BLKSIZE=80
//SYSIN    DD  DSNAME=*.PL360.SYSGO,DISP=(OLD,DELETE)
//         DD  *
//
ACCOUNT(INITIATION)                EHSPL360 GO                         DATE=69338     TIME=14.17.296
IEF236I ALLOC. FOR EHSPL360 GO
IEF237I JOBLIB   ON 233
IEF237I SYSPRINT ON 563
IEF237I SYSPUNCH ON 56C
IEF237I SYSIN    ON 437
IEF237I          ON 58A
IEF285I   PUB.EHS.PL360                                  PASSED
IEF285I   VOL SER NOS= PUB001.
IEF285I   SYS69338.T124933.RV000.EHSPL360.R0000829       DELETED
IEF285I   VOL SER NOS=        .
IEF285I   SYS69338.T124933.RV000.EHSPL360.R0000830       DELETED
IEF285I   VOL SER NOS=        .
IEF285I   SYS69338.T124933.RV000.EHSPL360.LOADSET        DELETED
IEF285I   VOL SER NOS= SCF0V3.
IEF285I   SYS69338.T124933.RV000.EHSPL360.R0000831       DELETED
IEF285I   VOL SER NOS=        .
ACCOUNT(STEP) 107                  EHSPL360 GO       RUN TIME=(    ,01.20) DATE=69338 END TIME=14.17.341 CODE=000FA690 HEX
IEF285I   PUB.EHS.PL360                                  KEPT
IEF285I   VOL SER NOS= PUB001.
ACCOUNT(JOB)  107                  EHSPL360          RUN TIME=(    ,02.67) DATE=69338 END TIME=14.17.341 CODE=000FA690 HEX
```

```
01 0000  CC 0000  0001  BEGIN COMMENT SYNTAX-PROCESSOR;                                                    00000100
01 0012  CC C048  0002                                                                                    00000200
01 0012  00 0048  0003     COMMENT OPTION CARDS RECOGNIZED BY THE SYNTAX PROCESSOR                         00000300
01 0012  00 0048  0004           $SYMBOLS CAUSES THE SYMBOLS TO BE READ IN,                               00000400
01 0012  CC 0048  0005           $MATRIX CAUSES THE MATRIX TO BE PRINTED,                                 00000500
01 0012  CC 0048  0006           $FUNCTION CAUSES THE FUNCTIONS TO BE COMPUTED,                           00000600
01 0012  00 0048  0007           $CHECK CAUSES PROD TO BE CHECKED FOR IDENTICAL RIGHT PARTS,              00000700
01 0012  00 0048  0008           $LEFT CAUSES THE RELATIONS TO BE PRINTED FOR THE LEFT SYMBOLS,           00000800
01 0012  00 0048  0009           $RIGHT CAUSES THE RELATIONS TO BE PRINTED FOR THE RIGHT SYMB,            00000900
01 0012  00 0048  0010           $SYMPUNCH CAUSES THE SYMBOLS TO BE PUNCHED,                              00001000
01 0012  00 0048  0011           $TABLE CAUSES PL360 TABLES TO BE PUNCHED;                                00001050
01 0012  CC 0048  0012     SHORT INTEGER LSS =1, GTR =2, EQL =3;                                          00001100
01 0012  00 004E  0013     SHORT INTEGER SIX =6;                                                          00001200
01 0012  CC 0050  0014     ARRAY 4 BYTE  RELATION =(" <>=");                                              00001300
01 0012  CC 0054  0015     ARRAY 12 BYTE  ERRMESS =("****** LINE ");                                      00001400
01 0012  CC 0060  0016     ARRAY 32 BYTE  MESS1 =("NO PRECEDENCE VIOLATION DETECTED");                    00001500
01 0012  00 0080  0017     ARRAY 19 BYTE  SYMBMESS =("NONTERMINAL SYMBOLS");                              00001700
01 0012  CC C093  0018     ARRAY 25 BYTE  PATTERN =("LINE ",#20,#21,#20,                                  00001800
01 0012  00 C0AC  0019        " SAME AS LINE ",#20,#21,#20);                                              00001900
01 0012  00 C0AC  0020     LOGICAL ONES =#FFFFFFFF;                                                       00002000
01 0012  CC C0B0  0021     BYTE CHANGE=0, ERRFLAG=0;                                                      00002100
01 0012  CC 00B2  0022     BYTE EOPROD=0,SYMPUNCHFLAG=0;                                                  00002200
01 0012  00 00B4  0023     BYTE MATRIXFLAG=0, TABLEFLAG=0, FUNCTIONFLAG=0;                                00002300
01 0012  00 00B7  0024     BYTE SYMBOLFLAG=0, READFLAG=0, EOF=0;                                          00002400
01 0012  00 00BA  0025     BYTE RIGHTFLAG=0, LEFTFLAG=0, CHECKFLAG=0;                                     00002500
01 0012  00 00BD  0026     SHORT INTEGER P,V;                                                             00002600
01 0012  00 00C2  0027     SHORT INTEGER M,MM,M2,MM2,MM32,M64,MM12,M12;                                   00002700
01 0012  CC 00D2  0028     SHORT INTEGER N,N2,N12,NN;                                                     00002800
01 0012  00 00DA  0029     SHORT INTEGER MSTART=0, MCCUNT=51;                                             00002900
01 0012  00 00DE  0030     INTEGER U;                                                                     00003000
01 0012  CC 00E4  0031     ARRAY  8  INTEGER FMASK=(#40202120,                                            00003100
01 0012  00 0104  0032        "                 ",#40202120,#40404020,#21200000);                        00003200
01 0012  00 0104  0033     LONG REAL DEC;                                                                 00003300
01 0012  CC 0110  0034     INTEGER MASK3 SYN FMASK;                                                       00003400
01 0012  00 0110  0035     ARRAY 6 BYTE MASK4 = (#40,#20,#20,#21,#20,#20);                                00003500
01 0012  CC 0116  0036     INTEGER R1SAVE,R2SAVE,R3SAVE,R4SAVE;                                           00003600
01 0012  00 0128  0037     ARRAY  8  BYTE MASK =(#80,#40,#20,#10,#08,#04,#02,#01);                        00003700
01 0012  00 0130  0038     ARRAY  20  INTEGER CBUF;                                                       00003800
01 0012  00 0160  0039     ARRAY 132 BYTE UNDEFMESS=(" UNDEFINED SYMBOL:",114(" "));                      00003900
01 0012  00 0204  0040     ARRAY 132 BYTE BLANKS=132(" ");                                                00004000
01 0012  00 0288  0041     ARRAY 256 BYTE BUFFER=256(" ");                                                00004100
01 0012  00 C388  0042     ARRAY  80  BYTE PBUF SYN BUFFER;                                               00004200
01 0012  00 C388  0043     ARRAY  9  LOGICAL SAVE;                                                        00004300
01 0012  00 03AC  0044     ARRAY 256 SHORT INTEGER F, G;                                                  00004400
01 0012  00 07AC  0045     ARRAY  16  BYTE TRTABLE=("0123456789ABCDEF");                                  00004500
01 0012  00 C7BC  0046     ARRAY 3000 SHORT INTEGER PRTB;                                                 00004600
01 0012  00 1F2C  0047    SEGMENT BASE  R11;                                                              00004700
01 0016  C6 0000  0048     ARRAY  3072  BYTE SYTB;                                                        00004800
01 0016  06 0C00  0049     INTEGER LL;                                                                    00004900
01 0016  06 0C04  0050     ARRAY  256  SHORT INTEGER MTB;                                                 00005000
01 0016  06 0E04  0051     ARRAY 9068 INTEGER WORKSPACE;                                                  00005100
01 0016  06 9BB4  0052        ARRAY 2048 LOGICAL L SYN WORKSPACE, R SYN MEM(R12);                         00005200
01 0016  06 9BB4  0053        ARRAY 4097 LOGICAL H SYN MEM(R10);                                          00005300
01 0016  06 9BB4  0054        ARRAY 36000 BYTE PR SYN WORKSPACE;                                          00005400
01 0016  06 9BB4  0055        ARRAY 2048  BYTE PRTB1  SYN WORKSPACE;                                      00005500
01 0016  06 9BB4  0056                                                                                    00005600
01 0016  06 9BB4  0057  PROCEDURE ERROREXIT(R4); GOTO EXIT;                                               00005700
01 0020  06 9BB4  0058                                                                                    00005800
```

```
01 002C  06 9BB4  0059  SEGMENT PROCEDURE  READINPUT(R9);                                                    00005900
01 0020  06 9BB4  0060  BEGIN COMMENT INPUT AND CONSTRUCT PRODUCTION TABLES;                                 00006000
07 0000  06 9BB4  0061  PROCEDURE READCARD(R10);                                                            00006100
07 0000  06 9BB4  0062     BEGIN COMMENT READ NEXT CARD IMAGE INTO CARD BUFFER;                             00006200
07 0004  06 9BB4  0063        IF EOF THEN ERROREXIT;                                                        00006300
07 0018  06 9BB4  0064  L:    R0 := @CBUF; READ; IF ¬= THEN                                                 00006500
07 002A  06 9BB4  0065        BEGIN  SET(EOF);  SET(EOPROD);  END;                                          00006600
07 0032  06 9BB4  0066        CLI("$",CBUF); IF = THEN                                                      00006700
07 003A  06 9BB4  0067         BEGIN CLC(7,"SYMPUNCH",CBUF(1)); IF = THEN                                   00006800
07 0044  06 9BB4  0068           BEGIN SET(SYMPUNCHFLAG); IF ¬EOF THEN GOTO L;                              00006900
07 0050  06 9BB4  0069           END; CLC(5,"SYMBOL",CBUF(1)); IF = THEN                                    00007000
07 005A  06 9BB4  0070           BEGIN SET(SYMBOLFLAG); IF ¬EOF THEN GOTO L;                                00007100
07 0066  06 9BB4  0071           END; CLC(5,"MATRIX",CBUF(1)); IF = THEN                                    00007200
07 0070  06 9BB4  0072           BEGIN SET(MATRIXFLAG); IF ¬EOF THEN GOTO L;                                00007300
07 007C  06 9BB4  0073           END; CLC(4,"CHECK",CBUF(1)); IF = THEN                                     00007400
07 0086  06 9BB4  0074           BEGIN SET(CHECKFLAG); IF ¬EOF THEN GOTO L;                                 00007500
07 0092  06 9BB4  0075           END; CLC(4,"RIGHT",CBUF(1)); IF = THEN                                     00007600
07 009C  06 9BB4  0076           BEGIN SET(RIGHTFLAG); IF ¬EOF THEN GOTO L;                                 00007700
07 00A8  06 9BB4  0077           END; CLC(3,"LEFT",CBUF(1)); IF = THEN                                      00007800
07 00B2  06 9BB4  0078           BEGIN SET(LEFTFLAG); IF ¬EOF THEN GOTO L;                                  00007900
07 00BE  06 9BB4  0079           END; CLC(6,"FUNCTION",CBUF(1)); IF = THEN                                  00007920
07 00C8  06 9BB4  0080           BEGIN SET(FUNCTIONFLAG); IF ¬EOF THEN GOTO L;                              00007940
07 00D4  06 9BB4  0081           END; CLC(4,"TABLE",CBUF(1)); IF = THEN                                     00008000
07 00DE  06 9BB4  0082           BEGIN SET(TABLEFLAG); IF ¬EOF THEN GOTO L;                                 00008100
07 00EA  06 9BB4  0083           END; CLC(7,"CONTINUE",CBUF(1));  IF = THEN                                 00008200
07 00F4  06 9BB4  0084           BEGIN SET(EOPROD);                                                         00008300
07 00F8  06 9BB4  0085           END;                                                                       00008400
07 00F8  06 9BB4  0086        END;                                                                          00008500
07 00F8  06 9BB4  0087     END;                                                                             00008600
07 00FA  06 9BB4  0088                                                                                      00008700
07 00FA  06 9BB4  0089   PROCEDURE WRITEPRODUCTION(R10);                                                    00008800
07 00FA  06 9BB4  0090     BEGIN COMMENT R9 = PRODUCTION COUNT;                                             00008900
07 00FA  06 9BB4  0091        R9 := R9 + 1; IF R9 > 499 THEN ERROREXIT;                                     00009000
07 0112  06 9BB4  0092        MVC(3,BUFFER(7),FMASK); MVC(109,BUFFER(16),BLANKS);                           00009100
07 011E  06 9BB4  0093        CVD(R9,DEC); ED(3,BUFFER(7),DEC(6));                                          00009200
07 0128  06 9BB4  0094        CLC(11,CBUF,BLANKS); IF ¬= THEN MVC(2,BUFFER(29),"::=");                      00009300
07 0138  06 9BB4  0095        R7 := @CBUF(12); R8 := @BUFFER(33); MVC(11,BUFFER(16),CBUF);                  00009400
07 0146  06 9BB4  0096        FOR R0 := 0 STEP 1 UNTIL 4 DO                                                 00009500
07 014A  06 9BB4  0097        BEGIN CLC(11,B7,B8); IF = THEN GOTO B;                                        00009600
07 0158  06 9BB4  0098           MVC(11,B8,B7); R7 := R7 + 12; R8 := R8 + 11;                               00009700
07 0166  06 9BB4  0099           CLI(" ",B8); WHILE = DO                                                    00009800
07 016E  06 9BB4  0100           BEGIN R8 := R8 - 1; CLI(" ",B8);                                           00009900
07 0176  06 9BB4  0101           END; R8 := R8 + 3;                                                         00010000
07 017E  06 9BB4  0102        END;                                                                          00010100
07 018A  06 9BB4  0103   B:   R0 := @BUFFER; WRITE;                                                        00010200
07 0198  06 9BB4  0104     END;                                                                             00010300
07 019A  06 9BB4  0105                                                                                      00010400
07 019A  06 9BB4  0106   SAVE := R9; READCARD; TEST(SYMBOLFLAG); IF = THEN                                  00010500
07 01AA  06 9BB4  01C7   BEGIN COMMENT READ SYMBOLS AND PRODUCTIONS;                                        00010600
07 01AA  C6 9BB4  0108     R3 := @SYTB; R2 := 0;                                                          00010700
07 01B2  06 9BB4  0109     COMMENT READ NONTERMINAL SYMBOLS;                                              00010800
07 01B2  06 9BB4  0110   LN: R3 := R3+12; R2 := R2+1; IF R2 > 255 THEN ERROREXIT;                          00010900
07 01CE  06 9BB4  0111     MVC(11,B3,CBUF); READCARD;                                                     00011000
07 01D8  06 9BB4  0112     CLC(1,"$$",CBUF); IF ¬= THEN GOTO LN;                                          00011100
07 01E2  06 9BB4  0113     MM := R2; R2 := R2 SHLL 1; MM2 := R2; R2 := R2 SHLL 4; MM32 := R2;             00011200
07 01F6  06 9BB4  0114     R2 := MM;                                                                      00011300
07 01FA  06 9BB4  0115     COMMENT READ TERMINAL SYMBOLS;                                                 00011400
07 01FA  06 9BB4  0116   LT: READCARD; CLC(1,"$$",CBUF);                                                   00011500
```

```
07 0204  C6 9BB4   0117      IF ¬= THEN                                                                    00011600
07 0208  06 9BB4   0118      BEGIN R3 := R3+12; R2 := R2+1; IF R2 > 255 THEN ERROREXIT;                    00011700
07 0224  06 9BB4   0119         MVC(11,B3,CBUF); GOTO LT;                                                  00011800
07 022E  06 9BB4   0120      END ;                                                                         00011900
07 022E  06 9BB4   0121      M := R2; R2 := R2 SHLL 1; M2 := R2; R2 := R2 SHLL 5; M64 := R2;               00012000
07 0242  06 9BB4   0122      R2 := M2 * 6S; M12 := R2;                                                     00012100
07 024E  06 9BB4   0123      COMMENT READ PRODUCTIONS;                                                     00012200
07 024E  06 9BB4   0124      MVC(131,BUFFER,BLANKS);                                                       00012300
07 0254  C6 9BB4   0125      R9 := 0; R5 := 12;                                                            00012400
07 025C  06 9BB4   0126   LP: READCARD; TEST(EOPROD); IF ¬= THEN                                          00012500
07 0268  06 9BB4   0127      BEGIN WRITEPRODUCTION; R3 := @CBUF;                                           00012600
07 027C  06 9BB4   0128         FOR R6 := 0 STEP 1 UNTIL 5 DO                                              00012700
07 0274  06 9BB4   0129         BEGIN R4 := @SYTB;                                                         00012800
07 027C  06 9BB4   0130            FOR R1 := 0 STEP 2 UNTIL M2 DO                                          00012900
07 0280  06 9BB4   0131            BEGIN CLC(11,B3,B4); IF = THEN GOTO LF ELSE R4 := R4+12;                00013000
07 0292  06 9BB4   0132            END ;                                                                   00013100
07 029E  06 9BB4   0133            MVC(11,UNDEFMESS(20),B3); R0 := @UNDEFMESS;                             00013200
07 02A8  06 9BB4   0134            WRITE; MVI(1,ERRFLAG);                                                  00013300
07 02B6  06 9BB4   0135         LF: PRTB(R5) := R1; R5 := R5+2; R3 := R3+12;                               00013400
07 02C2  C6 9BB4   0136         END ;                                                                      00013500
07 02CE  06 9BB4   0137         GOTO LP;                                                                   00013600
07 02D2  C6 9BB4   0138      END ;                                                                         00013700
07 02D2  06 9BB4   0139      N := R9; R2 := R9 SHLL 1; N2 := R2; R2 := R2 * SIX; N12 := R2;                00013800
07 02E8  06 9BB4   0140    END ELSE                                                                        00013900
07 02E8  06 9BB4   0141    BEGIN COMMENT READ PRODUCTIONS AND CONSTRUCT SYMBOL TABLE;                      00014000
07 02EC  06 9BB4   0142      MVC(71,PR(72),BUFFER); MVC(15,BUFFER,BLANKS);                                 00014100
07 02F8  06 9BB4   0143      MVC(71,BUFFER(16),PR(72));                                                    00014200
07 02FE  06 9BB4   0144      R9 := 0; R4 := @PR;                                                           00014300
07 0306  06 9BB4   0145   L1: WRITEPRODUCTION; R4 := R4 + 72; MVC(71,B4,CBUF);                           00014400
07 0314  06 9BB4   0146      READCARD; TEST(EOPROD); IF ¬= THEN GOTO L1;                                   00014500
07 0320  06 9BB4   0147      N := R9; R3 := R9 SHLL 1; N2 := R3; R3 := R3 * SIX; N12 := R3;                00014600
07 0336  06 9BB4   0148      R3 := R3 * SIX; R1 := @PR(R3); R2 := @PR(72); R3 := 0;                        00014700
07 0346  06 9BB4   0149      FOR R2 := R2 STEP 72 UNTIL R1 DO                                              00014800
07 0346  06 9BB4   0150      BEGIN FOR R5 := 0 STEP 12 UNTIL R3 DO                                         00014900
07 034E  06 9BB4   0151         BEGIN R4 := @SYTB(R5); CLC(11,B2,B4); IF = THEN GOTO F;                    00015000
07 036C  06 9BB4   0152         END;                                                                       00015100
07 036A  06 9BB4   0153         R3 := R3+12; R4 := @SYTB(R3); MVC(11,B4,B2);                               00015200
07 0378  06 9BB4   0154         IF R3 > 3072 THEN ERROREXIT;                                               00015300
07 038C  06 9BB4   0155      F: END;                                                                       00015400
07 0396  06 9BB4   0156      MM12 := R3; R2 := N12+10; N12 := R2; R2 := @PR(72);                           00015500
07 03AA  06 9BB4   0157      FOR R1 := 12 STEP 2 UNTIL N12 DO                                              00015600
07 03AE  06 9BB4   0158      BEGIN FOR R5 := 0 STEP 12 UNTIL R3 DO                                         00015700
07 03B6  06 9BB4   0159         BEGIN R4 := @SYTB(R5); CLC(11,B2,B4); IF = THEN GOTO F;                    00015800
07 03C8  06 9BB4   0160         END;                                                                       00015900
07 03D2  06 9BB4   0161         R3 := R3+12; R5 := R3; R4 := @SYTB(R3); MVC(11,B4,B2);                     00016000
07 03E2  06 9BB4   0162         IF R3 > 3072 THEN ERROREXIT;                                               00016100
07 03F6  06 9BB4   0163      F:    R4 := 0; R5 := R5/6;                                                    00016200
07 03FE  06 9BB4   0164         PRTB(R1) := R5; R2 := R2+12;                                               00016300
07 0406  06 9BB4   0165      END;                                                                          00016400
07 0412  06 9BB4   0166      M12 := R3;                                                                    00016500
07 0416  06 9BB4   0167      R0 := 0; R1 := MM12/12; MM := R1; R1 := R1 SHLL 1;                            00016600
07 042A  06 9BB4   0168      MM2 := R1; R1 := R1 SHLL 4; MM32 := R1;                                       00016700
07 0436  06 9BB4   0169      R1 := M12/12; M := R1; R1 := R1 SHLL 1;                                       00016800
07 0446  06 9BB4   0170      M2 := R1; R1 := R1 SHLL 5; M64 := R1;                                         00016900
07 0452  06 9BB4   0171      MVI(0,PR); R1 := @PR; R2 := R1 + 36000;                                       00017000
07 0460  06 9BB4   0172      FOR R1 := R1 STEP 256 UNTIL R2 DO MVC(255,B1(1),B1);                          00017100
07 0474  06 9BB4   0173   END;                                                                            00017200
07 0474  06 9BB4   0174      PAGE;                                                                         00017300
```

```
07 047E  C6 9BB4  0175      FOR R1 := 12 STEP 12 UNTIL N12 DO                                                00017400
07 0482  06 9BB4  0176         BEGIN R2 := PRTB(R1); IF R2 = 0 THEN                                          00017500
07 0490  06 9BB4  0177            BEGIN R2 := R1-12; R3 := PRTB(R2); PRTB(R1) := R3;                         00017600
07 049E  06 9BB4  0178            END ;                                                                      00017700
07 049E  06 9BB4  0179         END ;                                                                         00017800
07 04AA  06 9BB4  0180      COMMENT PRINT THE SYMBOLTABLE (IN TWO PARTS) ;                                   00017900
07 04AA  06 9BB4  0181      R4 := 1; R1 := @SYTB(12);                                                        00018000
07 04B2  06 9BB4  0182      MVC(131,BUFFER,BLANKS); MVC(18,BUFFER(8),SYMBMESS);                              00018100
07 04BE  06 9BB4  0183      R0 := @BUFFER; WRITE; MVC(18,BUFFER(8),BLANKS); WRITE;                           00018200
07 04DC  06 9BB4  0184   L2: FOR R2 := 7 STEP 24 UNTIL 103 DO                                                00018300
07 04EC  06 9BB4  0185      BEGIN IF R4 > MM THEN GOTO L3;                                                   00018400
07 04EC  06 9BB4  0186         CVD(R4,DEC); R3 := @BUFFER(R2); MVC(3,B3,FMASK);                              00018500
07 04FA  06 9BB4  0187         ED(3,B3,DEC(6)); MVC(11,B3(7),B1);                                            00018600
07 0506  06 9BB4  0188         R1 := R1 + 12; R4 := R4 + 1;                                                  00018700
07 050E  06 9BB4  0189      END;                                                                             00018800
07 051A  06 9BB4  0190      WRITE; MVC(131,BUFFER,BLANKS); GOTO L2;                                          00018900
07 052E  06 9BB4  0191                                                                                       00019000
07 052E  06 9BB4  0192   L3: WRITE; MVC(131,BUFFER,BLANKS); WRITE;                                           00019100
07 0548  06 9BB4  0193      MVC(15,BUFFER(8),SYMBMESS(3)); WRITE;                                            00019200
07 0558  06 9BB4  0194      MVC(15,BUFFER(8),BLANKS); WRITE;                                                 00019300
07 0568  06 9BB4  0195   L4: FOR R2 := 7 STEP 24 UNTIL 103 DO                                                00019400
07 056C  06 9BB4  0196      BEGIN IF R4 > M THEN GOTO L5;                                                    00019500
07 0578  06 9BB4  0197         CVD(R4,DEC); R3 := @BUFFER(R2); MVC(3,B3,FMASK);                              00019600
07 0586  06 9BB4  0198         ED(3,B3,DEC(6)); MVC(11,B3(7),B1);                                            00019700
07 0592  06 9BB4  0199         R1 := R1 + 12; R4 := R4 + 1;                                                  00019800
07 059A  06 9BB4  0200      END;                                                                             00019900
07 05A6  06 9BB4  0201      WRITE; MVC(131,BUFFER,BLANKS); GOTO L4;                                          00020000
07 05BA  06 9BB4  0202                                                                                       00020100
07 05BA  06 9BB4  0203   L5: WRITE; R9 := SAVE;                                                              00020200
07 05C8  06 9BB4  0204                                                                                       00020300
07 05C8  06 9BB4  0205    IF CHECKFLAG THEN                                                                  00020400
07 05D0  06 9BB4  0206    BEGIN MVC(131,BUFFER,BLANKS); R0 := @BUFFER; WRITE;                                00020500
07 05E4  06 9BB4  0207       R1 := @PRTB; R6 := R1 + N12;                                                    00020600
07 05EE  06 9BB4  0208       FOR R2 := R1 + 12 STEP 12 UNTIL R6 DO                                           00020700
07 05F4  06 9BB4  0209       BEGIN FOR R3 := R2 + 12 STEP 12 UNTIL R6 DO                                     00020800
07 05FE  06 9BB4  0210          BEGIN CLC(9,B2(2),B3(2)); IF = THEN                                          00020900
07 060C  06 9BB4  0211             BEGIN IF CHECKFLAG THEN                                                   00021000
07 0614  06 9BB4  0212                BEGIN MVC(31,BUFFER,"IDENTICAL RIGHT PART OCCURRENCES");               00021100
07 061A  06 9BB4  0213                   WRITE; RESET(CHECKFLAG); MVC(31,BUFFER,BLANKS);                     00021200
07 062E  06 9BB4  0214                END;                                                                   00021300
07 062E  06 9BB4  0215                R4 := 0; R5 := R2 - R1 / 12; CVD(R5,DEC);                              00021400
07 063E  06 9BB4  0216                MVC(24,BUFFER(3),PATTERN); ED(3,BUFFER(7),DEC(6));                     00021500
07 064A  06 9BB4  0217                R4 := 0; R5 := R3 - R1 / 12; CVD(R5,DEC);                              00021600
07 065A  06 9BB4  0218                ED(3,BUFFER(24),DEC(6)); WRITE; GOTO L;                                00021700
07 066E  06 9BB4  0219             END;                                                                      00021800
07 066E  06 9BB4  0220          END;                                                                         00021900
07 0678  06 9BB4  0221    L: END;                                                                            00022000
07 0682  06 9BB4  0222       IF CHECKFLAG THEN                                                               00022100
07 068A  06 9BB4  0223       BEGIN MVC(23,BUFFER,"NO IDENTICAL RIGHT PARTS");                                00022200
07 0690  06 9BB4  0224          R0 := @BUFFER; WRITE;                                                        00022300
07 069E  06 9BB4  0225       END;                                                                            00022400
07 069E  06 9BB4  0226    END;                                                                               00022500
07 069E  06 9BB4  0227                                                                                       00022600
07 069E  06 9BB4  0228    IF SYMPUNCHFLAG THEN                                                               00022700
07 06A6  06 9BB4  0229    BEGIN MVC(79,CBUF,BLANKS); MVC(7,CBUF,"$SYMBOLS");                                 00022800
07 06B2  06 9BB4  0230       R0 := @CBUF; PUNCH; R2 := @SYTB(12);                                            00022900
07 06C4  06 9BB4  0231       FOR R1 := 2 STEP 2 UNTIL M2 DO                                                  00023000
07 06C8  06 9BB4  0232       BEGIN MVC(11,CBUF,B2); PUNCH; R2 := R2 + 12;                                    00023100
```

```
07 06E0  06 9BB4   0233          IF R1 = MM2 OR R1 = M2 THEN                                  00023200
07 06F0  06 9BB4   0234          BEGIN MVC(11,CBUF,"$$          "); PUNCH;                    00023300
07 070C  06 9BB4   0235          END;                                                        00023400
07 070C  06 9BB4   0236       END;                                                           00023500
07 070C  06 9BB4   0237    END;                                                              00023600
07 070C  06 9BB4   0238                                                                      00023700
07 070C  06 9BB4   0239    END ;  COMMENT  END OF READINPUT;                                 00023800

                  SEGMENT 07  NAME = SEG#07      LENGTH = 07F8  BASE REG = 15

01 0020  06 9BB4   0240   COMMENT  * * * * * *  E X E C U T I O N * * * * *;                00023900
01 002C  C6 9BB4   0241                                                                      00024000
01 0020  06 9BB4   0242       R1 := 0;                                                       00024100
01 0024  06 9BB4   0243   LOOP:                                                             00024200
01 0024  06 9BB4   0244       R12 := @L + 8192;                                              00024300
01 002C  06 9BB4   0245       FOR R2 := 0 STEP 4 UNTIL 36268 DO  WORKSPACE(R2) := R1;        00024400
01 0044  06 9BB4   0246       MVC(11,SYTB,BLANKS); READINPUT;                                00024500
01 0054  06 9BB4   0247       CLI(0,ERRFLAG); IF > THEN GOTO EXIT;                           00024600
01 005C  06 9BB4   0248       PAGE;                                                          00024700
01 0066  06 9BB4   0249                                                                      00024800
01 0066  06 9BB4   0250    BEGIN COMMENT BUILD TABLES OF LEFT- AND RIGHT OCCURRENCES;        00024900
01 0066  C6 9BB4   0251        SEGMENT BASE  R10;                                           00025000
01 006A  C8 0000   0252       ARRAY  256  SHORT INTEGER C1,C2,BL,BR,SO,SL,SR;                00025100
01 006A  C8 0E00   0253                                                                      00025200
01 006A  08 0E00   0254       R0 := 0;                                                       00025300
01 006E  08 0E00   0255       FOR R1 := 0 STEP 2 UNTIL M2 DO                                 00025400
01 0072  C8 0E00   0256       BEGIN C1(R1) := R0;  C2(R1) := R0;                             00025500
01 007E  08 0E00   0257       END;                                                           00025600
01 008A  C8 0E00   0258       FOR R1 := 2 STEP 2 UNTIL N2 DO                                 00025700
01 008E  08 0E00   0259       BEGIN R2 := R1 * SIX; R0 := PRTB(R2); SO(R1) := R0;            00025800
01 00A0  08 0E00   0260          R2 := R2+2; R0 := PRTB(R2); SL(R1) := R0;                   00025900
01 00AC  08 0E00   0261          R2 := R2+8; R0 := PRTB(R2);                                 00026000
01 00B4  08 0E00   0262          WHILE R0 = 0 DO                                             00026100
01 00BA  C8 0E00   0263          BEGIN R2 := R2-2; R0 := PRTB(R2);                           00026200
01 00C2  C8 0E00   0264          END ;                                                       00026300
01 00C6  08 0E00   0265          SR(R1) := R0; R0 := 1; BL(R1) := R0; BR(R1) := R0;          00026400
01 00D6  08 0E00   0266          R2 := SO(R1); R0 := C1(R2)+1; C1(R2) := R0; C2(R2) := R0;   00026500
01 00EA  C8 0E00   0267       END ;                                                          00026600
01 00F6  08 0E00   0268                                                                      00026700
01 00F6  08 CE00   0269       R0 := N; NN := R0;                                             00026800
01 00FE  08 0E00   0270   L1: MVI(0,CHANGE);                                                00026900
01 0102  08 0E00   0271       FOR R1 := 2 STEP 2 UNTIL N2 DO                                 00027000
01 0106  C8 0E00   0272       BEGIN R2 := BL(R1); IF R2 > 0 THEN                             00027100
01 0114  08 0E00   0273          BEGIN R2 := SO(R1); R3 := SL(R1);                           00027200
01 011C  08 0E00   0274             R2 := R2 SHLL 4;      COMMENT R2 = "A" ROW INDEX;       00027300
01 012C  C8 0E00   0275             R4 := R3; SRDL(R4,4); R4 := R4+R2;                      00027400
01 0128  08 0E00   0276             R6 := R6-R6; IC(R6,L(R4));                              00027500
01 012E  08 0E00   0277             R5 := R5 SHRL 29;                                       00027600
01 0132  C8 CE00   0278             R7 := R7-R7; IC(R7,MASK(R5)); R0 := R6 AND R7;          00027700
01 013C  08 0E00   0279             IF  = THEN BEGIN R6 := R6 OR R7; STC(R6,L(R4));         00027800
01 0146  08 0E00   0280                              MVI(1,CHANGE);                          00027900
01 014A  08 0E00   0281                         END ;                                        00028000
01 014A  08 0E00   0282             IF R3 <= MM2 THEN                                       00028100
01 0152  C8 0E00   0283             BEGIN R5 := R3 SHLL 4; COMMENT R5 = "B" ROW INDEX;      00028200
01 0158  C8 0E00   0284                R6 := M SHRL 3 + R2;                                 00028300
01 0162  C8 0E00   0285                FOR R4 := R2 STEP 4 UNTIL R6 DO                      00028400
01 0164  C8 CE00   0286                BEGIN R0 := L(R4) XOR ONES AND L(R5);                00028500
01 0174  08 0E00   0287                   IF ¬= THEN                                        00028600
```

```
01 0178  C8 0E00  0288                BEGIN R0 := R0 OR L(R4); L(R4) := R0; MVI(1,CHANGE);     00028700
01 0184  08 0E00  0289                END ;                                                    00028800
01 0184  C8 0E00  0290                R5 := R5+4;                                              00028900
01 0188  C8 0E00  0291             END ;                                                       00029000
01 0192  08 0E00  0292          END ;                                                          00029100
01 0192  08 0E00  0293          R2 := R2 SHRL 4; R4 := C1(R3);                                 00029200
01 019A  C8 0E00  0294          IF R4 = 0 THEN                                                 00029300
01 01A0  08 0E00  0295             BEGIN BL(R1) := R4; R0 := C1(R2)-1; C1(R2) := R0;           00029400
01 01B0  C8 0E00  0296                R0 := NN-1; NN := R0;                                    00029500
01 01BC  C8 0E00  0297             END ;                                                       00029600
01 01BC  08 0E00  0298       END ;                                                            00029700
01 01BC  C8 0E00  0299    END ;                                                               00029800
01 01C8  C8 0E00  0300    CLI(1,CHANGE);                                                      00029900
01 01CC  08 0E00  0301    IF = THEN                                                           00030000
01 01D0  C8 0E00  0302    BEGIN R0 := NN; IF R0 > 0 THEN GOTO L1;                             00030100
01 01DA  08 0E00  0303    END ;                                                               00030200
01 01DA  C8 0E00  0304                                                                        00030300
01 01DA  08 0E00  0305    R0 := N; NN := R0;                                                  00030400
01 01E2  C8 0E00  0306 L2: MVI(0,CHANGE);                                                     00030500
01 01E6  08 0E00  0307    FOR R1 := 2 STEP 2 UNTIL N2 DO                                      00030600
01 01EA  C8 0E00  0308    BEGIN R2 := BR(R1); IF R2 > 0 THEN                                  00030700
01 01F8  08 0E00  0309       BEGIN R2 := S0(R1); R3 := SR(R1);                                00030800
01 0200  C8 0E00  0310          R2 := R2 SHLL 4;     COMMENT R2 = "A" ROW INDEX;             00030900
01 0204  08 0E00  0311          R4 := R3; SRDL(R4,4); R4 := R4+R2;                           00031000
01 020C  C8 0E00  0312          R6 := R6-R6; IC(R6,R(R4));                                   00031100
01 0212  C8 0E00  0313          R5 := R5 SHRL 29;                                            00031200
01 0216  C8 0E00  0314          R7 := R7-R7; IC(R7,MASK(R5)); R0 := R6 AND R7;               00031300
01 0220  08 0E00  0315          IF  = THEN BEGIN R6 := R6 OR R7; STC(R6,R(R4));              00031400
01 022A  C8 CE00  0316                      MVI(1,CHANGE);                                   00031500
01 022E  08 0E00  0317                   END ;                                               00031600
01 022E  C8 0E00  0318          IF R3 <= MM2 THEN                                            00031700
01 0236  08 0E00  0319          BEGIN R5 := R3 SHLL 4; COMMENT R5 = "B" ROW INDEX;           00031800
01 023C  C8 0E00  0320             R6 := M SHRL 3 + R2;                                      00031900
01 0246  08 0E00  0321             FOR R4 := R2 STEP 4 UNTIL R6 DO                           00032000
01 0248  C8 0E00  0322             BEGIN R0 := R(R4) XOR ONES AND R(R5);                     00032100
01 0258  08 0E00  0323                IF ¬= THEN                                             00032200
01 025C  C8 0E00  0324                BEGIN R0 := R0 OR R(R4); R(R4) := R0; MVI(1,CHANGE);   00032300
01 0268  08 0E00  0325                END ;                                                  00032400
01 0268  C8 0E00  0326                R5 := R5+4;                                            00032500
01 026C  08 0E00  0327             END ;                                                     00032600
01 0276  08 0E00  0328          END ;                                                        00032700
01 0276  08 0E00  0329          R2 := R2 SHRL 4; R4 := C2(R3);                               00032800
01 027E  08 0E00  0330          IF R4 = 0 THEN                                               00032900
01 0284  08 0E00  0331             BEGIN BR(R1) := R4; R0 := C2(R2)-1; C2(R2) := R0;         00033000
01 0294  C8 0E00  0332                R0 := NN-1; NN := R0;                                  00033100
01 02AC  C8 0E00  0333             END ;                                                     00033200
01 02A0  08 0E00  0334       END ;                                                          00033300
01 02AC  C8 0E00  0335    END ;                                                             00033400
01 02AC  C8 0E00  0336    CLI(1,CHANGE);                                                    00033500
01 02B0  08 0E00  0337    IF = THEN                                                         00033600
01 02B4  08 0E00  0338    BEGIN R0 := NN; IF R0 > 0 THEN GOTO L2;                           00033700
01 02BE  08 0E00  0339    END ;                                                             00033800
01 02BE  08 0E00  0340                                                                      00033900
01 02BE  C8 0E00  0341 END ;  COMMENT  END OF LR OCCURRENCES;                               00034000

                  SEGMENT C8  NAME = SEG#08      LENGTH = 0E00  BASE REG = 10

01 02BE  06 9BB4  0342                                                                      00034100
```

```
01 02BE  C6 9BB4  0343  R10 := R12 + 8192;                                                                    00034200
01 02C4  C6 9BB4  0344                                                                                       00034300
01 02C4  06 9BB4  0345   BEGIN COMMENT BUILD PRECEDENCE MATRIX;                                              00034400
01 02C4  06 9BB4  0346   PROCEDURE ENTER (R9);                                                               00034500
01 02C4  06 9BB4  0347      COMMENT ENTERS THE RELATION R0 AT COORDINATES R1,R2;                              00034600
01 02C4  C6 9BB4  0348      BEGIN STM(R0,R5,SAVE);                                                           00034700
01 02CC  06 9BB4  0349         R1 := R1 SHLL 5; SRDL(R2,5);                                                  00034800
01 02D4  06 9BB4  0350         R2 := R2 SHLL 2 + R1;                                                         00034900
01 02DA  06 9BB4  0351         R3 := R3 SHRL 28 SHLL 1;                                                      00035000
01 02E2  06 9BB4  0352         R5 := H(R2); SLDL(R4,B3); R6 := R4; R4 := 0;                                  00035100
01 02F0  06 9BB4  0353         SLDL(R4,2); COMMENT BITPAIR IS NOW IN R4;                                     00035200
01 02F4  06 9BB4  0354         IF R4 ¬= R0 THEN                                                              00035300
01 02FA  C6 9BB4  0355         IF R4 = 0 THEN                                                                00035400
01 0300  C6 9BB4  0356         BEGIN R4 := R0; SRDL(R4,2); R4 := R6; SRDL(R4,B3); H(R2) := R5;               00035500
01 0310  06 9BB4  0357         END ELSE                                                                      00035600
01 0310  C6 9BB4  0358         BEGIN COMMENT PRECEDENCE CONFLICT;                                            00035700
01 0314  06 9BB4  0359            MVC(131,BUFFER,BLANKS);                                                    00035800
01 031A  06 9BB4  0360            MVC(11,BUFFER,ERRMESS); SET(ERRFLAG);                                      00035900
01 0324  C6 9BB4  0361            R3 := R0; R0 := 0; R1 := R8/12;                                            00036000
01 0330  06 9BB4  0362            MVC(3,BUFFER(11),FMASK); CVD(R1,DEC);                                      00036100
01 033A  06 9BB4  0363            ED(3,BUFFER(11),DEC(6));                                                   00036200
01 0340  06 9BB4  0364            R1 := SAVE(4)*SIX; R2 := @SYTB(R1); MVC(11,BUFFER(20),B2);                 00036300
01 0352  C6 9BB4  0365            R2 := @RELATION(R4); MVC(0,BUFFER(33),B2);                                 00036400
01 035C  C6 9BB4  0366            R2 := @RELATION(R3); MVC(0,BUFFER(34),B2);                                 00036500
01 0366  C6 9BB4  0367            R1 := SAVE(8)*SIX; R2 := @SYTB(R1); MVC(11,BUFFER(36),B2);                 00036600
01 0378  06 9BB4  0368            R0 := @BUFFER; WRITE;                                                      00036700
01 0386  06 9BB4  0369         END ;                                                                         00036800
01 0386  06 9BB4  0370         LM(R0,R5,SAVE);                                                               00036900
01 038A  06 9BB4  0371   X: END;                                                                             00037000
01 038C  06 9BB4  0372                                                                                       00037100
01 038C  06 9BB4  0373      COMMENT BUILD PRECEDENCE MATRIX;                                                 00037200
01 038C  06 9BB4  0374      R5 := 0;                                                                         00037300
01 0390  06 9BB4  0375      FOR R8 := 12 STEP 12 UNTIL N12 DO                                                00037400
01 0394  06 9BB4  0376      BEGIN R4 := R8+12;                                                               00037500
01 039E  06 9BB4  0377      L1:   R4 := R4-2; R6 := PRTB(R4); IF R6 = 0 THEN GOTO L1;                        00037600
01 03AC  06 9BB4  0378      L2:   R3 := R4-2; V := R3;                                                       00037700
01 03B6  06 9BB4  0379         IF R3 > R8 THEN                                                               00037800
01 03BC  06 9BB4  0380         BEGIN R1 := PRTB(R3); R2 := PRTB(R4); P := R1; COMMENT Q = R2;                00037900
01 03C8  06 9BB4  0381            R0 := EQL; ENTER;                                                          00038000
01 03D0  06 9BB4  0382            IF R1 <= MM2 THEN                                                          00038100
01 03D8  06 9BB4  0383            BEGIN R3 := R1 SHLL 8; R0 := GTR;     COMMENT R3 = P ROW INDEX;            00038200
01 03E2  C6 9BB4  0384               FOR R1 := 2 STEP 2 UNTIL M2 DO                                          00038300
01 03E6  06 9BB4  0385               BEGIN R6 := R1+R3; SRDL(R6,4); R7 := R7 SHRL 29;                        00038400
01 03F6  06 9BB4  0386                  IC(R5,R(R6)); IC(R6,MASK(R7)); R6 := R6 AND R5;                      00038500
01 0400  C6 9BB4  0387                  IF ¬= THEN ENTER;                                                    00038600
01 0408  06 9BB4  0388               END ;                                                                   00038700
01 0414  06 9BB4  0389               IF R2 <= MM2 THEN                                                       00038800
01 041C  06 9BB4  0390               BEGIN R4 := R2 SHLL 8;             COMMENT R4 = Q ROW INDEX;            00038900
01 0422  C6 9BB4  0391                  FOR R2 := 2 STEP 2 UNTIL M2 DO                                       00039000
01 0426  06 9BB4  0392                  BEGIN R6 := R2+R4; SRDL(R6,4); R7 := R7 SHRL 29;                     00039100
01 0436  06 9BB4  0393                     R1 := P;                                                          00039200
01 043A  06 9BB4  0394                     IC(R5,L(R6)); IC(R6,MASK(R7)); R6 := R6 AND R5;                   00039300
01 0444  06 9BB4  0395                     IF ¬= THEN                                                        00039400
01 0448  06 9BB4  0396                     BEGIN R0 := LSS; ENTER; R0 := GTR;                                00039500
01 0454  06 9BB4  0397                        FOR R1 := 2 STEP 2 UNTIL M2 DO                                 00039600
01 0458  06 9BB4  0398                        BEGIN R6 := R1+R3; SRDL(R6,4); R7:=R7 SHRL 29;                 00039700
01 0468  06 9BB4  0399                           IC(R5,R(R6)); IC(R6,MASK(R7)); R6:=R6 AND R5;               00039800
01 0472  06 9BB4  0400                           IF ¬= THEN ENTER;                                           00039900
```

```
01 047A  C6 9BB4  0401                        END ;                                                                00040000
01 0486  06 9BB4  0402                     END ;                                                                   00040100
01 0486  06 9BB4  0403                  END ;                                                                      00040200
01 0492  C6 9BB4  0404               END ;                                                                         00040300
01 0492  06 9BB4  0405            END ELSE                                                                         00040400
01 0492  06 9BB4  0406            IF R2 <= MM2 THEN                                                                00040500
01 049E  06 9BB4  0407            BEGIN R4 := R2 SHLL 8; R0 := LSS;     COMMENT R4 = Q ROW INDEX;                  00040600
01 04A8  C6 9BB4  0408               FOR R2 := 2 STEP 2 UNTIL M2 DO                                                00040700
01 04AC  06 9BB4  0409               BEGIN R6 := R2+R4; SRDL(R6,4); R7 := R7 SHRL 29;                              00040800
01 04BC  06 9BB4  0410                  IC(R5,L(R6)); IC(R6,MASK(R7)); R6 := R6 AND R5;                            00040900
01 04C6  C6 9BB4  0411                  IF ¬= THEN ENTER;                                                          00041000
01 04CE  06 9BB4  0412               END ;                                                                         00041100
01 04DA  06 9BB4  0413            END ;                                                                            00041200
01 04DA  06 9BB4  0414            R4 := V; GOTO L2;                                                                00041300
01 04E2  C6 9BB4  0415         END ;                                                                               00041400
01 04E2  06 9BB4  0416      END ;                                                                                  00041500
01 04EE  06 9BB4  0417      TEST(ERRFLAG); IF = THEN GOTO EXIT;                                                    00041600
01 04F6  06 9BB4  0418      MVC(131,BUFFER,BLANKS); MVC(31,BUFFER,MESS1);                                          00041700
01 0502  06 9BB4  0419      R0 := @BUFFER; WRITE;                                                                  00041800
01 0510  C6 9BB4  0420   END ;                                                                                     00041900
01 0510  C6 9BB4  0421                                                                                             00042000
01 0510  06 9BB4  0422   IF MATRIXFLAG THEN                                                                        00042100
01 0518  06 9BB4  0423   BEGIN COMMENT  PRINT PRECEDENCE MATRIX;                                                    00042200
01 0518  06 9BB4  0424                                                                                             00042250
01 0518  06 9BB4  0425   PROCEDURE HEADING (R9);                                                                   00042300
01 0518  06 9BB4  0426   BEGIN  R4 := MSTART * 12S;  R4 := @SYTB(R4);  R6 := MCOUNT SHLL 1;                       00042400
01 0530  06 9BB4  0427     R5 := 0;  R0 := @BUFFER;  MVC(131,BUFFER,BLANKS);  WRITE;                               00042500
01 0548  06 9BB4  0428     FOR R1 := 0 STEP 1 UNTIL 11 DO                                                          00042600
01 054C  06 9BB4  0429     BEGIN  R2 := R4 + R1;                                                                   00042700
01 0554  C6 9BB4  0430       FOR R3 := 0 STEP 2 UNTIL R6  DO                                                       00042800
01 0558  06 9BB4  0431       BEGIN  IC(R5,B2);  STC(R5,BUFFER(R3+16));  R2 := R2 + 12;                             00042900
01 0568  C6 9BB4  0432       END;                                                                                  00043000
01 0572  06 9BB4  0433       WRITE;                                                                                00043100
01 057C  06 9BB4  0434     END;                                                                                    00043200
01 0588  06 9BB4  0435     MVC(131,BUFFER,BLANKS);   WRITE;                                                        00043300
01 0598  06 9BB4  0436   END;                                                                                      00043400
01 059A  06 9BB4  0437    PROCEDURE PRINTSECTION(R9);                                                              00043500
01 059A  06 9BB4  0438       BEGIN R7 := MSTART SHRL 2 + M64;  R8 := MCOUNT SHLL 1;                                00043600
01 05AE  06 9BB4  0439          FOR R1 := MSTART SHRL 2 + 64 STEP 64 UNTIL R7 DO                                   00043700
01 05BA  06 9BB4  0440          BEGIN R2 := R1 SHRL 6 * 12S;  R2 := @SYTB(R2);                                     00043800
01 05CC  06 9BB4  0441             MVC(11,BUFFER(1),B2); R2 := 0;                                                  00043900
01 05D6  06 9BB4  0442             FOR R3 := R1 STEP 1 UNTIL R3 DO                                                 00044000
01 05D8  06 9BB4  0443             BEGIN IC(R5,H(R3)); SLDL(R4,24); R6 := R2 + 6;                                  00044100
01 05EA  C6 9BB4  0444                FOR R2 := R2 STEP 2 UNTIL R6 DO                                              00044200
01 05EA  06 9BB4  0445                IF R2 <= R8 THEN                                                             00044300
01 05F4  06 9BB4  0446                BEGIN R4 := 0; SLDL(R4,2); IC(R4,RELATION(R4));                              00044400
01 0600  06 9BB4  0447                   STC(R4,BUFFER(R2+16));                                                    00044500
01 0604  06 9BB4  0448                END ELSE GOTO L1;                                                            00044600
01 060E  C6 9BB4  0449             END;                                                                            00044700
01 0618  C6 9BB4  0450   L1:       R0 := @BUFFER; WRITE;                                                           00044800
01 0626  06 9BB4  0451          END;                                                                               00044900
01 063C  06 9BB4  0452       END;                                                                                  00045000
01 0632  06 9BB4  0453                                                                                             00045100
01 0632  06 9BB4  0454    L: PAGE; R1 := M - MSTART; IF R1 < MCOUNT THEN MCOUNT := R1;                             00045200
01 065C  06 9BB4  0455       HEADING; PRINTSECTION; HEADING;                                                       00045300
01 065C  06 9BB4  0456       R1 := MSTART +  52; MSTART := R1; IF R1 < M THEN GOTO L;                              00045400
01 0670  06 9BB4  0457    END;                                                                                     00045500
01 067C  06 9BB4  0458                                                                                             00045600
```

```
01 067C  06 9BB4  0459    IF LEFTFLAG THEN                                                                  00045700
01 0678  06 9BB4  0460    BEGIN COMMENT WRITE OUT SYMBOL RELATIONS FROM LEFT SIDE;                          00045800
01 0678  06 9BB4  0461       MVC(131,BUFFER,BLANKS); PAGE;                                                  00045900
01 0688  06 9BB4  0462       MVC(10,BUFFER(3),"LEFT SYMBOL"); R0 := @BUFFER;                                00046000
01 0692  06 9BB4  0463       MVC(14,BUFFER(66),"RIGHT RELATIONS");                                          00046100
01 0698  06 9BB4  0464       WRITE; MVC(131,BUFFER,BLANKS); WRITE;                                          00046200
01 06B2  06 9BB4  0465       FOR R1 := 64 STEP 64 UNTIL M64 DO                                              00046300
01 06B6  06 9BB4  0466       BEGIN R2 := R1 SHRL 6; CVD(R2,DEC);                                            00046400
01 06C4  06 9BB4  0467          R2 := R2 * 12S; R2 := @SYTB(R2);                                            00046500
01 06CC  06 9BB4  0468          MVC(3,BUFFER(_1),FMASK); MVC(11,BUFFER(4),B2);                              00046600
01 06D8  06 9BB4  0469          ED(3,BUFFER(_1),DEC(6)); R6 := 0; R9 := 19;                                 00046700
01 06E6  06 9BB4  0470          FOR R2 := R1 STEP 4 UNTIL R2 DO                                             00046800
01 06E8  06 9BB4  0471          BEGIN R5 := H(R2); R7 := R6 + 15;                                           00046900
01 06F6  06 9BB4  0472             FOR R6 := R6 STEP 1 UNTIL R7 DO                                          00047000
01 06F6  06 9BB4  0473             IF R6 > M THEN GOTO L1 ELSE                                              00047100
01 0702  06 9BB4  0474             BEGIN R4 := R4 - R4; SLDL(R4,2); IC(R4,RELATION(R4));                    00047200
01 070C  06 9BB4  0475                IF R4 ¬= " " THEN                                                     00047300
01 0714  06 9BB4  0476                BEGIN IF R9 > 128 THEN                                                00047400
01 071C  06 9BB4  0477                   BEGIN R0 := @BUFFER; WRITE;                                        00047500
01 072A  06 9BB4  0478                      MVC(131,BUFFER,BLANKS); R9 := 19;                               00047600
01 0734  06 9BB4  0479                   END; R3 := @BUFFER(R9); MVI("|",B3); STC(R4,B3(2));                00047700
01 0740  06 9BB4  0480                   MVC(3,B3(3),FMASK); CVD(R6,DEC);                                   00047800
01 074A  06 9BB4  0481                   ED(3,B3(3),DEC(6)); R8 := R6 * 12S;                                00047900
01 0756  06 9BB4  0482                   R8 := @SYTB(R8); MVC(11,B3(8),B8); R9 := R9 + 22;                  00048000
01 0764  06 9BB4  0483                END;                                                                  00048100
01 0764  06 9BB4  0484             END;                                                                     00048200
01 076E  06 9BB4  0485          END;                                                                        00048300
01 0778  06 9BB4  0486    L1:   R0 := @BUFFER; WRITE; MVC(131,BUFFER,BLANKS); WRITE;                        00048400
01 0796  06 9BB4  0487       END;                                                                           00048500
01 07A2  06 9BB4  0488    END;                                                                              00048600
01 07A2  06 9BB4  0489                                                                                      00048700
01 07A2  06 9BB4  0490    IF RIGHTFLAG THEN                                                                 00048800
01 07AA  06 9BB4  0491    BEGIN COMMENT WRITE OUT SYMBOL RELATIONS FROM RIGHT SIDE;                         00048900
01 07AA  06 9BB4  0492       MVC(131,BUFFER,BLANKS); PAGE;                                                  00049000
01 07BA  06 9BB4  0493       MVC(11,BUFFER(113),"RIGHT SYMBOL"); R0 := @BUFFER;                             00049100
01 07C4  06 9BB4  0494       MVC(13,BUFFER(46),"LEFT RELATIONS");                                           00049200
01 07CA  06 9BB4  0495       WRITE; MVC(131,BUFFER,BLANKS); WRITE;                                          00049300
01 07E4  06 9BB4  0496       FOR R1 := 1 STEP 1 UNTIL M DO                                                  00049400
01 07E8  06 9BB4  0497       BEGIN CVD(R1,DEC); R2 := R1 * 12S; R2 := @SYTB(R2);                            00049500
01 07FA  06 9BB4  0498          MVC(3,BUFFER(110),FMASK); MVC(11,BUFFER(115),B2);                           00049600
01 0806  06 9BB4  0499          ED(3,BUFFER(110),DEC(6)); R9 := 86;                                         00049700
01 0810  06 9BB4  0500          R2 := R1; SRDL(R2,4); R3 := R3 SHRL 27; R2 := R2 SHLL 2;                    00049800
01 081E  06 9BB4  0501          R8 := M64 + R2; FOR R2 := R2 + 64 STEP 64 UNTIL R8 DO                       00049900
01 0828  06 9BB4  0502          BEGIN R5 := H(R2); SLDL(R4,B3(2)); R4 := R4 AND 3;                          00050000
01 0838  06 9BB4  0503             IC(R4,RELATION(R4)); IF R4 ¬= " " THEN                                   00050100
01 0844  06 9BB4  0504             BEGIN IF R9 < _2 THEN                                                    00050200
01 084C  06 9BB4  0505                BEGIN R0 := @BUFFER; WRITE;                                           00050300
01 085A  06 9BB4  0506                   MVC(131,BUFFER,BLANKS); R9 := 86;                                  00050400
01 0864  06 9BB4  0507                END; R5 := @BUFFER(R9); MVI("|",B5(21)); STC(R4,B5(19));              00050500
01 0870  06 9BB4  0508                R4 := R2 SHRL 6; CVD(R4,DEC); R4 := R4 * 12S;                         00050600
01 087E  06 9BB4  0509                R4 := @SYTB(R4); MVC(3,B5(1),FMASK);                                  00050700
01 0888  06 9BB4  0510                MVC(11,B5(6),B4); ED(3,B5(1),DEC(6)); R9 := R9 - 22;                  00050800
01 0898  06 9BB4  0511             END;                                                                     00050900
01 0898  06 9BB4  0512          END;                                                                        00051000
01 08A2  06 9BB4  0513          R0 := @BUFFER; WRITE; MVC(131,BUFFER,BLANKS); WRITE;                        00051100
01 08C0  06 9BB4  0514       END;                                                                           00051200
01 08CC  06 9BB4  0515    END;                                                                              00051300
01 08CC  06 9BB4  0516                                                                                      00051400
```

```
01 08CC  06 9BB4  0517    IF FUNCTIONFLAG THEN
01 08D4  06 9BB4  0518    BEGIN COMMENT FIND F AND G FUNCTIONS;
01 08D4  06 9BB4  0519       ARRAY  640  INTEGER STACK SYN R;
01 08D4  06 9BB4  0520       INTEGER REGISTER T SYN R0,I SYN R1,J SYN R2,
01 08D4  06 9BB4  0521          FMIN SYN R4,GMIN SYN R4,K SYN R5,K1 SYN R6;
01 08D4  C6 9BB4  0522
01 08D4  06 9BB4  0523    SEGMENT PROCEDURE FUNCTIONS(R3);
01 08D4  C6 9BB4  0524    BEGIN
09 0000  06 9BB4  0525
09 0000  06 9BB4  0526    PROCEDURE GET(R8);
09 0000  06 9BB4  0527       BEGIN COMMENT R1 = I, R2 = J, AND R0 = T AT EXIT;
09 0004  06 9BB4  0528          STM(R1,R3,SAVE); R1 := R1 SHLL 5; SRDL(R2,5);
09 0010  06 9BB4  0529          R2 := R2 SHLL 2 + R1; R3 := R3 SHRL 27;
09 001A  06 9BB4  0530          R1 := H(R2); SLDL(R0,B3(2)); R0 := R0 AND 3;
09 0026  06 9BB4  0531          LM(R1,R3,SAVE);
09 002A  06 9BB4  0532       END;
09 002C  06 9BB4  0533    PROCEDURE THRU(R8);
09 002C  06 9BB4  0534       BEGIN COMMENT PRINT FUNCTION CONFLICT LIST;
09 002C  C6 9BB4  0535          MVC(131,BUFFER,BLANKS);
09 0032  06 9BB4  0536    LOOP: R4 := R1 * SIX; R4 := @SYTB(R4); MVC(11,BUFFER(8),B4);
09 0042  06 9BB4  0537          R4 := R2 * SIX; R4 := @SYTB(R4); MVC(11,BUFFER(33),B4);
09 0052  06 9BB4  0538          GET; R4 := R0; IC(R4,RELATION(R4)); STC(R4,BUFFER(26));
09 0060  06 9BB4  0539          MVC(3,BUFFER(21),FMASK); MVC(3,BUFFER(28),FMASK);
09 006C  06 9BB4  0540          R1 := R1 SHRL 1; CVD(R1,DEC); ED(3,BUFFER(21),DEC(6));
09 007A  06 9BB4  0541          R2 := R2 SHRL 1; CVD(R2,DEC); ED(3,BUFFER(28),DEC(6));
09 0088  06 9BB4  0542          R0 := @BUFFER; WRITE; R12 := R12 - 16; LM(R1,R2,B12);
09 009E  06 9BB4  0543          IF R12 >= U THEN GOTO LOOP;
09 00A6  06 9BB4  0544          RESET(FUNCTIONFLAG);  RESET(TABLEFLAG);  GOTO EXIT;
09 00B2  06 9BB4  0545       END;
09 00B4  C6 9BB4  0546
09 00B4  06 9BB4  0547    PROCEDURE  FIXUP(R3);
09 00B4  C6 9BB4  0548       BEGIN COMMENT FIXUP ROW IF R7 = 0 ELSE FIXUP COLUMN;
09 00B4  06 9BB4  0549       STM(R1,R4,B12); R12 := R12 + 16; IF R7 = 0 THEN
09 00C2  06 9BB4  0550       BEGIN R4 := G(J) + R0; F(I) := R4;
09 00CC  C6 9BB4  0551          IF K = K1 THEN
09 00D2  06 9BB4  0552          BEGIN J := K; GET; IF T = LSS THEN
09 00EC  06 9BB4  0553             BEGIN IF R4 >= G(J) THEN THRU;
09 00EC  06 9BB4  0554             END ELSE IF T = EQL THEN
09 00F8  06 9BB4  0555             BEGIN IF R4 ¬= G(J) THEN THRU;
09 0104  06 9BB4  0556             END;
09 0104  06 9BB4  0557          END;
09 0104  06 9BB4  0558          FOR J := K1 STEP _2 UNTIL 2 DO
09 0106  C6 9BB4  0559          BEGIN GET; IF T = LSS THEN
09 0116  06 9BB4  0560             BEGIN IF R4 >= G(J) THEN BEGIN R0 := 1; R7 := 1; FIXUP; END;
09 012A  06 9BB4  0561             END ELSE IF T = EQL THEN
09 0136  C6 9BB4  0562             BEGIN IF R4 ¬= G(J) THEN BEGIN R0 := 0; R7 := 1; FIXUP; END;
09 014A  06 9BB4  0563             END;
09 014A  06 9BB4  0564          END;
09 0156  06 9BB4  0565       END ELSE
09 0156  06 9BB4  0566       BEGIN R4 := F(I) + R0; G(J) := R4;
09 0164  06 9BB4  0567          IF K1 ¬= K THEN
09 016A  06 9BB4  0568          BEGIN I := K; GET; IF T = GTR THEN
09 0178  06 9BB4  0569             BEGIN IF R4 >= F(I) THEN THRU;
09 0184  06 9BB4  0570             END ELSE IF T = EQL THEN
09 0190  06 9BB4  0571             BEGIN IF R4 ¬= F(I) THEN THRU;
09 019C  06 9BB4  0572             END;
09 019C  06 9BB4  0573          END;
09 019C  06 9BB4  0574          FOR I := K STEP _2 UNTIL 2 DO
```

```
09 019E  06 9BB4  0575          BEGIN GET; IF T = GTR THEN
09 01AE  06 9BB4  0576             BEGIN IF R4 >= F(I) THEN BEGIN R0 := 1; R7 := 0; FIXUP; END;
09 01C2  06 9BB4  0577             END ELSE IF T = EQL THEN
09 01CE  06 9BB4  0578             BEGIN IF R4 ¬= F(I) THEN BEGIN R0 := 0; R7 := 0; FIXUP; END;
09 01E2  06 9BB4  0579             END;
09 01E2  06 9BB4  0580          END;
09 01EE  06 9BB4  0581       END; R12 := R12 - 16; LM(R1,R4,B12);
09 01F6  06 9BB4  0582       END;   COMMENT  END OF FIXUP;
09 01F8  06 9BB4  0583
09 01F8  06 9BB4  0584       R3SAVE := R3;   R0 := @BLANKS; WRITE;
09 020A  06 9BB4  0585       K1 := 0; U := R12; PAGE;
09 021C  06 9BB4  0586       FOR K := 0 STEP 2 UNTIL 510 DO
09 0220  06 9BB4  0587       BEGIN  F(K) := K1;  G(K) := K1;
09 022C  06 9BB4  0588       END;
09 0238  06 9BB4  0589       FOR K := 2 STEP 2 UNTIL M2 DO
09 023C  06 9BB4  0590       BEGIN FMIN := 1; I := K;
09 0246  06 9BB4  0591          FOR J := 2 STEP 2 UNTIL K1 DO
09 024A  06 9BB4  0592          BEGIN GET; IF T = GTR THEN
09 025A  06 9BB4  0593             BEGIN IF FMIN <= G(J) THEN FMIN := G(J) + 1;
09 026A  06 9BB4  0594             END ELSE IF T = EQL THEN
09 0276  06 9BB4  0595             BEGIN IF FMIN < G(J) THEN FMIN := G(J);
09 0282  06 9BB4  0596             END;
09 0282  06 9BB4  0597          END;
09 028C  06 9BB4  0598          F(I) := FMIN;
09 0290  06 9BB4  0599          FOR J := K1 STEP _2 UNTIL 2 DO
09 0292  06 9BB4  0600          BEGIN GET; IF T = LSS THEN
09 02A2  06 9BB4  0601             BEGIN IF FMIN >= G(J) THEN BEGIN R0 := 1; R7 := 1; FIXUP; END;
09 02B6  06 9BB4  0602             END ELSE IF T = EQL THEN
09 02C2  06 9BB4  0603             BEGIN IF FMIN > G(J) THEN BEGIN R0 := 0; R7 := 1; FIXUP; END;
09 02D6  06 9BB4  0604             END;
09 02D6  06 9BB4  0605          END;
09 02E2  06 9BB4  0606          K1 := K1 + 2; GMIN := 1; J := K;
09 02EC  06 9BB4  0607          FOR I := 2 STEP 2 UNTIL K DO
09 02F0  06 9BB4  0608          BEGIN GET; IF T = LSS THEN
09 0300  06 9BB4  0609             BEGIN IF GMIN <= F(I) THEN GMIN := F(I) + 1;
09 0310  06 9BB4  0610             END ELSE IF T = EQL THEN
09 031C  06 9BB4  0611             BEGIN IF GMIN < F(I) THEN GMIN := F(I);
09 0328  06 9BB4  0612             END;
09 0328  06 9BB4  0613          END;
09 0332  06 9BB4  0614          G(J) := GMIN;
09 0336  06 9BB4  0615          FOR I := K STEP _2 UNTIL 2 DO
09 0338  06 9BB4  0616          BEGIN GET; IF T = GTR THEN
09 0348  06 9BB4  0617             BEGIN IF GMIN >= F(I) THEN BEGIN R0 := 1; R7 := 0; FIXUP; END;
09 035C  06 9BB4  0618             END ELSE IF T = EQL THEN
09 0368  06 9BB4  0619             BEGIN IF GMIN > F(I) THEN BEGIN R0 := 0; R7 := 0; FIXUP; END;
09 037C  06 9BB4  0620             END;
09 037C  06 9BB4  0621          END;
09 0388  06 9BB4  0622       END;
09 0394  06 9BB4  0623       R2 := @SYTB(12); MVC(131,BUFFER,BLANKS);
09 039E  06 9BB4  0624       MVC(19,BUFFER(7),"PRECEDENCE FUNCTIONS"); R0 := @BUFFER; WRITE;
09 03B2  06 9BB4  0625       MVC(19,BUFFER(7),BLANKS); R0 := @BUFFER; WRITE;
09 03C6  06 9BB4  0626          R2 := @SYTB(12);
09 03CA  06 9BB4  0627       FOR R1 := 2 STEP 2 UNTIL M2 DO
09 03CE  06 9BB4  0628       BEGIN MVC(29,BUFFER(7),FMASK); R0 := R1 SHRL 1; CVD(R0,DEC);
09 03E2  06 9BB4  0629          ED(3,BUFFER(7),DEC(6)); R0 := F(R1); CVD(R0,DEC);
09 03F0  06 9BB4  0630          ED(3,BUFFER(27),DEC(6)); R0 := G(R1); CVD(R0,DEC);
09 03FE  06 9BB4  0631          ED(3,BUFFER(33),DEC(6)); MVC(11,BUFFER(13),B2);
09 040A  06 9BB4  0632          R0 := @BUFFER; WRITE; R2 := R2 + 12;
```

```
09 041C   06 9BB4          0633       END;
09 0428   C6 9BB4          0634    EXIT:  R3 := R3SAVE;
09 042C   06 9BB4          0635     END;

                          SEGMENT 09  NAME = SEG#09       LENGTH = 0470  BASE REG = 15

01 08D4   06 9BB4          0636
01 08D4   06 9BB4          0637     FUNCTIONS;
01 08DE   06 9BB4          0638     END;
01 08DE   06 9BB4          0639
01 08DE   06 9BB4          C640     IF TABLEFLAG THEN                                                    00000050
01 08E6   06 9BB4          0641     BEGIN COMMENT NEW SYNTAX PUNCH CARD PROGRAM;                         00000100
01 08E6   06 9BB4          0642       SEGMENT BASE R12;                                                  00000200
01 08EA   1C 0000          0643        FUNCTION EDM(5,#DF00);                                            00000800
01 08EA   10 0000          0644        ARRAY  3  INTEGER SAVER9;                                         00000900
01 08EA   10 000C          0645        INTEGER REGISTER NEXT SYN R8,IND SYN R7,                          00001100
01 08EA   10 000C          0646           CNT SYN R6,NUM SYN R5,POUT SYN R9,ADDBASE SYN R10;             00001200
01 08EA   10 000C          0647        LONG REAL NUMWORK,NUMEDIT,CNTEDIT,                                00001300
01 08EA   10 0028          0648           PATTERN=#4020202020202120;                                     00001400
01 08EA   10 0030          0649        INTEGER L, T, PINIT;                                              00001500
01 08EA   10 003C          0650        ARRAY 6 BYTE NAME;                                                00001550
01 08EA   10 0042          0651        ARRAY  3  SHORT INTEGER MOVECNT=(#D200,@MEM(POUT),@B1),           00001600
01 08EA   10 0048          0652           MOVENUM=(#D200,@MEM(POUT),@B3);                                00001700
01 08EA   10 004E          0653        FUNCTION TR(5,#DC00);                                             00002300
01 08EA   10 004E          0654        SHORT INTEGER B4 SYN MEM(R4);                                     00002400
01 08EA   10 004E          0655                                                                          00002700
01 08EA   10 C04E          0656        INTEGER R9SAVE;                                                   00002800
01 08EA   10 0054          0657        ARRAY 256 SHORT INTEGER RMAP;                                     00002900
01 08EA   10 0254          0658        ARRAY 256 BYTE CMAP;                                              00003000
01 08EA   10 0354          0659        ARRAY 4097 LOGICAL MATRIX SYN H;                                  00003100
01 08EA   10 0354          C660        ARRAY 16388 BYTE MATRIX2;                                         00003200
01 08EA   10 4358          0661                                                                          00003300
01 08EA   10 4358          0662     PROCEDURE OUTPUT(R11); COMMENT OUTPUT NUM WITH A REPITION COUNT OF CNT;00003400
01 08EA   10 4358          0663        BEGIN CVD(NUM,NUMWORK); MVC(7,NUMEDIT,PATTERN);                   00003500
01 08F8   10 4358          0664           R1 := @NUMEDIT(7); EDM(7,NUMEDIT,NUMWORK(4));                  00003600
01 0902   10 4358          0665           R2 := @NUMEDIT( 7)- R1; R3 := R1;                              00003700
01 090A   10 4358          0666           R1 := R2 + 2 * CNT - 5; IF R1 > R2 THEN                        00003800
01 091C   10 4358          0667           BEGIN CVD(CNT,NUMWORK); MVC(7,CNTEDIT,PATTERN);                00003900
01 0926   10 4358          0668              R1 := @CNTEDIT(7); EDM(7,CNTEDIT,NUMWORK(4));               00004000
01 0930   10 4358          0669              R4 := @CNTEDIT( 7)- R1; R0 := POUT + R2 + R4 + 5;           00004100
01 0940   10 4358          0670              IF R0 > 72 THEN                                             00004200
01 0948   10 4358          0671              BEGIN R0 := @PBUF; PUNCH;                                   00004300
01 0956   10 4358          0672                 POUT := PINIT; R0 := POUT + R2 + R4 + 5;                 00004400
01 0964   10 4358          0673                 MVC(79,PBUF,BLANKS);                                     00004500
01 096A   10 4358          0674              END;                                                        00004600
01 096A   10 4358          0675              POUT := @PBUF(POUT); EX(R4,MOVECNT);                        00004700
01 0972   10 4358          0676              POUT := POUT + R4; MVI("(",MEM(POUT+1));                    00004800
01 0978   10 4358          0677              POUT := POUT + 2; EX(R2,MOVENUM);                           00004900
01 0980   10 4358          0678              POUT := POUT + R2; MVI(")",MEM(POUT+1));                    00005000
01 0986   10 4358          0679              MVI(",",MEM(POUT+2)); POUT := R0;                           00005100
01 098C   10 4358          0680           END ELSE                                                       00005200
01 098C   10 4358          0681           FOR CNT := CNT STEP _1 UNTIL 1 DO                              00005300
01 0990   10 4358          0682           BEGIN R1 := POUT + R2 + 2; IF R1 > 72 THEN                     00005400
01 09A4   10 4358          0683              BEGIN R0 := @PBUF; PUNCH;                                   00005500
01 09B2   10 4358          0684                 POUT := PINIT; R1 := POUT + R2 + 2;                      00005600
01 09BE   10 4358          0685                 MVC(79,PBUF,BLANKS);                                     00005700
01 09C4   10 4358          0686              END;                                                        00005800
01 09C4   10 4358          0687              POUT := @PBUF(POUT); EX(R2,MOVENUM);                        00005900
```

```
01 09CC   10 4358   0688          POUT := POUT + R2; MVI(",",MEM(POUT+1));                                    00006000
01 09D2   10 4358   0689          POUT := R1;                                                                  00006100
01 09D4   10 4358   0690       END;                                                                            00006200
01 09E0   10 4358   0691    END;                                                                               00006300
01 09E2   10 4358   0692                                                                                       00006400
01 09E2   10 4358   0693 PROCEDURE LENGTH1(R9); COMMENT PUNCH BYTE TABLES;                                     00009900
01 09E2   10 4358   0694    BEGIN BYTE NEXT SYN MEM(ADDBASE);                                                  00010000
01 09E2   10 4358   0695       STM(R9,R11,SAVER9);  MVC(79,PBUF,BLANKS);  ADDBASE := R0;                       00010100
01 09EE   10 4358   0696       POUT := PINIT-3; R1 := @PBUF(POUT); MVC(4,B1,"ARRAY");                          00010110
01 0A00   10 4358   0697       POUT := POUT + 6;  CNT := 1; NUM := L + 1;  OUTPUT;                             00010120
01 0A14   10 4358   0698       R1 := @PBUF(POUT-1);  MVC(4,B1," BYTE");                                        00010130
01 0A1E   10 4358   0699       MVC(5,B1(6),NAME);  MVI("=",B1(13));                                            00010140
01 0A28   10 4358   0700       R0 := @PBUF;  PUNCH;  MVC(79,PBUF,BLANKS);                                      00010150
01 0A3C   10 4358   0701       POUT := PINIT + 1; R1 := @PBUF(POUT-1); MVI("(",B1);                            00010200
01 0A4C   10 4358   0702       CNT := 1; NUM := 0; IC(NUM,NEXT);                                               00010300
01 0A58   10 4358   0703       FOR IND := 1 STEP 1 UNTIL L DO                                                  00010400
01 0A5C   10 4358   0704       BEGIN R1 := 0; IC(R1,NEXT(IND));                                                00010500
01 0A68   10 4358   0705          IF NUM = R1 THEN CNT := CNT + 1 ELSE                                         00010600
01 0A72   10 4358   0706          BEGIN OUTPUT; CNT := 1; NUM := 0; IC(NUM,NEXT(IND));                         00010700
01 0A86   10 4358   0707          END;                                                                         00010800
01 0A86   10 4358   0708       END;                                                                            00010900
01 0A92   10 4358   0709       OUTPUT; R1 := @PBUF(POUT-1); MVI(")",B1);                                       00011000
01 0A9E   10 4358   0710       IF POUT = 72 THEN                                                               00011100
01 0AA6   10 4358   0711       BEGIN R0 :=@PBUF; PUNCH; R1 := R0 + PINIT;                                      00011200
01 0ABA   10 4358   0712          MVC(79,PBUF,BLANKS);                                                         00011300
01 0AC0   10 4358   0713       END ELSE R1 := R1 + 1;                                                          00011400
01 0AC8   10 4358   0714       MVI(";",B1); R0 := @PBUF; PUNCH;                                                00011500
01 0ADA   10 4358   0715       MVC(79,PBUF,BLANKS); PUNCH;                                                     00011600
01 0AEA   10 4358   0716       LM(R9,R11,SAVER9);                                                              00011700
01 0AEE   10 4358   0717    END;                                                                               00011800
01 0AF0   10 4358   0718 PROCEDURE LENGTH2(R9); COMMENT PUNCH HALFWORD TABLES;                                 00011900
01 0AF0   10 4358   0719    BEGIN SHORT INTEGER NEXT SYN MEM(ADDBASE);                                         00012000
01 0AFC   10 4358   0720       STM(R9,R11,SAVER9);  MVC(79,PBUF,BLANKS);  ADDBASE := R0;                       00012100
01 0AFC   10 4358   0721       POUT := PINIT-3; R1 := @PBUF(POUT); MVC(4,B1,"ARRAY");                          00012110
01 0B0E   10 4358   0722       POUT := POUT + 6;  CNT := 1; NUM := M + 1;  OUTPUT;                             00012120
01 0B22   10 4358   0723       R1 := @PBUF(POUT-1);  MVC(13,B1," SHORT INTEGER");                              00012130
01 0B2C   10 4358   0724       MVC(5,B1(15),NAME);  MVI("=",B1(22));                                           00012140
01 0B36   10 4358   0725       R0 := @PBUF;  PUNCH;  MVC(79,PBUF,BLANKS);                                      00012150
01 0B4A   10 4358   0726       POUT := PINIT + 1; R1 := @PBUF(POUT-1); MVI("(",B1);                            00012200
01 0B5A   10 4358   0727       CNT := 1; NUM := NEXT;                                                          00012300
01 0B62   10 4358   0728       FOR IND := 2 STEP 2 UNTIL M2 DO                                                 00012400
01 0B66   10 4358   0729       IF NUM = NEXT(IND) THEN CNT := CNT + 1 ELSE                                     00012500
01 0B76   10 4358   0730       BEGIN OUTPUT; CNT := 1; NUM := NEXT(IND);                                       00012600
01 0B86   10 4358   0731       END;                                                                            00012700
01 0B92   10 4358   0732       OUTPUT; R1 := @PBUF(POUT-1); MVI(")",B1);                                       00012800
01 0B9E   10 4358   0733       IF POUT = 72 THEN                                                               00012900
01 0BA6   10 4358   0734       BEGIN R0 := @PBUF; PUNCH; R1 := R0 + PINIT;                                     00013000
01 0BBA   10 4358   0735          MVC(79,PBUF,BLANKS);                                                         00013100
01 0BC0   10 4358   0736       END ELSE R1 := R1 + 1;                                                          00013200
01 0BC8   10 4358   0737       MVI(";",B1); R0 := @ PBUF; PUNCH;                                               00013300
01 0BDA   10 4358   0738       MVC(79,PBUF,BLANKS); PUNCH;                                                     00013400
01 0BEA   10 4358   0739       LM(R9,R11,SAVER9);                                                              00013500
01 0BEE   10 4358   0740    END;                                                                               00013600
01 0BF0   10 4358   0741                                                                                       00013700
01 0BF0   10 4358   0742                                                                                       00013800
01 0BF0   10 4358   0743 PROCEDURE MATRIXMTBPRTB1(R9);                                                        00013900
01 0BF0   10 4358   0744    BEGIN  R1 := 0; MTB(0) := R1;  R0 := 255; STC(R0,PRTB1(0));                        00014000
01 0C00   10 4358   0745       FOR R2 := 2 STEP 2 UNTIL M2 DO                                                  00014100
```

```
01 0C04  10 4358  C746          BEGIN R6 := R1 + 1; MTB(R2) := R6; R5 := 1;                                  00014200
01 0C16  10 4358  0747             FOR R3 := 14 STEP 12 UNTIL N12 DO                                         00014300
01 0C1A  10 4358  0748             BEGIN R7 := PRTB(R3); IF R2 = R7 THEN                                     00014400
01 0C28  10 4358  C749                BEGIN R1 := R1 + 1; R6 := R1; R8 := R3 + 8;                            00014500
01 0C34  10 4358  0750                   FOR R4 := R3 STEP 2 UNTIL R8 DO                                     00014600
01 0C36  10 4358  0751                   BEGIN R7 := PRTB(R4) SHRL 1; IF R7 ¬= 0 THEN                        00014700
01 0C48  10 4358  0752                      BEGIN R1 := R1 + 1; STC(R7,PRTB1(R1));                           00014800
01 0C50  10 4358  0753                      END;                                                             00014900
01 0C50  10 4358  0754                   END;                                                                00015000
01 0C5A  10 4358  0755                   R7 := R1 - R6 - 1; STC(R7,PRTB1(R6));                               00015100
01 0C66  10 4358  0756                   R1 := R1 + 1; R7 := PRTB(R3-2) SHRL 1;                              00015200
01 0C72  10 4358  0757                   STC(R7,PRTB1(R1));                                                  00015300
01 0C76  10 4358  C758                   R1 := R1 + 1; STC(R5,PRTB1(R1));                                    00015400
01 0C7E  10 4358  0759                END; R5 := R5 + 1;                                                     00015500
01 0C82  10 4358  C760             END; R1 := R1 + 1; STC(R0,PRTB1(R1));                                     00015600
01 0C96  10 4358  0761          END; L := R1;                                                                00015700
01 0CA6  10 4358  0762       END;                                                                            00015800
01 0CA8  10 4358  0763                                                                                       00015805
01 0CA8  10 4358  0764    PROCEDURE FUNCTIONMTBPRTB1(R9);                                                    00015810
01 0CA8  10 4358  0765       BEGIN  R1 := 0; MTB(0) := R1;  R0 := 255; STC(R0,PRTB1(0));                     00015815
01 0CB8  10 4358  0766          FOR R2 := 2 STEP 2 UNTIL M2 DO                                               00015820
01 0CBC  10 4358  C767          BEGIN R6 := R1 + 1; MTB(R2) := R6; R5 := 1;                                  00015825
01 0CCE  10 4358  0768             FOR R3 := 14 STEP 12 UNTIL N12 DO                                         00015830
01 0CD2  10 4358  0769             BEGIN  R7 := PRTB(R3); IF R2 = R7 THEN                                    00015835
01 0CE0  10 4358  C770                BEGIN R1 := R1 + 1; R6 := R1;                                          00015840
01 0CE6  10 4358  0771                   FOR R4 := R3 + 8 STEP _2 UNTIL R3 DO                                00015845
01 0CEC  10 4358  C772                   BEGIN  R7 := PRTB(R4); IF R7 ¬= 0 THEN                              00015850
01 0CFA  10 4358  0773                      BEGIN R1 := R1 + 1; STC(R7,PRTB1(R1));                           00015855
01 0D02  10 4358  C774                      END;                                                             00015860
01 0D02  10 4358  0775                   END;                                                                00015865
01 0D0C  10 4358  C776                   R7 := R1 - R6 - 1;  STC(R7,PRTB1(R6));                              00015870
01 0D18  10 4358  0777                   R1 := R1 + 1; R7 := PRTB(R3-2); STC(R7,PRTB1(R1));                  00015875
01 0D24  10 4358  0778                   R1 := R1 + 1; STC(R5,PRTB1(R1));                                    00015880
01 0D2C  10 4358  0779                END; R5 := R5 + 1;                                                     00015885
01 0D30  10 4358  C780             END; R1 := R1 + 1; STC(R0,PRTB1(R1));                                     00015890
01 0D44  10 4358  0781          END; L := R1;                                                                00015895
01 0D54  10 4358  0782       END;                                                                            00015896
01 0D56  10 4358  0783                                                                                       00015900
01 0D56  10 4358  0784    SEGMENT PROCEDURE BUILDMATRIX2(R9);                                                00016000
01 0D56  10 4358  0785       BEGIN                                                                           00016100
11 0000  10 4358  C786                                                                                       00016150
11 0000  10 4358  0787       PROCEDURE   FETCH (R14);                                                        00016200
11 0000  10 4358  0788       COMMENT FETCH   TAKES THE RELATION BETWEEN C(R1) AND C(R2) FROM                 00016300
11 0000  10 4358  0789               THE MATRIX AND PUTS IT IN R4.   REGISTERS R1 - R5  ARE                  00016400
11 0000  10 4358  0790               USED AND DESTROYED.  R1 AND R2 ARE CLEARED;                             00016500
11 0000  10 4358  C791       BEGIN                                                                           00016600
11 0004  10 4358  C792          R1 := R1 SHLL 6;      SRDL(R2,4);                                            00016700
11 000C  10 4358  C793          R2 := R2 SHLL 2 + R1;   R3 := R3 SHRL 28 SHLL 1;                             00016800
11 001A  10 4358  0794          R5 := MATRIX(R2);     SLDL(R4,B3);                                           00016900
11 0022  10 4358  0795          R4 := 0;     R1 := R4;    R2 := R4;                                          00017000
11 002A  10 4358  0796          SLDL(R4,2);   COMMENT  BITPAIR IS NOW IN R4;                                 00017100
11 002E  10 4358  0797       END;                                                                            00017200
11 0030  10 4358  C798                                                                                       00017300
11 0030  10 4358  0799       R9SAVE := R9;  MVI(0,CMAP(0));                                                  00017400
11 0038  10 4358  0800       R8 := 1;  COMMENT COLUMN COUNT;                                                 00017500
11 003C  10 4358  0801       FOR R6 := 1 STEP 1 UNTIL  M  DO                                                 00017600
11 0040  10 4358  0802       BEGIN  R7 := R6-1;  T := R7;                                                    00017700
11 004E  10 4358  0803          FOR R9 := 0 STEP 1 UNTIL T DO                                                00017800
```

```
11 0052   10 4358    0804       BEGIN                                                                          00017900
11 0056   10 4358    0805         FOR R7 := 1 STEP 1 UNTIL  M  DO                                              00018000
11 005A   10 4358    0806         BEGIN                                                                        00018100
11 005E   10 4358    0807           R1 := R7;  R2 := R6;  FETCH;  R0 := R4;  R1 := R7;  R2 := R9;              00018200
11 006C   10 4358    0808           FETCH;  IF R0 ¬= R4 THEN GOTO L2;                                          00018300
11 0076   10 4358    0809         END;  COMMENT ELEMENT COMPARED;                                              00018400
11 0082   10 4358    0810         GOTO L3;                                                                     00018500
11 0086   10 4358    0811   L2: END; COMMENT ROW COMPARED;                                                     00018600
11 0092   10 4358    0812       R8 := R8+1;  COMMENT STEP COLUMN COUNT;                                        00018700
11 0096   10 4358    0813   L3: STC(R9,CMAP(R6));                                                              00018800
11 009A   10 4358    0814     END;                                                                             00019000
11 00A6   10 4358    0815     MVC(131,BUFFER,BLANKS);  CVD(R8,NUMWORK);  MVC(7,BUFFER,PATTERN);               00019100
11 00B6   10 4358    0816     ED(7,BUFFER,NUMWORK(4));  MVC(12,BUFFER(9),"BYTES PER ROW");                    00019200
11 00C2   10 4358    0817     R0 := @BUFFER;  WRITE;    R0 := 0;  RMAP(0) := R0;                             00019300
11 00D8   10 4358    0818     COMMENT  BUILD NEW MATRIX;                                                       00019400
11 00D8   10 4358    0819     R2 := 0; R8 := R8 - 1;  COMMENT NUMBERING STARTS FROM 0;                         00019500
11 00E0   10 4358    0820     FOR R1 := 0 STEP 1 UNTIL R8 DO STC(R0,MATRIX2(R1));                             00019550
11 00F6   10 4358    0821     FOR R6 := 2 STEP 2 UNTIL M2 DO                                                   00019600
11 00FA   10 4358    0822     BEGIN  R1 := R6 SHLL 5; R1 := @MATRIX(R1);  R3 := R1 - 64;                       00019700
11 010E   10 4358    0823       FOR R7 := @ MATRIX STEP 64 UNTIL R3 DO                                         00019800
11 0112   10 4358    0824       BEGIN  CLC(63,B1,B7);  IF = THEN GOTO L; END;                                  00019900
11 012A   10 4358    0825       COMMENT  NEW ROW;                                                              00020000
11 012A   10 4358    0826       R8 := R8 + 1;  STC(R0,MATRIX2(R8));   RMAP(R6) := R8;                          00020100
11 0136   10 4358    0827       FOR R9 := 1 STEP 1 UNTIL M DO                                                  00020200
11 013A   10 4358    0828       BEGIN  IC(R2,CMAP(R9));                                                        00020300
11 0142   10 4358    0829         IF R9 = R2 THEN                                                              00020400
11 0148   10 4358    0830         BEGIN  R1 := R6 SHRL 1; FETCH; R8 := R8 + 1;  STC(R4,MATRIX2(R8));           00020500
11 0156   10 4358    0831         END;                                                                         00020600
11 015A   10 4358    0832       END;                                                                           00020700
11 0166   10 4358    0833       GOTO EXIT;                                                                     00020800
11 016A   10 4358    0834     L:  R1 := @MATRIX;  R7 := R7 - R1 SHRL 5;  R7 := RMAP(R7);                       00021000
11 0178   10 4358    0835       RMAP(R6) := R7;                                                              00021100
11 017C   10 4358    0836     EXIT:                                                                          00021200
11 017C   10 4358    0837       END;                                                                         00021300
11 0188   10 4358    0838       R7 := 0;  R8 := R8 + 1;                                                      00021400
11 0190   10 4358    0839       MVC(131,BUFFER,BLANKS);  CVD(R8,NUMWORK);  MVC(7,BUFFER,PATTERN);            00021500
11 01A0   10 4358    0840       ED(7,BUFFER,NUMWORK(4));  MVC(15,BUFFER(9),"BYTES IN MATRIX2");              00021600
11 01AC   10 4358    0841       R0 := @BUFFER; WRITE;  R8 := R8 - 1; L := R8; R6 := 0;                       00021700
11 01C6   10 4358    0842       FOR R9 := 1 STEP 1 UNTIL  M  DO                                              00021800
11 01CA   10 4358    0843       BEGIN  IC(R6,CMAP(R9));                                                      00021900
11 01D2   10 4358    0844         IF R9 = R6 THEN                                                            00022000
11 01D8   10 4358    0845         BEGIN  R7 := R7+1;  STC(R7,CMAP(R9));                                      00022100
11 01E0   10 4358    0846         END ELSE                                                                   00022200
11 01E0   10 4358    0847         BEGIN  IC(R6,CMAP(R6));  STC(R6,CMAP(R9));                                 00022300
11 01EC   10 4358    0848         END;                                                                       00022400
11 01EC   10 4358    0849       END;                                                                         00022500
11 01F8   10 4358    0850       R9 := R9SAVE;                                                                00022600
11 01FC   10 4358    0851     END ;                                                                          00022700

                    SEGMENT 11  NAME = SEG#11      LENGTH = 0230  BASE REG = 15

01 0D56   10 4358    0852                                                                                   00022800
01 0D56   10 4358    0853       R1 := M SHLL 1;  M2 := R1;                                                  00022810
01 0D62   10 4358    0854       IF FUNCTIONFLAG THEN                                                        00022820
01 0D6A   10 4358    0855       BEGIN  COMMENT PRODUCE TABLES IN PL360 FORMAT;                              00022830
01 0D6A   10 4358    0856         FUNCTIONMTBPRTB1;    R0 := 7; PINIT := R0;                             00022840
01 0D76   10 4358    0857         R0 := @F;  MVC(5,NAME,"F     "); LENGTH2;                              00022850
01 0D84   10 4358    0858         R0 := @G;  MVC(5,NAME,"G     "); LENGTH2;                              00022860
```

```
01 0D92   10 4358    0859      R0 := @MTB; MVC(5,NAME,"MTB   "); LENGTH2;                                  00022870
01 0DA0   10 4358    0860      R0 := @PRTB1; MVC(5,NAME,"PRTB  "); LENGTH1;                                00022880
01 0DAE   10 4358    0861    END ELSE                                                                      00022890
01 0DAE   10 4358    0862    BEGIN  COMMENT PRODUCE TABLES IN ALGOL PASS 2 FORMAT;                         00022900
01 0DB2   10 4358    0863      MATRIXMTBPRTB1;      R0 := 3; PINIT := R0;                                  00025100
01 0DBE   10 4358    0864      R0 := @MTB; MVC(5,NAME,"MTB   "); LENGTH2;                                  00025200
01 0DCC   10 4358    0865      R0 := @PRTB1; MVC(5,NAME,"PRTB  "); LENGTH1;                                00025300
01 0DDA   10 4358    0866      MVC(131,BUFFER,BLANKS); R0 := L + 1;                                        00025400
01 0DE8   10 4358    0867      CVD(R0,NUMWORK); MVC(7,BUFFER,PATTERN);                                     00025500
01 0DF2   10 4358    0868      ED(7,BUFFER,NUMWORK(4)); MVC(12,BUFFER(9),"BYTES IN PRTB");                 00025600
01 0DFE   10 4358    0869      R0 := @BUFFER; WRITE;                                                       00025700
01 0E0C   10 4358    0870      BUILDMATRIX2;  R0 := @MATRIX2;  MVC(5,NAME,"MATRIX"); LENGTH1;              00025800
01 0E24   10 4358    0871      R8 := M; L := R8;  R0 := @RMAP;  MVC(5,NAME,"RMAP  "); LENGTH2;             00025900
01 0E3A   10 4358    0872      R0 := @CMAP;  MVC(5,NAME,"CMAP  "); LENGTH1;                                00026000
01 0E48   10 4358    0873    END;                                                                          00026100
01 0E48   10 4358    0874  END;                                                                            00026200

                   SEGMENT 10  NAME = SEG#10      LENGTH = 4358  BASE REG = 12

01 0E48   06 9BB4    0875                                                                                  00026300
01 0E48   06 9BB4    0876  EXIT:                                                                           00026400
01 0E48   06 9BB4    0877    IF ¬EOF THEN                                                                  00026500
01 0E50   06 9BB4    0878    BEGIN  R1 := 49; MCOUNT := R1;  MVI(0,CHANGE); MVI(0,ERRFLAG);                00026600
01 0E60   06 9BB4    0879      MVI(0,SYMPUNCHFLAG); MVI(0,MATRIXFLAG); MVI(0,TABLEFLAG);                   00026700
01 0E6C   06 9BB4    0880      MVI(0,READFLAG); MVI(0,EOF);  MVI(0,RIGHTFLAG);                             00026800
01 0E78   06 9BB4    0881      MVI(0,LEFTFLAG); MVI(0,EOPROD);  R1 := 0; MSTART := R1;                     00026900
01 0E88   06 9BB4    0882      MVI(0,CHECKFLAG);  PAGE;  MVI(0,SYMBOLFLAG);                                00027000
01 0E9A   06 9BB4    0883      GOTO LOOP;                                                                  00027100
01 0E9E   06 9BB4    0884    END;                                                                          00027200
01 0E9E   06 9BB4    0885                                                                                  00027300
01 0E9E   06 9BB4    0886    END.                                                                          00027400

                   SEGMENT 06  NAME = SEG#06      LENGTH = 9BB8  BASE REG = 11

                   SEGMENT 00  NAME = SEG#00      LENGTH = 1F2C  BASE REG = 13

                   SEGMENT 01  NAME = SEG#01      LENGTH = 0FA8  BASE REG = 15
```

LOAD MAP

-SEGMENT-	-ORIGIN-	-LENGTH-
READ	0FB050	000000
WRITE	0FB0C0	000000
DUMP	0F9ED0	000000
PAGE	0FB124	000000
PUNCH	0FB12A	000000
SEG#07	111800	0007F8
SEG#01	1232D8	000FA8
SEG#08	111FF8	000E00
SEG#09	112DF8	000470
SEG#11	113268	000230
SEG#10	113498	004358
SEG#06	1177F0	009BB8
SEG#00	1213A8	001F30

```
<K REG>       ::= <ID>
<T CELL ID>   ::= <ID>
<PROC ID>     ::= <ID>
<FUNC ID>     ::= <ID>
<T CELL>      ::= <T CELL ID>
                  <T CELL1>  )
                  <T CELL2>  )
<T CELL1>     ::= <T CELL2>  <ARITH OP>  <T NUMBER>
                  <T CELL3>  <T NUMBER>
<T CELL2>     ::= <T CELL3>  <K REG>
<T CELL3>     ::= <T CELL ID>  (
<UNARY OP>    ::= ABS
                  NEG
                  NEG  ABS
<ARITH OP>    ::= +
                  -
                  *
                  /
                  +  +
                  -  -
<LOG OP>      ::= AND
                  OR
                  XOR
<K REG ASS>   ::= <K REG>  :=  <T CELL>
                  <K REG>  :=  <T NUMBER>
                  <K REG>  :=  <STRING>
                  <K REG>  :=  <K REG>
                  <K REG>  :=  <UNARY OP>  <T CELL>
                  <K REG>  :=  <UNARY OP>  <T NUMBER>
                  <K REG>  :=  <UNARY OP>  <K REG>
                  <K REG>  :=  @  <T CELL>
                  <K REG ASS>  <ARITH OP>  <T CELL>
                  <K REG ASS>  <ARITH OP>  <T NUMBER>
                  <K REG ASS>  <ARITH OP>  <K REG>
                  <K REG ASS>  <LOG OP>  <T CELL>
                  <K REG ASS>  <LOG OP>  <T NUMBER>
                  <K REG ASS>  <LOG OP>  <K REG>
                  <K REG ASS>  <SHIFT OP>  <T NUMBER>
                  <K REG ASS>  <SHIFT OP>  <K REG>
<FUNC1>       ::= <FUNC2>  <T NUMBER>
                  <FUNC2>  <K REG>
                  <FUNC2>  <T CELL>
                  <FUNC2>  <STRING>
<FUNC2>       ::= <FUNC ID>  (
                  <FUNC1>  ,
<CASE SEQ>    ::= CASE  <K REG>  OF  BEGIN
                  <CASE SEQ>  <STATEMENT>  ;
<SIMPLE ST>   ::= <T CELL>  :=  <K REG>
                  <K REG ASS>
                  NULL
                  <PROC ID>
                  <FUNC ID>
                  <FUNC1>  )
                  <CASE SEQ>  END
                  <BLOCKBODY>  END
<REL OP>      ::= <
                  =
                  >
                  <  =
                  >  =
```

```
                        ¬  =
<NOT>           ::= ¬
<CONDITION>     ::= <K REG>  <REL OP>  <T CELL>
                    <K REG>  <REL OP>  <T NUMBER>
                    <K REG>  <REL OP>  <K REG>
                    <K REG>  <REL OP>  <STRING>
                    OVERFLOW
                    <REL OP>
                    <T CELL>
                    <NOT>  <T CELL>
<COMP COND>     ::= <CONDITION>
                    <COMP AOR>  <CONDITION>
<COMP AOR>      ::= <COMP COND>  AND
                    <COMP COND>  OR
<COND THEN>     ::= <COMP COND>  THEN
<GOTO ST*>      ::= GOTO  <ID>
<GOTO ST>       ::= <GOTO ST*>
<TRUE PART>     ::= <SIMPLE ST>  ELSE
                    <GOTO ST*>  ELSE
<WHILE>         ::= WHILE
<COND DO>       ::= <COMP COND>  DO
<ASS STEP>      ::= <K REG ASS>  STEP  <T NUMBER>
<LIMIT>         ::= UNTIL  <K REG>
                    UNTIL  <T CELL>
                    UNTIL  <T NUMBER>
<DO>            ::= DO
<STATEMENT->    ::= <SIMPLE ST>
                    IF  <COND THEN>  <STATEMENT->
                    IF  <COND THEN>  <GOTO ST>
                    IF  <COND THEN>  <TRUE PART>  <STATEMENT->
                    IF  <COND THEN>  <TRUE PART>  <GOTO ST>
                    <WHILE>  <COND DO>  <STATEMENT*>
                    IF  <COND THEN>  <TRUE PART>  <GOTO ST>
                    <WHILE>  <COND DO>  <STATEMENT*>
                    FOR  <ASS STEP>  <LIMIT>  <DO>  <STATEMENT*>
<STATEMENT*>    ::= <STATEMENT->
                    <GOTO ST>
<STATEMENT>     ::= <STATEMENT*>
<SI T TYPE>     ::= SHORT  INTEGER
                    INTEGER
                    LOGICAL
                    REAL
                    LONG  REAL
                    BYTE
                    CHARACTER
<T TYPE>        ::= <SI T TYPE>
                    ARRAY  <T NUMBER>  <SI T TYPE>
<FILL>          ::= <STRING>
                    @  <T CELL>
                    <T NUMBER>
                    <REP LIST3>
<REP LIST1>     ::= <T NUMBER>  (
                    (
                    <REP LIST2>  ,
<REP LIST2>     ::= <REP LIST1>  <FILL>
<REP LIST3>     ::= <REP LIST2>  )
<T DECL1>       ::= <T TYPE>  <ID>
                    <T DECL2>  <ID>
<T DECL2>       ::= <T DECL4>  ,
<T DECL3>       ::= <T DECL1>  =
<T DECL4>       ::= <T DECL1>
                    <T DECL3>  <FILL>
```

```
121     <FUNC DC1>    ::= FUNCTION
122                       <FUNC DC7>  ,
123     <FUNC DC2>    ::= <FUNC DC1>  <ID>
124     <FUNC DC3>    ::= <FUNC DC2>  (
125     <FUNC DC4>    ::= <FUNC DC3>  <T NUMBER>
126     <FUNC DC5>    ::= <FUNC DC4>  ,
127     <FUNC DC6>    ::= <FUNC DC5>  <T NUMBER>
128     <FUNC DC7>    ::= <FUNC DC6>  )
129     <SYN DC1>     ::= <T TYPE>  <ID>  SYN
130                       <SI T TYPE>  REGISTER  <ID>  SYN
131                       <SYN DC3>  <ID>  SYN
132     <SYN DC2>     ::= <SYN DC1>  <T CELL>
133                       <SYN DC1>  <T NUMBER>
134                       <SYN DC1>  <K REG>
135     <SYN DC3>     ::= <SYN DC2>  ,
136     <PROC HD1>    ::= PROCEDURE  <ID>
137     <PROC HD2>    ::= <PROC HD1>  (
138     <PROC HD3>    ::= <PROC HD2>  <K REG>
139     <PROC HD4>    ::= <PROC HD3>  )
140     <PROC HD5>    ::= <PROC HD4>  ;
141     <PROC HD6>    ::= <PROC HD5>
142                       GLOBAL  <PROC HD5>
143                       EXTERNAL  <PROC HD5>
144                       SEGMENT  <PROC HD5>
145     <DSEG TYPE>   ::= GLOBAL  DATA  <ID>
146                       EXTERNAL  DATA  <ID>
147                       COMMON  DATA  <ID>
148                       COMMON
149                       SEGMENT
150                       DUMMY
151     <DECL>        ::= <T DECL4>
152                       <FUNC DC7>
153                       <SYN DC2>
154                       <PROC HD6>  <STATEMENT*>
155                       <DSEG TYPE>  BASE  <K REG>
156                       CLOSE  BASE
157     <LABEL DEF>   ::= <ID>  :
158     <BLOCKHEAD>   ::= BEGIN
159                       <BLOCKHEAD>  <DECL>  ;
160     <BLOCKBODY>   ::= <BLOCKHEAD>
161                       <BLOCKBODY>  <STATEMENT>  ;
162                       <BLOCKBODY>  <LABEL DEF>
163     <PROGRAM->    ::= .
164                       .  GLOBAL  <PROC HD5>
165     <PROGRAM*>    ::= <PROGRAM->  <STATEMENT*>
166     <PROGRAM>     ::= <PROGRAM*>  .
```

NONTERMINAL SYMBOLS

1	<K REG>	2	<T CELL ID>	3	<PROC ID>	4	<FUNC ID>	5	<T CELL>
6	<T CELL1>	7	<T CELL2>	8	<T CELL3>	9	<UNARY OP>	10	<ARITH OP>
11	<LOG OP>	12	<K REG ASS>	13	<FUNC1>	14	<FUNC2>	15	<CASE SEQ>
16	<SIMPLE ST>	17	<REL OP>	18	<NOT>	19	<CONDITION>	20	<COMP COND>
21	<COMP AOR>	22	<COND THEN>	23	<GOTO ST>	24	<GOTO ST*>	25	<TRUE PART>
26	<WHILE>	27	<COND DO>	28	<ASS STEP>	29	<LIMIT>	30	<DO>
31	<STATEMENT->	32	<STATEMENT*>	33	<STATEMENT>	34	<SI T TYPE>	35	<T TYPE>
36	<FILL>	37	<REP LIST1>	38	<REP LIST2>	39	<REP LIST3>	40	<T DECL1>
41	<T DECL2>	42	<T DECL3>	43	<T DECL4>	44	<FUNC DC1>	45	<FUNC DC2>
46	<FUNC DC3>	47	<FUNC DC4>	48	<FUNC DC5>	49	<FUNC DC6>	50	<FUNC DC7>
51	<SYN DC1>	52	<SYN DC2>	53	<SYN DC3>	54	<PROC HD1>	55	<PROC HD2>
56	<PROC HD3>	57	<PROC HD4>	58	<PROC HD5>	59	<PROC HD6>	60	<DSEG TYPE>
61	<DECL>	62	<LABEL DEF>	63	<BLOCKHEAD>	64	<BLOCKBODY>	65	<PROGRAM->
66	<PROGRAM*>	67	<PROGRAM>						

TERMINAL SYMBOLS

68	<ID>	69	<T NUMBER>	70	<STRING>	71	<SHIFT OP>	72	;
73	+	74	-	75	=	76	<	77	>
78	¬	79	(	80	)	81	*	82	/
83	:	84	,	85	@	86	.	87	:=
88	NEG	89	DO	90	IF	91	OF	92	OR
93	ABS	94	AND	95	END	96	FOR	97	SYN
98	XOR	99	BASE	100	BYTE	101	CASE	102	DATA
103	ELSE	104	GOTO	105	LONG	106	NULL	107	REAL
108	STEP	109	THEN	110	ARRAY	111	BEGIN	112	CLOSE
113	DUMMY	114	SHORT	115	UNTIL	116	WHILE	117	COMMON
118	GLOBAL	119	INTEGER	120	LOGICAL	121	SEGMENT	122	EXTERNAL
123	FUNCTION	124	OVERFLOW	125	REGISTER	126	CHARACTER	127	PROCEDURE

IDENTICAL RIGHT PART OCCURRENCES

LINE 1 SAME AS LINE 2
LINE 2 SAME AS LINE 3
LINE 3 SAME AS LINE 4

NO PRECEDENCE VIOLATION DETECTED

PRECEDENCE FUNCTIONS

1	<K REG>	8	5
2	<T CELL ID>	9	6
3	<PROC ID>	3	4
4	<FUNC ID>	9	4
5	<T CELL>	8	5
6	<T CELL1>	3	6
7	<T CELL2>	3	6
8	<T CELL3>	5	6
9	<UNARY OP>	5	5
10	<ARITH OP>	5	3
11	<LOG OP>	5	3
12	<K REG ASS>	3	4
13	<FUNC1>	3	4
14	<FUNC2>	5	4
15	<CASE SEQ>	1	4
16	<SIMPLE ST>	2	4
17	<REL OP>	5	8
18	<NOT>	5	3
19	<CONDITION>	5	2
20	<COMP COND>	4	2
21	<COMP AOR>	2	2
22	<COND THEN>	3	1
23	<GOTO ST>	2	3
24	<GOTO ST*>	2	4
25	<TRUE PART>	3	3
26	<WHILE>	1	4
27	<COND DO>	2	1
28	<ASS STEP>	1	1
29	<LIMIT>	1	1
30	<DO>	2	1
31	<STATEMENT->	2	3
32	<STATEMENT*>	2	2
33	<STATEMENT>	1	1
34	<SI T TYPE>	7	9
35	<T TYPE>	6	8
36	<FILL>	4	1
37	<REP LIST1>	1	2
38	<REP LIST2>	3	2
39	<REP LIST3>	4	2
40	<T DECL1>	9	8
41	<T DECL2>	6	8
42	<T DECL3>	1	8
43	<T DECL4>	3	8
44	<FUNC DC1>	6	8
45	<FUNC DC2>	9	8
46	<FUNC DC3>	5	8
47	<FUNC DC4>	3	8
48	<FUNC DC5>	5	8
49	<FUNC DC6>	3	8
50	<FUNC DC7>	3	8
51	<SYN DC1>	5	8
52	<SYN DC2>	3	8
53	<SYN DC3>	6	8

54	<PROC HD1>	9	9
55	<PROC HD2>	5	9
56	<PROC HD3>	3	9
57	<PROC HD4>	1	9
58	<PROC HD5>	7	8

59	<PROC HD6>	2	8
60	<DSEG TYPE>	1	8
61	<DECL>	1	7
62	<LABEL DEF>	7	1
63	<BLOCKHEAD>	7	4
64	<BLOCKBODY>	1	4
65	<PROGRAM->	2	1
66	<PROGRAM*>	1	1
67	<PROGRAM>	1	1
68	<ID>	10	6
69	<T NUMBER>	9	5
70	<STRING>	8	5
71	<SHIFT OP>	5	3
72	;	11	1
73	+	7	7
74	-	7	7
75	=	10	9
76	<	9	9
77	>	9	9
78	¬	9	9
79	(	10	9
80	)	9	3
81	*	7	4
82	/	7	4
83	:	7	10
84	,	10	3
85	@	5	5
86	.	8	1
87	:=	5	8
88	NEG	7	6
89	DO	7	4
90	IF	1	4
91	OF	4	8
92	OR	10	4
93	ABS	7	7
94	AND	10	4
95	END	3	1
96	FOR	1	4
97	SYN	7	10
98	XOR	7	4
99	BASE	5	1
100	BYTE	8	10
101	CASE	5	4
102	CATA	6	8
103	ELSE	7	2
104	GOTO	6	4
105	LONG	10	10
106	NULL	3	4
107	REAL	8	10
108	STEP	5	3
109	THEN	7	4
110	ARRAY	5	8
111	BEGIN	11	4
112	CLOSE	1	8
113	DUMMY	2	8
114	SHORT	10	10
115	UNTIL	5	2
116	WHILE	10	4
117	COMMON	8	8
118	GLOBAL	8	8

119	INTEGER	8	10
120	LOGICAL	8	10
121	SEGMENT	8	8
122	EXTERNAL	8	8
123	FUNCTION	7	8
124	OVERFLOW	5	3
125	REGISTER	6	7
126	CHARACTER	8	10
127	PROCEDURE	6	9

```
ARRAY 128 SHORT INTEGER F       =
   (0,8,9,3,9,8,3,3,4(5),3,3,5,1,2,3(5),4,2,3,2,2,3,1,2,1,1,3(2),1,
   7,6,4,1,3,4,9,6,1,3,6,9,5,3,5,3,3,5,3,6,9,5,3,1,7,2,1,1,7,7,1,2,
   1,1,10,9,8,5,11,7,7,10,3(9),10,9,3(7),10,5,8,5,7,7,1,4,10,7,10,3,
   1,7,7,5,8,5,6,7,6,10,3,8,5,7,5,11,1,2,10,5,10,6(8),7,5,6,8,6);

ARRAY 128 SHORT INTEGER G       =
   (0,5,6,4,4,5,3(6),5,3,3,5(4),8,3,3(2),1,3,4,3,4,4(1),3,2,1,9,8,1,
   3(2),14(8),4(9),3(8),7,1,4,4,3(1),6,5,5,3,1,7,7,5(9),3,4,4,10,3,
   5,1,8,6,4,4,8,4,7,4,1,4,10,4,1,10,4,8,2,4,10,4,10,3,4,8,4,8,8,10,
   2,4,8,8,10,10,3(8),3,7,10,9);

ARRAY 128 SHORT INTEGER MTB     =
   (0,1,78,88,93,103,114,120,132,143,144,145,146,205,216,237,249,
   259,264,270,275,296,302,303,308,318,319,326,327,328,329,330,335,
   340,341,353,365,366,372,383,388,398,404,410,420,426,432,438,444,
   450,456,466,482,492,499,505,511,517,523,528,534,541,542,543,554,
   571,577,578,579,601,611,616,617,618,628,638,643,653,663,673,678,
   679,684,689,690,691,697,708,709,719,724,751,752,757,762,767,768,
   777,778,783,784,789,797,798,799,805,811,816,821,822,823,830,835,
   841,846,852,868,873,884,896,901,906,916,928,933,938,939,944);

ARRAY 950 BYTE PRTB     =
   (255,2,10,174,2,24,24,2,138,174,2,24,25,2,140,174,2,24,26,2,2,
   174,2,24,27,3,10,18,174,2,24,28,3,138,18,174,2,24,29,3,2,18,174,
   2,24,30,3,10,170,174,2,24,31,2,10,34,2,38,63,2,138,34,2,38,64,2,
   2,34,2,38,65,2,140,34,2,38,66,255,0,4,10,5,1,158,4,16,11,255,0,6,
   32,51,255,1,158,8,28,44,0,8,32,52,255,2,2,174,10,32,48,0,10,38,
   69,255,1,160,12,10,6,255,1,160,14,10,7,2,138,20,14,12,8,255,1,
   138,16,12,9,1,2,16,14,10,4(255),2,10,20,24,24,32,2,138,20,24,24,
   33,2,2,20,24,24,34,2,10,22,24,24,35,2,138,22,24,24,36,2,2,22,24,
   24,37,2,138,142,24,24,38,2,2,142,24,24,39,0,24,32,49,2,138,216,
   24,56,82,255,1,168,26,28,45,1,160,26,32,53,255,1,138,28,26,40,1,
   2,28,26,41,1,10,28,26,42,1,140,28,26,43,255,2,144,66,30,30,47,1,
   190,30,32,54,255,1,206,32,50,78,0,32,62,87,255,0,34,38,68,255,1,
   10,36,38,70,255,0,38,40,71,255,1,188,40,42,73,1,184,40,42,74,1,
   218,40,44,75,1,178,40,54,81,255,1,38,42,40,72,2(255),0,46,64,95,
   255,0,48,46,77,1,206,48,50,79,2(255),2,64,54,52,62,92,5(255),0,
   62,64,94,255,0,64,66,96,2(255),0,68,70,104,3,194,136,250,68,102,
   130,255,1,136,70,80,115,2,194,136,70,102,129,2(255),1,72,74,76,
   113,255,1,168,76,74,112,1,160,76,78,114,255,0,78,72,109,255,1,
   150,80,84,118,0,80,86,119,255,1,136,82,80,116,255,1,72,84,86,120,
   255,1,168,86,82,117,0,86,122,151,255,1,136,88,90,123,255,1,158,
   90,92,124,255,1,138,92,94,125,255,1,168,94,96,126,255,1,138,96,
   98,127,255,1,160,98,100,128,255,1,168,100,88,122,0,100,122,152,
   255,1,10,102,104,132,1,138,102,104,133,1,2,102,104,134,255,1,168,
   104,106,135,0,104,122,153,255,2,194,136,106,102,131,255,1,158,
   108,110,137,255,1,2,110,112,138,255,1,160,112,114,139,255,1,144,
   114,116,140,255,0,116,118,141,255,1,64,118,122,154,255,2,2,198,
   120,122,155,3(255),2,144,122,2(126),159,0,126,128,160,255,1,190,
   128,32,55,2,144,66,2(128),161,1,124,2(128),162,255,1,64,130,132,
   165,3(255),0,136,2,1,0,136,4,2,0,136,6,3,0,136,8,4,1,166,136,124,
   157,255,0,138,72,108,1,158,138,74,110,255,0,140,72,106,3(255),0,
   146,20,15,1,2(146),20,19,255,0,148,20,16,1,2(148),20,20,255,0,
   150,34,57,255,0,152,34,56,1,150,152,34,59,255,0,154,34,58,1,150,
   154,34,60,255,1,150,156,34,61,0,156,36,62,255,0,158,74,111,
   2(255),0,162,20,17,255,0,164,20,18,3(255),1,10,170,72,107,255,0,
   172,130,163,2,116,236,172,130,164,2(255),0,176,18,13,1,186,176,
   18,14,255,0,178,60,86,255,2,62,44,180,62,88,2,46,44,180,62,89,3,
   62,50,44,180,62,90,3,46,50,44,180,62,91,2(255),0,184,22,22,255,0,
```

```
186,18,12,255,0,188,22,21,2(255),4,64,60,58,56,192,62,93,2(255),
0,196,22,23,2(255),0,200,68,102,255,3,222,182,2,202,30,46,3(255),
1,136,208,48,76,255,1,214,210,68,101,255,0,212,32,50,255,0,214,
68,100,3(255),2,68,138,220,70,105,255,0,222,126,158,255,1,198,
224,122,156,255,0,226,120,150,255,1,238,228,68,97,255,1,2,230,58,
83,1,10,230,58,84,1,138,230,58,85,255,0,232,52,80,255,2,136,204,
234,120,147,0,234,120,148,255,1,116,236,118,142,2,136,204,236,
120,145,255,0,238,68,98,255,0,240,68,99,255,1,116,242,118,144,0,
242,120,149,255,1,116,244,118,143,2,136,204,244,120,146,255,0,
246,88,121,255,0,248,38,67,2(255),0,252,68,103,255,1,136,254,108,
136,255);
```

JOB STATISTICS

04 DEC 69 EHSPL360 GP=CG   SEQ NO=273 CLASS=E PRTY=8

****

CLOCK TIMES (HH:MM:SS)

ON RDR	14:15:14	BEGIN EXEC	14:17:14	ON PRT	14:17:38
OFF RDR	14:16:44	END EXEC	14:17:34	OFF PRT	14:20:54
RDR TIME	0:01:29	EXEC TIME	0:00:19	PRT TIME	0:03:16

****

RESOURCES USED		UNITS	CHARGE FORMULA
CARDS READ		1,200	R = MAX((N*0.075 - 300.0), 0)
CARDS PUNCHED		0	
LINES PRINTED		1,383	L = N * 3.75E-03
7-TRACK	ACCESSES	0	N7
	ERRORS	N.A.	
	MOUNTS	N.A.	M7
9-TRACK	ACCESSES	0	N9
	ERRORS	N.A.	
	MOUNTS	N.A.	M9
D.A.	ACCESSES	46	ND
	MOUNTS	N.A.	MD, M = (M7 + M9)*60 + MD*600
OTHER	ACCESSES	0	NM, F = (N7+N9+ND+NM) * 0.075
CORE	AMOUNT	300K	CF = N / 150
CORE	TIME USED	0:00:06	G = (8/9)*CF**2 - (2/3)*CF + (7/9)
CPU	TIME USED	0:00:02.67	TSEC
SVC	WAIT	54	
	OTHER	747 (NOT INCL I/O)	

****

TOTAL UNITS = 23.54 UNITS = G*(TSEC+F) + R + L + M

****

H A S P S Y S T E M L O G

*23.49.36 JOB 2326 IEF233A M 0C2,WIR001,TOTAPE
*23.50.2C JOB 2326 IEF280I K 0C2,WIR001,TOTAPE

```
//
//TOTAPE  JOB  (T123***,914,2,10,,,,T),' ED SATTERTHWAITE ',MSGLEVEL=1    JOB2326
//JOBLIB DD DSNAME=T123.PLLIB,UNIT=2314,VOLUME=SER=SYS11,                X
//             DISP=(OLD,PASS)
//PL360 EXEC PGM=PL360,PARM='DECK,NOLOAD'
//SYSPRINT DD SYSOUT=A,DCB=BLKSIZE=133
//SYSPUNCH DD DSNAME=AWXCMPA1,UNIT=TAPE9,VOLUME=SER=WIR001,              X
//             LABEL=(5,SL),DISP=(NEW,KEEP),                             X
//             DCB=(BLKSIZE=2000,LRECL=80,RECFM=FB)
//SYSIN  DD *
IEF236I ALLOC. FOR TOTAPE   PL360
IEF237I JOBLIB   ON 391
IEF237I SYSPUNCH ON 0C2
IEF237I SYSIN    ON 00C
```

```
01 0000  00 0000  0001    COMMENT
01 0000  00 0000  0002                    STANFORD ALGOL W COMPILER
01 0000  00 0000  0003                  PHASE A - SCAN AND PARSE PASSES
01 0000  00 0000  0004                       JULY 1969 VERSION                ;
01 0000  00 0000  0005
01 0000  00 0000  0006  GLOBAL PROCEDURE AWXCMPA1(R14);
01 0000  00 0000  0007  BEGIN
06 0000  00 0000  0008   FUNCTION BALR(1,#0500);
06 0000  00 0000  0009   STM(R14,R12,B13(12));  R4 := R1; R5 := R13;
06 0008  00 0000  0010   R0 := 4096; R1 := R0;  R3 := B4(20); BALR(R2,R3);  COMMENT GETMAIN;
06 0014  00 0000  0011   R13 := R0;  COMMENT ESTABLISH 4K INITIAL DATA SEGMENT, BASE R13;
06 0016  00 0000  0012
06 0016  00 0000  0013  BEGIN
06 0016  00 0000  0014  DUMMY BASE R13;   COMMENT  SHARED DATA SEGMENT;
06 0016  07 0000  0015   LOGICAL SYSINF, OLDSAVE, NEWSAVE;  ARRAY 15 LOGICAL SAVEAREA;
06 0016  07 0048  0016   ARRAY 16 INTEGER XFERVECTOR;   COMMENT SUPV ENTRY ADDRESSES;
06 0016  07 0088  0017    ARRAY 2 INTEGER COMMLIM SYN XFERVECTOR(0);
06 0016  07 0088  0018    INTEGER APUTLINE SYN XFERVECTOR(8);       COMMENT WRITE ENTRY;
06 0016  07 0088  0019     BYTE CARRCONT SYN APUTLINE(0);           COMMENT PRINT CONTROL CHAR;
06 0016  07 0088  0020    INTEGER AGETCARD SYN XFERVECTOR(12);      COMMENT READ ENTRY;
06 0016  07 0088  0021    INTEGER APUTCARD SYN XFERVECTOR(16);      COMMENT PUNCH ENTRY;
06 0016  07 0088  0022    INTEGER AGETMAIN SYN XFERVECTOR(20);      COMMENT GETMAIN ENTRY;
06 0016  07 0088  0023    INTEGER AFREEMAIN SYN XFERVECTOR(24);     COMMENT FREEMAIN ENTRY;
06 0016  07 0088  0024    INTEGER AGETTIME SYN XFERVECTOR(28);      COMMENT GETTIME ENTRY;
06 0016  07 0088  0025    INTEGER ARUNID SYN XFERVECTOR(60);        COMMENT ADDRESS OF SYSTEM ID;
06 0016  07 0088  0026   ARRAY 5 INTEGER BASESAVE;    COMMENT DATA BASES FOR PASS 2;
06 0016  07 009C  0027   SHORT INTEGER LINENO, PAGENO;
06 0016  07 00A0  0028   LONG REAL PKDEC;             COMMENT USED WITH CVD;
06 0016  07 00A8  0029   ARRAY 132 BYTE HEADING;      COMMENT PAGE HEADER;
06 0016  07 012C  0030
06 0016  07 012C  0031   PROCEDURE DUMP(R14); NULL;    COMMENT REPLACE FOR DEBUGGING;
06 001C  07 012C  0032
06 001C  07 012C  0033   PROCEDURE PRINT(R1);
06 001C  07 012C  0034    COMMENT DETERMINES REQUIRED CONTROL CHARACTER, PRINTS LINE AT (R0);
06 001C  07 012C  0035    BEGIN  ARRAY 4 LOGICAL SAVE03;  STM(R0,R3,SAVE03);
06 0020  07 013C  0036      R1 := R1-R1; IC(R1,CARRCONT);  R2 := LINENO + 1;  R3 := APUTLINE;
06 0032  07 013C  0037      IF R1 = "0" THEN
06 003A  07 013C  0038      BEGIN  IF R2 >= 60 THEN R1 := "1" ELSE R2 := R2 + 1;
06 004E  07 013C  0039      END;
06 004E  07 013C  0040      IF R1 = "1" THEN
06 0056  07 013C  0041      BEGIN  R0 := PAGENO + 1; PAGENO := R0; CVD(R0,PKDEC);
06 0066  07 013C  0042        MVC(3,HEADING(117),#40202120); ED(3,HEADING(117),PKDEC(6));
06 0072  07 013C  0043        R0 := @HEADING;  BALR(R2,R3);  R0 := SAVE03(0);
06 007C  07 013C  0044        R2 := 3;  R1 := "0";
06 0084  07 013C  0045      END;
06 0084  07 013C  0046      LINENO := R2;
06 0088  07 013C  0047      IF R2 = 60 THEN MVI("1",CARRCONT) ELSE MVI(" ",CARRCONT);
06 009C  07 013C  0048      BALR(R2,R3); COMMENT LINK TO WRITE;   LM(R0,R3,SAVE03);
06 00A2  07 013C  0049    END;
06 00A4  07 013C  0050
06 00A4  07 013C  0051   SEGMENT PROCEDURE PASS1(R10);
06 00A4  07 013C  0052   BEGIN  COMMENT  PASS 1 -- SCANNER;
08 0000  07 013C  0053
08 0000  07 013C  0054   DUMMY BASE R13;  COMMENT WORK AREA SHARED WITH PASS 2;
08 0000  09 0000  0055    ARRAY 18 LOGICAL SAVEINFO;   COMMENT REPEAT PRECEDING DECLARATIONS;
08 0000  09 0048  0056    ARRAY 16 LOGICAL XFERVECTOR;
08 0000  09 0088  0057     INTEGER NTBASE SYN XFERVECTOR(0);  COMMENT COMMON BASE ADDRESS;
08 0000  09 0088  0058    ARRAY 5 LOGICAL BASESAVE;
```

```
08 0000  09 009C  0059   SHORT INTEGER LINENO, PAGENO;
08 0000  09 00A0  0060   LONG REAL PKDEC;
08 0000  09 00A8  0061   ARRAY 132 BYTE HEADING;  COMMENT PAGE HEADER;
08 0000  09 012C  0062   ARRAY 4 LOGICAL PRINTEMP;
08 0000  09 013C  0063
08 0000  09 013C  0064  COMMENT   * * *  LOCAL VARIABLE DECLARATIONS BEGIN HERE  * * * ;
08 0000  09 013C  0065   INTEGER SAVE14, RETADDR;
08 0000  09 0144  0066
08 0000  09 0144  0067   COMMENT REGISTER ALLOCATION
08 0000  09 0144  0068         R0-R4 GENRAL PURPOSE
08 0000  09 0144  0069            R5 CONSTANT 1
08 0000  09 0144  0070            R6 INDEX TO PASS TWO OUTPUT
08 0000  09 0144  0071            R7 PROCEDURE LINKAGE AND CURRENT AND NEXT BUF BASE
08 0000  09 0144  0072            R8 NEXT CHARACTER
08 0000  09 0144  0073            R9 INDEX TO INPUT BUFFER
08 0000  09 0144  0074         RA-RB PROCEDURE LINKAGE
08 0000  09 0144  0075         RC-RD DATA BASE REGISTERS
08 0000  09 0144  0076            RF PROGRAM BASE REGISTER     ;
08 0000  09 0144  0077
08 0000  09 0144  0078   LOGICAL ALGOLDATAORG SYN B13(72); COMMENT ALGOLRUN DATA START;
08 0000  09 0144  0079   ARRAY  9  INTEGER RESADDR,CODEADDR;
08 0000  09 018C  0080   INTEGER RFRCLISTBASE, IDDIRBASE, IDLISTBASE, PRBASE, PROGLIMIT;
08 0000  09 01A0  0081   INTEGER SAVERB, SAVERB1, SAVER4, SAVER10, SAVEN;
08 0000  09 01B4  0082   ARRAY 3 LOGICAL SAVE13; COMMENT USED BY FETCHCARD;
08 0000  09 01C0  0083   ARRAY 4 LOGICAL FLAGS;
08 0000  09 01D0  0084    BYTE LISTFLAG SYN FLAGS(0),    TRACEFLAG SYN FLAGS(1),
08 0000  09 01D0  0085         NOPASSTWO SYN FLAGS(2),   EOF SYN FLAGS(3),
08 0000  09 01D0  0086         ENDDOT SYN FLAGS(4),      EFLAG SYN FLAGS(5),
08 0000  09 01D0  0087         NEWCARD SYN FLAGS(6),
08 0000  09 01D0  0088         REFERFLAG SYN FLAGS(7),   SIGN SYN FLAGS(8),
08 0000  09 01D0  0089         EXPOSIGN SYN FLAGS(9),    TYPEFLAG SYN FLAGS(10),
08 0000  09 01D0  0090         IMAGFLAG SYN FLAGS(11),   FARRAYFLAG SYN FLAGS(12),
08 0000  09 01D0  0091         LAST SYN FLAGS(13);
08 0000  09 01D0  0092   LOGICAL PARD;
08 0000  09 01D4  0093    BYTE PROCFLAG SYN PARD;
08 0000  09 01D4  0094    BYTE ARRAYFLAG SYN PARD(1);
08 0000  09 01D4  0095    BYTE RECORDFLAG SYN PARD(2);
08 0000  09 01D4  0096    BYTE DECLARFLAG SYN PARD(3);
08 0000  09 01D4  0097   ARRAY  132  BYTE INBUF;
08 0000  09 0258  0098      ARRAY 132 BYTE OUTPUT SYN INBUF;  COMMENT USED FOR ERROR LOG ONLY;
08 0000  09 0258  0099   BYTE LINEHOLD, BEGINSET;   COMMENT LISTING CONTROL;
08 0000  09 025A  0100   SHORT INTEGER NESTLEVEL;   COMMENT BEGIN/END NESTING LEVEL;
08 0000  09 025C  0101   SHORT INTEGER SYMBOLINDEX, IDDIRINDEX, IDLISTINDEX;
08 0000  09 0262  0102   SHORT INTEGER BLLIMIT, IDDIRLIMIT, IDLISTLIMIT;
08 0000  09 0268  0103   SHORT INTEGER STACKINDEX;  COMMENT POINTER TO TOP OF STACK TABLE;
08 0000  09 026A  0104   SHORT INTEGER SCRATCHINDEX;   COMMENT POINTER TO TOP OF SCRATCH TABLE;
08 0000  09 026C  0105   LOGICAL TYPE;      COMMENT FIELD CONTAINING TYPE INFO OF ID BEING
08 0000  09 0270  0106                        PROCESSED;
08 0000  09 0270  0107   SHORT INTEGER BLOCKNO;
08 0000  09 0272  0108   SHORT INTEGER DIMCNT;  INTEGER FARRAYPNT;
08 0000  09 0278  0109   SHORT INTEGER ERRORCOUNT, ERRORLIMIT;
08 0000  09 027C  0110   ARRAY  50  INTEGER ERRORS;
08 0000  09 0344  0111   INTEGER OUTCHAR,SAVEADD;
08 0000  09 034C  0112   SHORT INTEGER CARDCOUNT;
08 0000  09 034E  0113   INTEGER CURBUFBASE,NEXTBUFBASE;
08 0000  09 0358  0114   SHORT INTEGER RECORDNO;
08 0000  09 035A  0115   ARRAY 2 LONG REAL FIRSTNO;
08 0000  09 0370  0116    INTEGER FIRSTNOI SYN FIRSTNO;  REAL FIRSTNOR SYN FIRSTNO;
```

```
08 0000  09 0370  0117    LONG REAL FIRSTNOLR SYN FIRSTNO;
08 0000  09 0370  0118   INTEGER NUMPOS,IMAGPOS;
08 0000  09 0378  0119   LONG REAL FCON1, FCON2;
08 0000  09 0380  0120    INTEGER FCON1LOW SYN FCON1 (4);
08 0000  09 0388  0121   LONG REAL REALPART, SAVERL;
08 0000  09 0398  0122    INTEGER SAVE SYN SAVERL;
08 0000  09 0398  0123    INTEGER REALPARTR SYN REALPART;
08 0000  09 0398  0124    INTEGER REALPARTR1 SYN REALPART(4);
08 0000  09 0398  0125   SHORT INTEGER SIMTYPEINFO;
08 0000  09 039A  0126   SHORT INTEGER RLISTPOINT;
08 0000  09 039C  0127   INTEGER CHAIN,HCODE;
08 0000  09 03A4  0128   ARRAY 160 SHORT INTEGER HASHID;
08 0000  09 04E4  0129   ARRAY  66  LONG REAL SYMBUFFERS;
08 0000  09 06F8  0130   ARRAY  66  LONG REAL SYMBOLBUFFERS SYN B7;
08 0000  09 06F8  0131    BYTE CURBUF SYN SYMBOLBUFFERS(4);
08 0000  09 06F8  0132    SHORT INTEGER CURBUFINT SYN CURBUF;
08 0000  09 06F8  0133    SHORT INTEGER CURBUFTYPE SYN SYMBOLBUFFERS;
08 0000  09 06F8  0134    SHORT INTEGER CURBUFCOUNT SYN SYMBOLBUFFERS(2);
08 0000  09 06F8  0135    INTEGER CURBUFI SYN CURBUF;
08 0000  09 06F8  0136    REAL CURBUFR SYN CURBUF;
08 0000  09 06F8  0137    LONG REAL CURBUFLR SYN CURBUF(4);
08 0000  09 06F8  0138    BYTE NEXTBUF SYN SYMBOLBUFFERS(4);
08 0000  09 06F8  0139    SHORT INTEGER NEXTBUFINT SYN NEXTBUF;
08 0000  09 06F8  0140    SHORT INTEGER NEXTBUFTYPE SYN SYMBOLBUFFERS;
08 0000  09 06F8  0141    SHORT INTEGER NEXTBUFCOUNT SYN SYMBOLBUFFERS(2);
08 0000  09 06F8  0142    INTEGER NEXTBUFI SYN NEXTBUF;
08 0000  09 06F8  0143    REAL NEXTBUFR SYN NEXTBUF;
08 0000  09 06F8  0144    LONG REAL NEXTBUFLR SYN NEXTBUF(4);
08 0000  09 06F8  0145   ARRAY    80  INTEGER STACK;
08 0000  09 0838  0146    SHORT INTEGER STACKLNG SYN STACK;  COMMENT LENGTH FIELD OF STACK;
08 0000  09 0838  0147    SHORT INTEGER STACKIND SYN STACK;  COMMENT INDEX FILED TO SCRATCH TAB;
08 0000  09 0838  0148   BYTE PROGRAMM4 SYN #0000;
08 0000  09 0838  0149   BYTE PROGRAMM3 SYN #0001;
08 0000  09 0838  0150   BYTE PROGRAMM2 SYN #0002;
08 0000  09 0838  0151   BYTE PROGRAMM1 SYN #0003;
08 0000  09 0838  0152   BYTE PROGRAM SYN #0004;
08 0000  09 0838  0153
08 0000  09 0838  0154  DUMMY BASE R11;  COMMENT COMPILER COMMON FORMAT;
08 0000  10 0000  0155   INTEGER COMMTIME;
08 0000  10 0004  0156   SHORT INTEGER COMMLINE, COMMPAGE;
08 0000  10 0008  0157   ARRAY 3 LOGICAL COMMFLAGS;
08 0000  10 0014  0158    BYTE NOGO SYN COMMFLAGS(0);
08 0000  10 0014  0159    BYTE TRACE SYN COMMFLAGS(1);
08 0000  10 0014  0160    BYTE CHECKFLAG SYN COMMFLAGS(2);
08 0000  10 0014  0161    BYTE CARDFLAG SYN COMMFLAGS(3);
08 0000  10 0014  0162    ARRAY 2 LOGICAL TRACEBITS SYN COMMFLAGS(4);
08 0000  10 0014  0163   SHORT INTEGER BLOCKLISTSIZE, NAMETABLESIZE;
08 0000  10 0018  0164   INTEGER REFRECBASE, IDDBASE, IDLBASE, INPOINT;
08 0000  10 0028  0165   INTEGER TREELINK, TREEBASE, TREETOP;
08 0000  10 0034  0166   ARRAY 512 INTEGER BLOCKLIST;
08 0000  10 0834  0167    ARRAY 512 SHORT INTEGER BLOCKLISTLNG SYN BLOCKLIST(0);
08 0000  10 0834  0168    ARRAY 512 SHORT INTEGER BLOCKLISTIND SYN BLOCKLIST(2);
08 0000  10 0834  0169   COMMENT  *** SIZE OF OTHER TABLES IS DYNAMIC *** ;
08 0000  10 0834  0170   ARRAY 3 LOGICAL NAMETABLE;
08 0000  10 0840  0171   ARRAY 6 SHORT INTEGER SCRATCHNAMETABLE SYN NAMETABLE(2);
08 0000  10 0840  0172    SHORT INTEGER SCRNAMETAB  SYN SCRATCHNAMETABLE(0);
08 0000  10 0840  0173    SHORT INTEGER SIMTYPEIN   SYN SCRATCHNAMETABLE(4);
08 0000  10 0840  0174    INTEGER       TYPEINFO    SYN SCRATCHNAMETABLE(6);
```

```
08 0000   10 0840   0175     SHORT INTEGER IDNOSC     SYN SCRATCHNAMETABLE(10);
08 0000   10 0840   0176    ARRAY 1 SHORT INTEGER REFRCDLIST SYN B1;
08 0000   10 0840   0177    ARRAY 2 SHORT INTEGER IDDIR SYN B12;
08 0000   10 0840   0178     SHORT INTEGER IDDIRLNG SYN IDDIR(0);
08 0000   10 0840   0179     SHORT INTEGER IDDIRIND SYN IDDIR(2);
08 0000   10 0840   0180    ARRAY 1 BYTE IDLIST SYN B12;
08 0000   10 0840   0181   CLOSE BASE;
08 0000   09 0838   0182
08 0000   09 0838   0183    FUNCTION NI(4,#9400), OI(4,#9600), LTR(1,#1200), XC(5,#D700);
08 0000   09 0838   0184
08 0000   09 0838   0185    ARRAY 2 LOGICAL CHAININFO;   COMMENT STORAGE CONTROL INFO;
08 0000   09 0840   0186    ARRAY 3072 SHORT INTEGER CHAINID SYN B11;
08 0000   09 0840   0187
08 0000   09 0840   0188   SEGMENT BASE R14;
08 0004   11 0000   0189    COMMENT  ***  THIS SEGMENT IS READ-ONLY DATA  *** ;
08 0004   11 0000   0190    ARRAY   0  BYTE RESERVED;
08 0004   11 0000   0191    ARRAY  16  BYTE RESERVED1
08 0004   11 0000   0192       =(";():=+-*/,<>|#""¬");
08 0004   11 0010   0193    ARRAY  12  BYTE RESERVED2
08 0004   11 0010   0194       =("DOGOIFISOFOR");
08 0004   11 001C   0195    ARRAY  24  BYTE RESERVED3
08 0004   11 001C   0196       =("ABSANDDIVENDFORREMSHLSHR");
08 0004   11 0034   0197    ARRAY  40  BYTE RESERVED4
08 0004   11 0034   0198       =("BITSCASEELSEGOTOLONGNULLREALSTEPTHENTRUE");
08 0004   11 005C   0199    ARRAY  35  BYTE RESERVED5
08 0004   11 005C   0200       =("ARRAYBEGINFALSESHORTUNTILVALUEWHILE");
08 0004   11 007F   0201    ARRAY  18  BYTE RESERVED6
08 0004   11 007F   0202       =("RECORDRESULTSTRING");
08 0004   11 0091   0203    ARRAY  28  BYTE RESERVED7
08 0004   11 0091   0204       =("BOOLEANCOMPLEXINTEGERLOGICAL");
08 0004   11 00AD   0205    ARRAY   0  BYTE RESERVED8;
08 0004   11 00AD   0206    ARRAY  18  BYTE RESERVED9
08 0004   11 00AD   0207       =("PROCEDUREREFERENCE");
08 0004   11 00BF   0208    ARRAY   0  SHORT INTEGER CODE ;
08 0004   11 00C0   0209    ARRAY  16  SHORT INTEGER CODE1
08 0004   11 00C0   0210       =(#7009 ,#6A06 ,#6707 ,#9904 ,#9001 ,#7E01 ,#7F01 ,#7405
08 0004   11 00E0   0211       ,#8301 ,#6908 ,#8F01 ,#9101 ,#7601 ,#8E02 ,#8103 ,#8701);
08 0004   11 00E0   0212    ARRAY   6  SHORT INTEGER CODE2
08 0004   11 00E0   0213       =(#9301 ,#FF01 ,#7801 ,#7D01 ,#7C01 ,#8001);
08 0004   11 00EC   0214    ARRAY   8  SHORT INTEGER CODE3
08 0004   11 00EC   0215       =(#8D01 ,#8601 ,#8401 ,#6F0D ,#9B0A ,#8501 ,#8801 ,#8901);
08 0004   11 00FC   0216    ARRAY  10  SHORT INTEGER CODE4
08 0004   11 00FC   0217       =(#080F ,#7B01 ,#7A01 ,#9401 ,#8C0B ,#8201 ,#020E ,#9C01
08 0004   11 0110   0218       ,#7901 ,#8A01);
08 0004   11 0110   0219    ARRAY   7  SHORT INTEGER CODE5
08 0004   11 0110   0220       =(#6E13 ,#970C ,#8B01 ,#9F01 ,#9D01 ,#7211 ,#9E01);
08 0004   11 011E   0221    ARRAY   3  SHORT INTEGER CODE6
08 0004   11 011E   0222       =(#7514 ,#7312 ,#0716);
08 0004   11 0124   0223    ARRAY   4  SHORT INTEGER CODE7
08 0004   11 0124   0224       =(#060E ,#040E ,#010E ,#060E);
08 0004   11 012C   0225    ARRAY   0  SHORT INTEGER CODE8;
08 0004   11 012C   0226    ARRAY   2  SHORT INTEGER CODE9
08 0004   11 012C   0227       =(#7115 ,#6810);
08 0004   11 0130   0228     ARRAY  9  SHORT INTEGER RESNO         COMMENT NO. OF  RES. WRDS - 1;
08 0004   11 0130   0229       =(_1 , 5 , 7 , 9 , 6 , 2 , 3 ,_1 , 1);
08 0004   11 0142   0230    COMMENT  SPECIAL CODES;
08 0004   11 0142   0231    SHORT INTEGER NEWCARDCODE=#FE,SPECCOMMA=#66,SPECCOLON=#6D,
08 0004   11 0148   0232       SIMPLETYPE=#0D,NUMBERCODE=#77,ASSIGNCODE=#9A,EXPONENT=#88,
```

```
08 0004  11 0150  0233      IDCODE=#65,ENDOFPROGRAM=#92,FILETYPE=#6B,
08 0004  11 0156  0234      PLUSCODE=#7E,MINUSCODE=#7F,LPARENCODE=#6A,COMMACODE=#69,
08 0004  11 015E  0235      GOCODE=#FF,GOTOCODE=#94;
08 0004  11 0162  0236    SHORT INTEGER STRNGLENGTH=15;
08 0004  11 0164  0237    ARRAY  2  BYTE TO =("TO");
08 0004  11 0166  0238    COMMENT  INSTRUCTIONS TO BE USED WITH EXECUTE INSTRUCTION;
08 0004  11 0166  0239    ARRAY  3  SHORT INTEGER MOVE34=(#D200,#3000,#4000),
08 0004  11 016C  0240                            MOVE12=(#D200,#1000,#2000),
08 0004  11 0172  0241                            MOVE23=(#D200,#2000,#3000),
08 0004  11 0178  0242                            COMP23=(#D500,#2000,#3000);
08 0004  11 017E  0243    SHORT INTEGER IIDLISTINDEX=357S;
08 0004  11 0180  0244    ARRAY  357  BYTE IDLISTFILL =
08 0004  11 02E5  0245     ( "MAIN", "WRITEON", "READON", "READCARD", "IOCONTROL", "ODD",
08 0004  11 02E5  0246       "BITSTRING", "NUMBER", "DECODE", "TRUNCATE", "ROUND", "ENTIER",
08 0004  11 02E5  0247       "LONGREALPART", "LONGIMAGPART", "LONGSQRTERR", "LONGEXPERR",
08 0004  11 02E5  0248       "LONGLNLOGERR", "LONGLOG", "LONGSINCOSERR", "LONGCOS", "LONGARCTAN",
08 0004  11 02E5  0249       "LONGCOMPLEXSQRT", "LONGBASE10", "LONGBASE16", "INTBASE10",
08 0004  11 02E5  0250       "INTBASE16", "TIME", "INTFIELDSIZE", "MAXINTEGER", "LONGEPSILON",
08 0004  11 02E5  0251       "PI", "MAXREAL", "INTOVFL", "UNFL", "INTDIVZERO", "EXCEPTION",
08 0004  11 02E5  0252       "XCPNOTED", "XCPLIMIT", "XCPACTION", "XCPMARK", "XCPMSG",
08 0004  11 02E5  0253       "ROUNDTOREAL", "EXPONENT"
08 0004  11 02E5  0254     );
08 0004  11 02E5  0255    SHORT INTEGER IIDDIRINDEX=268S;
08 0004  11 02E8  0256    ARRAY  134  SHORT INTEGER IDDIRFILL =
08 0004  11 03F4  0257     ( 3,0, 4,4, 6,4, 3,11, 5,11, 7,17, 8,25, 2,34, 8,37, 5,46, 5,52,
08 0004  11 03F4  0258       3,54, 7,58, 4,66, 5,71, 7,81, 7,93, 11,77, 11,89, 3,93, 7,89,
08 0004  11 03F4  0259       3,105, 2,116, 1,126, 2,138, 2,145, 2,158, 5,165, 7,101, 6,112,
08 0004  11 03F4  0260       5,122, 6,134, 6,141, 6,154, 9,161, 10,175, 14,171, 5,190, 5,200,
08 0004  11 03F4  0261       9,186, 9,196, 8,206, 8,215, 3,224, 11,228, 9,240, 6,254, 1,261,
08 0004  11 03F4  0262       6,263, 10,250, 6,270, 9,281, 3,273, 3,277, 6,284, 6,105, 5,116,
08 0004  11 03F4  0263       7,126, 8,145, 8,291, 7,300, 7,308, 8,316, 6,325, 5,332, 10,338,
08 0004  11 03F4  0264       7,349
08 0004  11 03F4  0265     );
08 0004  11 03F4  0266    SHORT INTEGER ISYMBOLINDEX=804S;
08 0004  11 03F6  0267    ARRAY  201  INTEGER NAMETFILL =
08 0004  11 071C  0268     ( #00000000,  #00000000,  #03000000,
08 0004  11 071C  0269       #00000000,  #00000100,  #09000001,
08 0004  11 071C  0270       #00000000,  #00000100,  #09000002,
08 0004  11 071C  0271       #00000000,  #00000200,  #09000003,
08 0004  11 071C  0272       #00000000,  #00000200,  #09000004,
08 0004  11 071C  0273       #00000000,  #00000200,  #09000005,
08 0004  11 071C  0274       #00000000,  #00000100,  #09000006,
08 0004  11 071C  0275       #00000000,  #00000001,  #07060007,
08 0004  11 071C  0276       #00000000,  #00000001,  #07080008,
08 0004  11 071C  0277       #00000000,  #00000008,  #07010009,
08 0004  11 071C  0278       #00000000,  #00000007,  #0701000A,
08 0004  11 071C  0279       #00000000,  #00000001,  #0707000B,
08 0004  11 071C  0280       #00000000,  #00000002,  #0701000C,
08 0004  11 071C  0281       #00000000,  #00000002,  #0701000D,
08 0004  11 071C  0282       #00000000,  #00000002,  #0701000E,
08 0004  11 071C  0283       #00000000,  #00000002,  #07010042,
08 0004  11 071C  0284       #00000000,  #00000003,  #07020041,
08 0004  11 071C  0285       #00000000,  #00000004,  #0702000F,
08 0004  11 071C  0286       #00000000,  #00000004,  #07020010,
08 0004  11 071C  0287       #00000000,  #00000005,  #07030011,
08 0004  11 071C  0288       #00000000,  #00000005,  #07030012,
08 0004  11 071C  0289       #00000000,  #00000002,  #07040013,
08 0004  11 071C  0290       #00000000,  #00000003,  #07050014,
```

```
C8 0004  11 071C  0291      #00000000,  #00000002,  #07020015,
08 0004  11 071C  0292      #00000000,  #00000002,  #07020016,
C8 0004  11 071C  0293      #00000000,  #00000002,  #07020017,
C8 0004  11 071C  0294      #00000000,  #00000002,  #07020018,
08 0004  11 071C  0295      #00000000,  #00000002,  #07020019,
C8 0004  11 071C  0296      #00000000,  #00000002,  #0702001A,
08 0004  11 071C  0297      #00000000,  #00000002,  #0702001B,
08 0004  11 071C  0298      #00000000,  #00000003,  #0703001C,
08 0004  11 071C  0299      #00000000,  #00000003,  #0703001D,
08 0004  11 071C  0300      #00000000,  #00000003,  #0703001E,
C8 0004  11 071C  0301      #00000000,  #00000003,  #0703001F,
C8 0004  11 071C  0302      #00000000,  #00000003,  #07030020,
08 0004  11 071C  0303      #00000000,  #00000003,  #07030021,
08 0004  11 071C  0304      #00000000,  #00000003,  #07030022,
08 0004  11 071C  0305      #00000000,  #00000004,  #07040023,
08 0004  11 071C  0306      #00000000,  #00000005,  #07050024,
C8 0004  11 071C  0307      #00000000,  #000B0001,  #07070029,
08 0004  11 071C  0308      #00000000,  #000B0001,  #0707002A,
08 0004  11 071C  0309      #00000000,  #000B0002,  #07070025,
C8 0004  11 071C  0310      #00000000,  #000B0002,  #07070026,
08 0004  11 071C  0311      #00000000,  #00130003,  #07070027,
C8 0004  11 071C  0312      #00000000,  #00130003,  #07070028,
C8 0004  11 071C  0313      #00000000,  #00000001,  #0701002B,
08 0004  11 071C  0314      @ALGOLDATAORG(#1D0),  #00000000,  #0001002C,
08 0004  11 071C  0315      @ALGOLDATAORG(#1CC),  #00000000,  #0001002D,
C8 0004  11 071C  0316      @ALGOLDATAORG(#1C8),  #00000000,  #0002002E,
08 0004  11 071C  0317      @ALGOLDATAORG(#1B0),  #00000000,  #0003002F,
08 0004  11 071C  0318      @ALGOLDATAORG(#1B8),  #00000000,  #00030030,
08 0004  11 071C  0319      @ALGOLDATAORG(#1C0),  #00000000,  #00030031,
08 0004  11 071C  0320      @ALGOLDATAORG(#1D4),  #00010001,  #00090032,
08 0004  11 071C  0321      @ALGOLDATAORG(#1D8),  #00010001,  #00090033,
08 0004  11 071C  0322      @ALGOLDATAORG(#1DC),  #00010001,  #00090034,
08 0004  11 071C  0323      @ALGOLDATAORG(#1E0),  #00010001,  #00090035,
08 0004  11 071C  0324      @ALGOLDATAORG(#1E8),  #00010001,  #00090036,
08 0004  11 071C  0325      @ALGOLDATAORG(#1EC),  #00010001,  #00090037,
08 0004  11 071C  0326      @ALGOLDATAORG(#1F0),  #00010001,  #00090038,
C8 0004  11 071C  0327      @ALGOLDATAORG(#1F4),  #00010001,  #00090039,
08 0004  11 071C  0328      @ALGOLDATAORG(#1F8),  #00010001,  #0009003A,
C8 0004  11 071C  0329      #000D0000,  #004C0501,  #0400003B,
08 0004  11 071C  0330      #000D0001,  #00000001,  #0506003C,
08 0004  11 071C  0331      #000D0004,  #00000001,  #0501003D,
08 0004  11 071C  0332      #000D0008,  #00000001,  #0501003E,
08 0004  11 071C  0333      #000D0002,  #00000001,  #0506003F,
08 0004  11 071C  0334      #000D000C,  #003F0001,  #05070040
C8 0004  11 071C  0335    );
08 0004  11 071C  0336    ARRAY 256 CHARACTER TRTBLANKS =(64(#03),#00,10(#03),#04,49(#03),
C8 0004  11 081C  0337                                   #02,67(#03),9(#01),7(#03),
08 0004  11 081C  0338                                   9(#01),8(#03),8(#01),6(#03),10(#02),
08 0004  11 081C  0339                                   6(#03));
C8 0004  11 081C  0340    ARRAY 3 SHORT INTEGER SCANBLANKS =(#DD00,#8000,@TRTBLANKS);
08 0004  11 0822  0341    ARRAY 256 CHARACTER TRTIDS =(193(#01),9(#00),7(#01),9(#00),8(#01),
08 0004  11 0922  0342                                 8(#00),6(#01),10(#00),6(#03));
C8 0004  11 0922  0343    ARRAY 3 SHORT INTEGER IDCHARS =(#DD00,#8000,@TRTIDS);
08 0004  11 0928  0344    ARRAY 256 CHARACTER TRTCOMMENT =(94(#00),#01,161(#00));
08 0004  11 0A23  0345    ARRAY 3 SHORT INTEGER COMMNTCHARS =(#DD00,#8000,@TRTCOMMENT);
08 0004  11 0A2E  0346    ARRAY 3 SHORT INTEGER MVCCHARS =(#D200,#4000,#8000);
C8 0004  11 0A34  0347    ARRAY 52 CHARACTER TRRESERVED1 =(20,2,10,24,12(255),14,4,0,30,12,
08 0004  11 0A68  0348                                    16,9(255),18,2(255),22,11(255),6,
```

```
C8 00C4   11 0A68          0349                                    26,2(255),8,28);
C8 00C4   11 0A68          0350     SHORT INTEGER STACKLIMIT=312; COMMENT MAXIMUM STACK INDEX;
08 0004   11 0A6A          0351   CLOSE BASE;

                         SEGMENT 11  NAME = SEG#11      LENGTH = 0A6A  BASE REG = 14

08 00C4   09 0840          0352
08 00C4   09 0840          0353     PROCEDURE OUTCODE(R11);
08 00C4   09 0840          0354        BEGIN COMMENT R1 HAS CODE TO BE OUTPUT AS 1 BYTE;
08 00C8   09 0840          0355           STC(R1,PROGRAM(R6));  R6 := R6 + 1;
08 00D0   09 0840          0356        END;
08 00D2   09 0840          0357
08 00D2   09 0840          0358     PROCEDURE OUTCODE2(R11);
08 00D2   09 0840          0359        BEGIN COMMENT R2 HAS CODE TO BE OUTPUT AS 2 BYTES;
08 00D2   09 0840          0360           OUTCHAR := R2;  R1 := @PROGRAM(R6);
08 00DA   09 0840          0361           MVC(1,B1,OUTCHAR(2));  R6 := R6 + 2;
C8 00E4   09 0840          0362        END;
C8 00E6   09 0840          0363
08 00E6   09 0840          0364     PROCEDURE OUTCODE4(R11);
C8 00E6   09 0840          0365        BEGIN  COMMENT R2 HAS CODE TO BE OUTPUT AS 4 BYTES;
08 00E6   09 0840          0366           OUTCHAR := R2;  R1 := @PROGRAM(R6);
08 00EE   09 0840          0367           MVC(3,B1,OUTCHAR);  R6 := R6 + 4;
C8 00F8   09 0840          0368        END;
08 00FA   09 0840          0369
08 00FA   09 0840          0370     PROCEDURE SETTYPE(R10);
C8 00FA   09 0840          0371        BEGIN    COMMENT  R1 HAS PARTIAL TYPE CODE;
08 00FA   09 0840          0372           CLI(#01,PROCFLAG);  IF = THEN R1 := R1 + #1000  ELSE
C8 0106   09 0840          0373           BEGIN CLI(#01,RECORDFLAG);
08 010E   09 0840          0374              IF  =  THEN
08 0112   09 0840          0375              BEGIN  R2 := RECORDNO SHLL 16;  R1 := R1 OR R2 + #500;
08 0120   09 0840          0376              END;
C8 0120   09 0840          0377           END;
08 0120   09 0840          0378        END;
C8 0122   09 0840          0379
C8 0122   09 0840          0380     PROCEDURE HASH(R11);
08 0122   09 0840          0381        BEGIN  COMMENT  R4 HAS LENGTH, R1 HAS FIRST CHARACTER IN LOW BYTE;
C8 0122   09 0840          0382           R1 := R1 AND #F SHLL 4;  IF R4 >= 15 THEN
C8 0132   09 0840          0383           BEGIN  SET(LAST);  R1 := R1 OR 15;  END ELSE
08 013A   09 0840          0384           BEGIN  RESET(LAST);  R1 := R1 OR R4;  END;
C8 0144   09 0840          0385        END;
C8 0146   09 0840          0386
08 0146   09 0840          0387     PROCEDURE ERROR(R11);
C8 0146   09 0840          0388        BEGIN COMMENT R4 HAS ERROR NUMBER;
08 0146   09 0840          0389          SAVERB := R11;  R11 := NTBASE;
08 014E   09 0840          0390          SET(NOGO);  R1 := ERRORCOUNT + 5;
08 015A   09 0840          0391              IF R1 = ERRORLIMIT THEN R4 := 11;
08 0166   09 0840          0392              IF R1 <= ERRORLIMIT THEN
C8 016E   09 0840          0393              BEGIN  STC(R4,ERRORS(R1));  ERRORCOUNT := R1;
C8 0176   09 0840          0394                 R4 := @ERRORS(R1+1);  MVC(3,B4,INBUF);
08 0180   09 0840          0395              END;
08 0180   09 0840          0396          R11 := SAVERB;
C8 0184   09 0840          0397        END;
08 0186   09 0840          0398
08 0186   09 0840          0399     PROCEDURE SOFTSTOP(R11);
C8 0186   09 0840          0400        GOTO ENDPROGRAM;
08 018C   09 0840          0401
08 018C   09 0840          0402     SEGMENT PROCEDURE FETCHCARD(R11);
C8 018C   09 0840          0403        BEGIN  COMMENT  PRINTS LAST LINE, READS NEW CARD, DETECTS OPTIONS,
```

```
12 0000  09 0840  0404                        SETS   R8 = ADDRESS OF NEXT CHARACTER,
12 0000  09 0840  0405                               R9 = (NUMBER - 1) CHARACTERS ON CARD;
12 0000  09 0840  0406          STM(R1,R3,SAVE13);
12 0004  09 0840  0407          IF LISTFLAG AND LINEHOLD THEN
12 0014  09 0840  0408          BEGIN  RESET(LINEHOLD);
12 0018  09 0840  0409             R0 := @INBUF;  PRINT;
12 0023  09 0840  0410          END;
12 0028  09 0840  0411    L:    R0 := @INBUF(13);  R8 := R0;  IF EOF THEN
12 0036  09 0840  0412          BEGIN  R4 := 7;  ERROR;  SOFTSTOP;  END;
12 0052  09 0840  0413          R3 := AGETCARD;  BALR(R2,R3); COMMENT READ;  IF ¬= THEN
12 005C  09 0840  0414          BEGIN  R8 := @INBUF(80);  MVI(";",B8);  MVI(";",B8(1));
12 0068  09 0840  0415             R9 := 1;  SET(EOF);  GOTO XIT;
12 0074  09 0840  0416          END;
12 0074  09 0840  0417          CLI("$",B8);  IF = THEN
12 007C  09 0840  0418          BEGIN  SAVERB := R11;  R11 := NTBASE;
12 0084  09 0840  0419             CLC(4,"TRACE",B8(1)); IF = THEN
12 008E  09 0840  0420             BEGIN  MVC(0,TRACEFLAG,B8(6));  NI(#0F,TRACEFLAG);
12 0098  09 0840  0421                MVC(0,TRACE,B8(7));  NI(#0F,TRACE);
12 00A2  09 0840  0422                CLC(1,B8(9)," ");  IF = THEN
12 00AC  09 0840  0423                BEGIN  R0 := #FFFFFFFF;  R1 := R0;
12 00B2  09 0840  0424                END ELSE
12 00B2  09 0840  0425                BEGIN
12 00B6  09 0840  0426                   IC(R1,B8(10));  R1 := R1 AND #F;
12 00BE  09 0840  0427                   IC(R2,B8(9));  R2 := R2 AND #F * 10S + R1;
12 00CC  09 0840  0428                   R0 := #80000000;  R1 := R1 - R1;  SRDL(R0,B2);
12 00D6  09 0840  0429                   R0 := R0 OR TRACEBITS(0);  R1 := R1 OR TRACEBITS(4);
12 00DE  09 0840  0430                END;
12 00DE  09 0840  0431                STM(R0,R1,TRACEBITS);  GOTO M;
12 00E6  09 0840  0432             END;
12 00E6  09 0840  0433             CLC(4,"STACK",B8(1)); IF = THEN
12 00F0  09 0840  0434             BEGIN  MVI(3,TRACE);  MVI(#FF,TRACEBITS);
12 00F8  09 0840  0435                MVC(6,TRACEBITS(1),TRACEBITS);  GOTO  M;
12 0102  09 0840  0436             END;
12 0102  09 0840  0437             CLC(3,"LIST",B8(1));  IF  =  THEN
12 010C  09 0840  0438             BEGIN  SET(LISTFLAG);  GOTO M;  END;
12 0114  09 0840  0439             CLC(5,"NOLIST",B8(1));  IF  =  THEN
12 011E  09 0840  0440             BEGIN  RESET(LISTFLAG);  GOTO M;  END;
12 0126  09 0840  0441             CLC(6,"NOCHECK",B8(1)); IF = THEN
12 0130  09 0840  0442             BEGIN  RESET(CHECKFLAG); GOTO M;  END;
12 0138  09 0840  0443             CLC(5,"SYNTAX",B8(1));  IF = THEN
12 0142  09 0840  0444             BEGIN  SET(NOGO); GOTO M;  END;
12 014A  09 0840  0445             R11 := SAVERB;  GOTO N;
12 0152  09 0840  0446    M:       R11 := SAVERB;  GOTO L;
12 015A  09 0840  0447          END;
12 015A  09 0840  0448    N:    SET(NEWCARD);  SET(LINEHOLD);  RESET(BEGINSET);
12 0166  09 0840  0449          MVC(1,INBUF(5),"--");  R9 := 71;  R1 := CARDCOUNT + R5;
12 0176  09 0840  0450          CARDCOUNT := R1;  CVD(R1,PKDEC);  UNPK(3,7,INBUF,PKDEC);
12 0184  09 0840  0451          OI("0",INBUF(3));  MVC(7,INBUF(93),INBUF(85));
12 018E  09 0840  0452          MVC(7,INBUF(85),"       ");
12 0194  09 0840  0453    XIT:  LM(R1,R3,SAVE13);
12 0198  09 0840  0454       END;

                          SEGMENT 12  NAME = SEG#12      LENGTH = 01E8  BASE REG = 15

08 00CC  09 0840  0455
08 00CC  09 0840  0456    PROCEDURE NEXTCHAR(R3);
08 00CC  09 0840  0457       BEGIN  R8 := R8 + R5;  IF R8 > R9 THEN
08 00D4  09 0840  0458          BEGIN  FETCHCARD;  R0 := R0-R0;  R9 := @INBUF(84);  END;
```

```
C8 00E4  09 0840  0459          IC(R0,B8);
C8 00E8  09 0840  0460       END;
08 00EA  09 0840  0461
C8 00EA  09 0840  0462
C8 00EA  09 0840  0463    PROCEDURE STOP(R1);
08 00EA  09 0840  0464       BEGIN  R4 := 12;  ERROR;  SET(NOPASSTWO);  GOTO ENDPROGRAM;
C8 00FA  09 0840  0465       END;
C8 00FC  09 0840  0466
08 00FC  09 0840  0467    PROCEDURE INSYMBOL(R11);
C8 00FC  09 0840  0468       BEGIN COMMENT R2 HAS ID NUMBER AT ENTRANCE;
08 00FC  09 0840  0469          SAVERB := R11;  R11 := NTBASE;
08 0104  09 0840  0470          R1 := STACKINDEX - 4;  R0 := STACKLNG(R1) + 12;
08 0114  09 0840  0471          STACKLNG(R1) := R0;  R0 := TYPE;  R1 := SCRATCHINDEX;
08 0120  09 0840  0472          IF R1 < SYMBOLINDEX THEN STOP;  COMMENT NT OVERFLOW;
08 012C  09 0840  0473          R3 := R3-R3; SCRNAMETAB(R1) := R3; SCRNAMETAB(R1+2) := R3;
08 0136  09 0840  0474          TYPEINFO(R1) := R0;  IDNOSC(R1) := R2;  R3 := SIMTYPEINFO;
08 0142  09 0840  0475          SIMTYPEIN(R1) := R3;  R1 := R1 - 12;
08 014A  09 0840  0476          SCRATCHINDEX := R1;
08 014E  09 0840  0477          R11 := SAVERB;
08 0152  09 0840  0478       END;
08 0154  09 0840  0479
08 0154  09 0840  0480    PROCEDURE IDSEARCH(R10);
08 0154  09 0840  0481       BEGIN    COMMENT  R2 RETURNS ID NUMBER;
08 0154  09 0840  0482          R4 := CURBUFCOUNT;  R3 := @CURBUF; IC(R1,B3);  HASH;
08 0164  09 0840  0483          R1 := R1 SHLL 1;  R11 := CHAIN;  HCODE := R1;  R1 := HASHID(R1);
08 0174  09 0840  0484          WHILE R1 ¬= 0 DO
08 017A  09 0840  0485          BEGIN  R1 := R1 SHLL 1;  R2 := IDDIRIND(R1) + IDLISTBASE;
08 0186  09 0840  0486             IF ¬LAST THEN EX(R4,COMP23) ELSE
08 0192  09 0840  0487             IF R4 = IDDIRLNG(R1) THEN EX(R4,COMP23);
08 01A2  09 0840  0488             IF  =  THEN GOTO FINISH;
08 01A6  09 0840  0489             R1 := R1 SHRL 1;  R1 := CHAINID(R1);
08 01AE  09 0840  0490          END;
08 01B2  09 0840  0491          COMMENT NOT IN TABLE;
08 01B2  09 0840  0492          R1 := IDDIRINDEX + 4;  R2 := IDLISTINDEX;
08 01BE  09 0840  0493          IF R1 > IDDIRLIMIT THEN STOP;
08 01CA  09 0840  0494          IF R2 > IDLISTLIMIT THEN STOP;
08 01D6  09 0840  0495          IDDIRIND(R1) := R2;  IDDIRLNG(R1) := R4;
08 01DE  09 0840  0496          R0 := R2 + R4 + 1;  IDLISTINDEX := R0;
08 01EA  09 0840  0497          IDDIRINDEX := R1;  R2 := R2 + IDLISTBASE;  EX(R4,MOVE23);
08 01F6  09 0840  0498          R4 := HCODE;  R2 := HASHID(R4);  R3 := R1 SHRL 1;
08 02C4  09 0840  0499          CHAINID(R3) := R2;  HASHID(R4) := R3;
08 020C  09 0840  0500    FINISH:R2 := R1 SHRL 2;
08 0212  09 0840  0501          CLI(#01,DECLARFLAG);  IF  =  THEN
08 021A  09 0840  0502          BEGIN  CLI(#01,REFERFLAG);
08 021E  09 0840  0503             IF = THEN
08 0222  09 0840  0504             BEGIN  R1 := RLISTPOINT + 2;  RLISTPOINT := R1;
08 022E  09 0840  0505                R1 := R1 + REFRCLISTBASE;  REFRCDLIST := R2;
08 0236  09 0840  0506                IF R1 >= IDDIRBASE THEN STOP;
08 0242  09 0840  0507             END ELSE INSYMBOL;
08 024A  09 0840  0508          END;
08 024A  09 0840  0509          R1 := IDCODE;  OUTCODE;  OUTCODE2;
08 0256  09 0840  0510    XIT:END;
08 0258  09 0840  0511
08 0258  09 0840  0512     PROCEDURE OUTSTRING(R10);
08 0258  09 0840  0513        BEGIN COMMENT R1 HAS (NUMBER OF CHAR - 1) TO MOVE FROM BUF WITH BASE
08 0258  09 0840  0514                      R7;
08 0258  09 0840  0515          R4 := @SYMBOLBUFFERS(4);  R3 := @PROGRAM(R6); EX(R1,MOVE34);
08 02E4  09 0840  0516          R6 := R6 + R1 + 1;
```

```
08 026A  09 0840   0517      END;
08 026C  09 0840   0518
08 026C  09 0840   0519    PROCEDURE OPENBLOCK(R10);
08 026C  09 0840   0520       BEGIN
08 026C  09 0840   0521          R1 := BLOCKNO + R5;  BLOCKNO := R1;  R2 := STACKINDEX;
08 027A  09 0840   0522          IF R1 > BLLIMIT OR R2 >= STACKLIMIT THEN STOP;
08 028E  09 0840   0523          COMMENT TOO MANY BLOCKS OR NESTING TOO DEEP;
08 028E  09 0840   0524          STACK(R2) := R1;  R1 := SCRATCHINDEX;  STACK(R2+4) := R1;
08 029A  09 0840   0525          R2 := R2 + 8;  STACKINDEX := R2;
08 02A2  09 0840   0526       END;
08 02A4  09 0840   0527
08 02A4  09 0840   0528    PROCEDURE  CLOSEBLOCK(R10);
08 02A4  09 0840   0529       BEGIN
08 02A4  09 0840   0530          R1 := STACKINDEX - 8;
08 02AC  09 0840   0531          IF ¬ENDDOT AND R1 = 0 THEN R0 := 1 ELSE R0 := R0 - R0;
08 02C4  09 0840   0532          IF R0 = 1 OR R1 < 0 THEN
08 02D2  09 0840   0533          BEGIN
08 02D2  09 0840   0534             R4 := 4;  ERROR;  GOTO XIT;
08 02DE  09 0840   0535          END;
08 02DE  09 0840   0536          SAVERB := R11;  R11 := NTBASE;
08 02E6  09 0840   0537          STACKINDEX := R1;
08 02EA  09 0840   0538          R3 := STACK(R1) SHLL 2;  R0 := STACKLNG(R1+4);
08 02F6  09 0840   0539          R1 := STACKIND(R1+6);
08 02FA  09 0840   0540          COMMENT  R0 HAS NO OF IDS * 12
08 02FA  09 0840   0541                   R1 HAS INDEX TO SCRATCH AND IDNOSCRATCH TABLES
08 02FA  09 0840   0542                   R3 HAS INDEX TO BLOCKLIST (BLOCK NUMBER * 4);
08 02FA  09 0840   0543          IF R0=0 THEN BLOCKLIST(R3) := R0 ELSE
08 03C4  09 0840   0544          BEGIN R2 := SYMBOLINDEX;  BLOCKLIST(R3) := R2;
08 0310  09 0840   0545             BLOCKLISTLNG(R3) := R0;  R3 := R0 + R2;  SYMBOLINDEX := R3;
08 031C  09 0840   0546             SCRATCHINDEX := R1;
08 0320  09 0840   0547             R3 := @NAMETABLE(R2); R4 := @SCRATCHNAMETABLE(R1);
08 0328  09 0840   0548             FOR R1:=12 STEP 12 UNTIL R0 DO
08 032C  09 0840   0549             BEGIN  R2 := R3 + 12; IF R2 >= R4 THEN STOP;
08 0340  09 0840   0550                MVC(11,B3,B4);  R3 := R2; R4 := R4 - 12;
08 034C  09 0840   0551             END;
08 0356  09 0840   0552          END;
08 0356  09 0840   0553          R11 := SAVERB;
08 035A  09 0840   0554    XIT:END;
08 035C  09 0840   0555
08 035C  09 0840   0556     PROCEDURE MATCHSYMBOL(R10);
08 035C  09 0840   0557        BEGIN COMMENT  SYMBOL IN CURBUF
08 035C  09 0840   0558                       R1 RETURNS OUTPUT CODE
08 035C  09 0840   0559                       R2 RETURNS CASE STATEMENT INDEX
08 035C  09 0840   0560                       R3 = 1 IF SUCCESSFUL, =0 OTHERWISE;
08 035C  09 0840   0561          R2 := R2 - R2;  R1 := R2;  IC(R2,CURBUF);
08 0364  09 0840   0562          IF R2 >= 76 AND R2 <= 127 THEN
08 0374  09 0840   0563          BEGIN  TR(0,CURBUF,TRRESERVED1(_76));  IC(R1,CURBUF);
08 037E  09 0840   0564             STC(R2,CURBUF);
08 0382  09 0840   0565             IF R1 = 255 THEN R3 := R3 - R3 ELSE
08 038C  09 0840   0566             BEGIN  IC(R2,CODE1(R1+1));  IC(R1,CODE1(R1));  R3 := 1;
08 039C  09 0840   0567             END;
08 039C  09 0840   0568          END ELSE  R3 := R3 - R3;
08 03A2  09 0840   0569       END;
08 03A4  09 0840   0570
08 03A4  09 0840   0571     SEGMENT PROCEDURE  NUMBERCONVERT(R11);
08 03A4  09 0840   0572       BEGIN
13 0000  09 0840   0573       PROCEDURE FLOAT(R10);
13 0000  09 0840   0574          BEGIN FCON1LOW := R1;  F01 := FCON1  + 0L;
```

```
13 0010  09 0840   0575          END;
13 0012  09 0840   0576       PROCEDURE SCALEFACTOR(R10);
13 0012  09 0840   0577          BEGIN  COMMENT ADDS TO THE SCALEFACTOR IN R2;
13 0012  09 0840   0578             NEXTCHAR;  IF R0 = "-" THEN
13 0026  09 0840   0579                BEGIN  SET(EXPOSIGN);  NEXTCHAR;
13 0036  09 0840   0580                END ELSE RESET(EXPOSIGN);
13 003E  09 0840   0581             IF R0 = "+" THEN NEXTCHAR;
13 0052  09 0840   0582             IF R0 < "0" THEN
13 005A  09 0840   0583             BEGIN  R4 := 2; ERROR; R1 := 0;   GOTO XIT;
13 0072  09 0840   0584             END;
13 0072  09 0840   0585             R1 := R0 AND #F;  NEXTCHAR;
13 0084  09 0840   0586             WHILE R0 >= "0" DO
13 008C  09 0840   0587             BEGIN  R0 := R0 AND #F;  R1 := R1 * 10S + R0;  NEXTCHAR;
13 00A2  09 0840   0588             END;
13 00A6  09 0840   0589             TEST(EXPOSIGN);
13 00AA  09 0840   0590             IF  =  THEN R2 := R2 - R1 ELSE R2 := R2 + R1;
13 00B6  09 0840   0591    XIT:  END;
13 00B8  09 0840   0592       PROCEDURE  CONVERT(R11);
13 00B8  09 0840   0593          BEGIN  SAVERB1 := R11;  IF R0 >= "0" THEN
13 00C4  09 0840   0594             BEGIN  R1 := R0 AND #F;  R2 := 0;  NEXTCHAR; MVI(1,TYPEFLAG);
13 00DE  09 0840   0595                WHILE R0 >= "0" DO
13 00E6  09 0840   0596                BEGIN IF R2 > 7 THEN IF R1 >= 214748364 THEN
13 00F6  09 0840   0597                   BEGIN  IF  =  THEN IF R0 <= "7" THEN GOTO OK;
13 0102  09 0840   0598                      R4 := 2;  ERROR;  WHILE R0 >= "0" DO NEXTCHAR;
13 012A  09 0840   0599                      GOTO FL;
13 012E  09 0840   0600                   END;
13 012E  09 0840   0601    OK:            R0 := R0 AND #F;  R1 := R1 * 10S + R0;  R2 := R2 + 1;
13 013C  09 0840   0602                   NEXTCHAR;
13 0148  09 0840   0603                END;
13 014C  09 0840   0604
13 014C  09 0840   0605                COMMENT  INTEGER IN R1, NEXT CHARACTER IN R0;
13 014C  09 0840   0606    FL:         R2 := 0;  COMMENT R2 IS THE DECIMAL SCALE FACTOR;
13 0150  09 0840   0607                IF R0 = "'" THEN
13 0158  09 0840   0608                BEGIN MVI(2,TYPEFLAG);  FLOAT;  SCALEFACTOR;
13 0164  09 0840   0609                END ELSE
13 0164  09 0840   0610                IF R0 = "." THEN
13 0170  09 0840   0611                BEGIN  COMMENT PROCESS FRACTION;
13 0170  09 0840   0612                   FLOAT;  NEXTCHAR;  MVI(2,TYPEFLAG);
13 0184  09 0840   0613                   WHILE R0 >= "0"  DO
13 018C  09 0840   0614                   BEGIN  R0 :=R0 AND #F SHLL 4;
13 0194  09 0840   0615                      STC(R0,FCON2(4));
13 0198  09 0840   0616                      F01 := F01 * 10L + FCON2;  R2 := R2 - 1;
13 01A4  09 0840   0617                      NEXTCHAR;
13 01B0  09 0840   0618                   END;
13 01B4  09 0840   0619                   IF R0 = "'" THEN SCALEFACTOR;
13 01C0  09 0840   0620                END;
13 01C0  09 0840   0621                CLI(1,TYPEFLAG);
13 01C4  09 0840   0622                IF > THEN
13 01C8  09 0840   0623                BEGIN  IF R2 ¬= 0 THEN
13 01CE  09 0840   0624                   BEGIN  F23 := 10L;  F45 := 1L;  F67 := 1L;
13 01DA  09 0840   0625                      IF R2 < 0 THEN
13 01E0  09 0840   0626                      BEGIN  R2 := ABS R2;
13 01E2  09 0840   0627                         WHILE R2 ¬= 0 DO
13 01E8  09 0840   0628                         BEGIN  F23 := F23 * F45;  F45 := F23;
13 01EC  09 0840   0629                            SRDL(R2,1);  LTR(R3,R3);
13 01F2  09 0840   0630                            IF < THEN F01 := F01 / F23;
13 01F8  09 0840   0631                         END;
13 01FC  09 0840   0632                      END ELSE
```

```
13 01FC  09 0840  0633                        WHILE R2 ¬=  0 DO
13 0206  09 0840  0634                           BEGIN  F23 := F23 * F45;  F45 := F23;
13 020A  09 0840  0635                              SRDL(R2,1);  LTR(R3,R3);
13 0210  09 0840  0636                              IF < THEN F01 := F01 * F23;
13 0216  09 0840  0637                           END;
13 021A  09 0840  0638                     END;
13 021A  09 0840  0639                  END;
13 021A  09 0840  0640               END;
13 021A  09 0840  0641               R11 := SAVERB1;
13 021E  09 0840  0642            END;
13 0220  09 0840  0643            RESET(IMAGFLAG);  RESET(EXPOSIGN);  RESET(SIGN);
13 022C  09 0840  0644            R0 := R0-R0;  IC(R0,B8);  SAVEN := R11;  R9 := @INBUF(84);
13 023A  09 0840  0645            IF R0 = "." THEN
13 0242  09 0840  0646            BEGIN  R0 := "0";  R8 := R8 - R5;  END;
13 0248  09 0840  0647            IF R0 = "'" THEN
13 0250  09 0840  0648            BEGIN  R0 := "1";  R8 := R8 - R5;  END;
13 0256  09 0840  0649            CONVERT;
13 025A  09 0840  0650            IF R0 = "I" THEN
13 0262  09 0840  0651            BEGIN  SET(IMAGFLAG);  CLI(1,TYPEFLAG);
13 026A  09 0840  0652               IF  =  THEN FLOAT;  NEXTCHAR;  MVI(2,TYPEFLAG);
13 0282  09 0840  0653            END;
13 0282  09 0840  0654            IF R0 = "L" THEN
13 028A  09 0840  0655            BEGIN CLI(1,TYPEFLAG);  IF  =  THEN FLOAT;  NEXTCHAR;
13 02A2  09 0840  0656               MVI(3,TYPEFLAG);
13 02A6  09 0840  0657            END;
13 02A6  09 0840  0658            R2 := R2 - R2;  IC(R2,TYPEFLAG);  TEST(IMAGFLAG);
13 02B0  09 0840  0659            IF  =  THEN R2 := R2 + 2;
13 02B8  09 0840  0660            NEXTBUFCOUNT := R2;
13 02BC  09 0840  0661            CASE R2 OF
13 02BC  09 0840  0662            BEGIN
13 02BC  09 0840  0663               NEXTBUFI := R1;
13 02C8  09 0840  0664               NEXTBUFR := F0;
13 02D0  09 0840  0665               NEXTBUFLR := F01;
13 02D8  09 0840  0666               BEGIN  F2 := F2 - F2;  NEXTBUFR := F2;  NEXTBUFR(4) := F0 ;
13 02E6  09 0840  0667               END;
13 02E6  09 0840  0668               BEGIN  F23 := F23 - F23;  NEXTBUFLR := F23;
13 02F0  09 0840  0669                  NEXTBUFLR(8) := F01;
13 02F4  09 0840  0670               END;
13 02F4  09 0840  0671            END;
13 030C  09 0840  0672            R9 := @INBUF(84) - R8;  R11 := SAVEN;
13 0316  09 0840  0673         END;

                               SEGMENT 13  NAME = SEG#13      LENGTH = 0370  BASE REG = 15

08 03A4  09 0840  0674
08 03A4  09 0840  0675     SEGMENT PROCEDURE ADVANCESYMBOL(R10);
08 03A4  09 0840  0676        BEGIN  COMMENT  FILL NEXTBUF WITH NEXT SYMBOL
14 0000  09 0840  0677                       R7 MUST HAVE BASE ADDRESS OF NEXTBUF;
14 000C  09 0840  0678           SAVER4 := R4;  SAVER10 := R10;  TEST(NEWCARD);  IF  =  THEN
14 0010  09 0840  0679           BEGIN  R1 := NEWCARDCODE;  OUTCODE;  R2 := CARDCOUNT;
14 0024  09 0840  0680              OUTCODE2;  RESET(NEWCARD);
14 0034  09 0840  0681           END;
14 0034  09 0840  0682           R2 := R2 - R2;
14 0036  09 0840  0683     L1:   EX(R9,SCANBLANKS);  IF  =  THEN
14 003E  09 0840  0684           BEGIN  FETCHCARD;  GOTO L1;
14 004C  09 0840  0685           END;
14 004C  09 0840  0686           R8 := R1;  R9 := @INBUF(84) - R8;  NEXTBUFTYPE:= R2;
14 0058  09 0840  0687           CASE R2 OF
```

```
14 0058  09 0840  0688          BEGIN
14 0058  09 0840  0689             BEGIN  COMMENT  LETTER;
14 0060  09 0840  0690                R2 := R2 - R2;  R3 := R2;
14 0064  09 0840  0691    L2:         EX(R9,IDCHARS);  IF  =  THEN
14 006C  09 0840  0692                BEGIN  R4 := @NEXTBUF(R3);  R3 := R3 + R9 + R5;
14 0074  09 0840  0693                   IF R3 > 256 THEN
14 007C  09 0840  0694                   BEGIN  R2 := @NEXTBUF(255) - R4;
14 0082  09 0840  0695                      IF R2 >= 0 THEN EX(R2,MVCCHARS);  R3 := 257;
14 0090  09 0840  0696                   END ELSE
14 0090  09 0840  0697                   EX(R9,MVCCHARS);
14 0098  09 0840  0698                   FETCHCARD;  GOTO L2;
14 00A6  09 0840  0699                END ELSE
14 00A6  09 0840  0700                IF R1 ¬= R8 THEN
14 00B0  09 0840  0701                BEGIN  R4 := @NEXTBUF(R3);  R2 := R1 - R8;  R3 := R3 + R2;
14 00BA  09 0840  0702                   R2 := R2 - R5;
14 00BC  09 0840  0703                   IF R3 > 256 THEN
14 00C4  09 0840  0704                   BEGIN  R2 := @NEXTBUF(255) -R4;
14 00CA  09 0840  0705                      IF R2 >= 0 THEN EX(R2,MVCCHARS);
14 00D4  09 0840  0706                      R3 := 256;  R2 := R1;  LA(R4,13);  ERROR;  R1 := R2;
14 00EC  09 0840  0707                   END ELSE EX(R2,MVCCHARS);
14 00F4  09 0840  0708                END;   R3 := R3 - R5;
14 00F6  09 0840  0709                R8 := R1;  R9 := @INBUF(84) - R8;  NEXTBUFCOUNT := R3;
14 0102  09 0840  0710                IF R3 = 6 THEN
14 010A  09 0840  0711                BEGIN  CLC(6,NEXTBUF,"COMMENT");  IF  =  THEN
14 0114  09 0840  0712                   BEGIN
14 0114  09 0840  0713    L3:               EX(R9,COMMNTCHARS);  IF  =  THEN
14 011C  09 0840  0714                      BEGIN  FETCHCARD;  GOTO L3;
14 012A  09 0840  0715                      END ELSE IF EOF THEN  GOTO L1 ELSE
14 0136  09 0840  0716                      BEGIN  R8 := R1 + R5;  R9 := @INBUF(84) - R8;
14 0140  09 0840  0717                         IF R9 < 0 THEN FETCHCARD;  GOTO L1;
14 0154  09 0840  0718                      END;
14 0154  09 0840  0719                   END;
14 0154  09 0840  0720                END ELSE
14 0154  09 0840  0721                IF R3 = 4 THEN
14 0160  09 0840  0722                BEGIN  CLC(4,NEXTBUF,"BEGIN");  IF = THEN
14 016A  09 0840  0723                   BEGIN  R1 := NESTLEVEL + R5;  NESTLEVEL := R1;
14 0174  09 0840  0724                      IF ¬BEGINSET THEN
14 017C  09 0840  0725                      BEGIN  SET(BEGINSET);  R0 := 0; R1 := ABS R1/10;
14 018A  09 0840  0726                         STC(R0,INBUF(5)); OI("0",INBUF(5));
14 0192  09 0840  0727                      END;
14 0192  09 0840  0728                   END;
14 0192  09 0840  0729                END ELSE
14 0192  09 0840  0730                IF R3 = 2 THEN
14 019E  09 0840  0731                BEGIN  CLC(2,NEXTBUF,"END");  IF = THEN
14 01A8  09 0840  0732                   BEGIN  R0 := 0; R1 := ABS NESTLEVEL / 10;
14 01B6  09 0840  0733                      STC(R0,INBUF(6));  OI("0",INBUF(6));
14 01BE  09 0840  0734                      R1 := NESTLEVEL - R5;  NESTLEVEL := R1;
14 01C8  09 0840  0735                   END;
14 01C8  09 0840  0736                END;
14 01C8  09 0840  0737             END;
14 01C8  09 0840  0738             BEGIN  COMMENT  DIGIT, ' ;
14 01CC  09 0840  0739                NUMBERCONVERT;
14 01D6  09 0840  0740             END;
14 01D6  09 0840  0741             BEGIN  COMMENT  SYMBOL;
14 01DA  09 0840  0742                CLI("#",B8);  IF  =  THEN
14 01E2  09 0840  0743                BEGIN MVC(0,NEXTBUF,B8);  R9 := @INBUF(84);
14 01EC  09 0840  0744                   R1 := R1 - R1;  R0 := R1;
14 01F0  09 0840  0745    L4:            NEXTCHAR;
```

```
14 01FC  09 0840  0746                IF R0 >= "A" AND R0 <= "F" THEN R0 := R0 - #B7  ELSE
14 0210  09 0840  0747                IF R0 <= "9" AND R0 >= "0" THEN R0 := R0 AND #F  ELSE
14 0228  09 0840  0748                BEGIN  R9 := @INBUF(84) - R8;  NEXTBUFCOUNT := R1;
14 0236  09 0840  0749                   GOTO XIT;
14 023A  09 0840  0750                END;
14 023A  09 0840  0751                R1 := R1 + R5;  STC(R0,NEXTBUF(R1));  GOTO L4;
14 0244  09 0840  0752             END;
14 0244  09 0840  0753             R1 := R1 - R1;  NEXTBUFCOUNT := R1;  MVC(0,NEXTBUF,B8);
14 0250  09 0840  0754             IF R9 = 0 THEN FETCHCARD ELSE
14 0260  09 0840  0755             BEGIN  R8 := R8 + R5;  R9 := R9 - R5;
14 0268  09 0840  0756             END;
14 0268  09 0840  0757  XIT:    END;
14 0268  09 0840  0758          BEGIN  COMMENT . ;
14 026C  09 0840  0759             IF R9 = 0 THEN FETCHCARD ELSE
14 027C  09 0840  0760             BEGIN  R8 := R8 + R5;  R9 := @INBUF(84) -R8;
14 0288  09 0840  0761             END;
14 0288  09 0840  0762             R0 := R0 - R0;  IC(R0,B8);
14 028E  09 0840  0763             IF R0 >= "0" THEN
14 0296  09 0840  0764             BEGIN  R8 := R8 - R5;  SAVEADD := R8;  MVI(".",B8);
14 02A0  09 0840  0765                R9 := @INBUF(84) - R8;
14 02A6  09 0840  0766                R1 := 2;  NEXTBUFTYPE := R1;  NUMBERCONVERT;
14 02B8  09 0840  0767                R1 := @INBUF(12); IF R1 = SAVEADD THEN MVI(" ",B1);
14 02C8  09 0840  0768             END ELSE
14 02C8  09 0840  0769             BEGIN  R1 := 3;  NEXTBUFTYPE := R1;
14 02D4  09 0840  0770                R1 := 0;  NEXTBUFCOUNT := R1;  MVI(".",NEXTBUF);
14 02E0  09 0840  0771             END;
14 02E0  09 0840  0772          END;
14 02E0  09 0840  0773       END;
14 02F4  09 0840  0774       R7 := CURBUFBASE;  R4 := SAVER4;  R10 := SAVER10;
14 C300  09 0840  0775    END;

                         SEGMENT 14  NAME = SEG#14      LENGTH = 0350  BASE REG = 15

08 03A4  09 0840  0776
08 03A4  09 0840  0777    PROCEDURE MATCHRESERVED(R10);
08 03A4  09 0840  0778       BEGIN
08 03A4  09 0840  0779       COMMENT MATCH ALPHANUMERIC IN CURBUF WITH RESERVE WORDS
08 03A4  09 0840  0780            IF SUCCESSFUL R3=0 ELSE R3=1
08 03A4  09 0840  0781            R1 RETURNS OUTPUT CODE
08 03A4  09 0840  0782            R2 RETURNS CASE STATEMENT INDEX;
08 03A4  09 0840  0783         R11 := CURBUFCOUNT;  IF R11<1  THEN R3 := 1 ELSE
08 03B4  09 0840  0784         IF R11 >= 9 THEN R3 := 1  ELSE
08 03C4  09 0840  0785         BEGIN  R1 := R11 SHLL 2;  R3 := RESADDR(R1);
08 03D2  09 0840  0786            R2 := @CURBUF;  R0 := CODEADDR(R1);  R1 := R1 SHRL 1;
08 03DE  09 0840  0787            FOR R4:=0 STEP 1 UNTIL RESNO(R1) DO
08 03E2  09 0840  0788            BEGIN  EX(R11,COMP23);
08 03EA  09 0840  0789               IF  =  THEN GOTO L ELSE
08 03EE  09 0840  0790                  IF <  THEN GOTO NO ELSE R3 := R3 + R11 + 1;
08 03F8  09 0840  0791            END;
08 0404  09 0840  0792  NO:       R3 := 1;  GOTO D;
08 040C  09 0840  0793  L:        R4 := R4 SHLL 1;  R3 := R0 + R4;  R1 := 0;  R2 := R1;
08 041A  09 0840  0794            IC(R1,B3);  IC(R2,B3(1));  R3 := 0;
08 0426  09 0840  0795            IF R1 = GOCODE THEN
08 042E  09 0840  0796            BEGIN  R7 := NEXTBUFBASE;
08 0432  09 0840  0797               IF R5 ¬= NEXTBUFCOUNT THEN
08 043A  09 0840  0798               BEGIN  R3 := 1;  R7 := CURBUFBASE;
08 0442  09 0840  0799               END ELSE
08 0442  09 0840  0800               BEGIN  CLC(1,NEXTBUF,TO);
```

```
08 044C  09 0840  0801                  IF = THEN
08 0450  09 0840  0802                  BEGIN  SAVE := R10;  ADVANCESYMBOL;  R10 := SAVE;
08 0462  09 0840  0803                     R1 := GOTOCODE;  R2 := 1;   R3 := 0;
08 046E  09 0840  0804                  END ELSE
08 046E  09 0840  0805                  BEGIN  R3 := 1;  R7 := CURBUFBASE;
08 047A  09 0840  0806                  END;
08 047A  09 0840  0807               END;
08 047A  09 0840  0808            END;
08 047A  09 0840  0809         END;
08 047A  09 0840  0810   D:  END;
08 047C  09 0840  0811
08 047C  09 0840  0812    PROCEDURE FETCHSYMBOL(R4);
08 047C  09 0840  0813       BEGIN  R7 := NEXTBUFBASE;  R2 := _1;  R1 := NEXTBUFTYPE;
08 0488  09 0840  0814          CLI("""",NEXTBUF);  IF  = AND R1 = 3  THEN
08 0498  09 0840  0815          BEGIN  R7 := CURBUFBASE;  R1 := NEXTBUFBASE;  R9 := @INBUF(84);
08 04A4  09 0840  0816             NEXTBUFBASE := R7; CURBUFBASE := R1;  R0 := R0-R0;  IC(R0,B8);
08 04AE  09 0840  0817   QUOTE:    WHILE R0 ¬= """" DO
08 04BA  09 0840  0818             BEGIN  IF R2 >= 255 THEN
08 04C2  09 0840  0819                BEGIN  R0 := R4;  LA(R4,8);  ERROR;  R4 := R0; R0:=R0-R0;
08 04D0  09 0840  0820                   WHILE R0 ¬= """" DO NEXTCHAR;  GOTO XIT;
08 04E4  09 0840  0821                END;
08 04E4  09 0840  0822                R2 := R2 + R5;  STC(R0,NEXTBUF(R2));  NEXTCHAR;
08 04EE  09 0840  0823             END;
08 04F2  09 0840  0824   XIT:      NEXTCHAR;  IF R0 = """" THEN
08 04FE  09 0840  0825             BEGIN  IF R2 < 256 THEN
08 0506  09 0840  0826                BEGIN  R2 := R2 + R5;  STC(R0,NEXTBUF(R2));
08 050C  09 0840  0827                END;
08 050C  09 0840  0828                NEXTCHAR;  GOTO QUOTE;
08 0514  09 0840  0829             END;
08 0514  09 0840  0830             NEXTBUFCOUNT := R2;  NEXTBUFTYPE := R5;
08 051C  09 0840  0831             R7 := CURBUFBASE;  R9 := @INBUF(84) - R8;
08 0526  09 0840  0832          END ELSE
08 0526  09 0840  0833          BEGIN  R7 := CURBUFBASE;  R1 := NEXTBUFBASE;  NEXTBUFBASE := R7;
08 0536  09 0840  0834             CURBUFBASE := R1;  ADVANCESYMBOL;
08 0544  09 0840  0835          END;
08 0544  09 0840  0836       END;
08 0546  09 0840  0837
08 0546  09 0840  0838    SEGMENT PROCEDURE NUMBEROPT(R11);
08 0546  09 0840  0839       BEGIN  COMMENT  R1 CONTAINS TYPE OF NUMBER AT ENTRY AND EXIT;
15 0000  09 0840  0840          PROCEDURE CONVERT1(R3);
15 0000  09 0840  0841             BEGIN  R0 := FIRSTNOI;  IF R0 < 0 THEN
15 000E  09 0840  0842                BEGIN  R0 := NEG R0;  FCON1LOW := R0;  F01 := NEG FCON1;
15 001A  09 0840  0843                END ELSE
15 001A  09 0840  0844                BEGIN  FCON1LOW := R0;  F01 := FCON1;
15 0026  09 0840  0845                END;
15 0026  09 0840  0846                F01 := F01 + 0L;
15 002A  09 0840  0847             END;
15 002C  09 0840  0848
15 002C  09 0840  0849          PROCEDURE CONVERT2(R3);
15 002C  09 0840  0850             BEGIN  R0 := CURBUFI;  IF R0 < 0 THEN
15 0036  09 0840  0851                BEGIN  R0 := NEG R0;  FCON1LOW := R0;  F01 := NEG FCON1;
15 0042  09 0840  0852                END ELSE
15 0042  09 0840  0853                BEGIN  FCON1LOW := R0;  F01 := FCON1;
15 004E  09 0840  0854                END;
15 004E  09 0840  0855                F01 := F01 + 0L;
15 0052  09 0840  0856             END;
15 0054  09 0840  0857
15 0054  09 0840  0858          R2 := R1;  R0 := R0-R0; IC(R0,PROGRAMM1(R6));
```

```
15 005C  09 0840  0859          IF R0 = MINUSCODE THEN
15 0064  09 0840  0860          BEGIN  R7 := NEXTBUFBASE;  R3 := R3-R3; IC(R3,NEXTBUF);
15 006E  09 0840  0861             R7 := CURBUFBASE;  IF R3 ¬= "*" THEN
15 007A  09 0840  0862             BEGIN  R3 := @PROGRAMM3(R6);  CLI(#FE,B3); IF ¬= THEN
15 0086  09 0840  0863                BEGIN  CASE R2 OF
15 0086  09 0840  0864                 BEGIN
15 0086  09 0840  0865                   BEGIN  R0 := NEG CURBUFI; CURBUFI := R0;  END;
15 0093  09 0840  0866                   BEGIN  F0 := NEG CURBUFR; CURBUFR := F0;  END;
15 00A6  09 0840  0867                   BEGIN  F01 := NEG CURBUFLR; CURBUFLR := F01;  END;
15 00B4  09 0840  0868                   BEGIN  F0 := NEG CURBUFR(4); CURBUFR(4) := F0;  END;
15 00C2  09 0840  0869                   BEGIN  F01 := NEG CURBUFLR(8); CURBUFLR(8) := F01;  END;
15 00D0  09 0840  0870                 END;
15 00E8  09 0840  0871                 R0 := PLUSCODE;  STC(R0,PROGRAMM1(R6));
15 00F0  09 0840  0872                END;
15 00F0  09 0840  0873             END;
15 00F0  09 0840  0874          END;
15 00F0  09 0840  0875          IF R6 = IMAGPOS AND R0 = PLUSCODE THEN
15 0100  09 0840  0876          BEGIN  R2 := NUMPOS;  R3 := R3 - R3;  IC(R3,PROGRAMM1(R2));
15 010A  09 0840  0877             IF R3 = LPARENCODE OR R3 = ASSIGNCODE OR R3 = PLUSCODE OR
15 0122  09 0840  0878                R3 = COMMACODE THEN
15 012A  09 0840  0879             BEGIN  R7 := NEXTBUFBASE;  R0 := NEXTBUFTYPE;
15 0132  09 0840  0880                IF R0 = 1 AND R1 = 1 THEN R7 := CURBUFBASE ELSE
15 0146  09 0840  0881             BEGIN  R0 := R0 - R0;  IC(R0,NEXTBUF);
15 0150  09 0840  0882                IF R0 = "*" OR R0 = "/" OR R0 = "¬" THEN R7 := CURBUFBASE
15 0168  09 0840  0883                ELSE
15 016C  09 0840  0884             BEGIN  R6 := R2;  R2 := R2 - R2;  IC(R2,PROGRAM(R6+1));
15 0178  09 0840  0885                R3 := R1;  R7 := CURBUFBASE;
15 017E  09 0840  0886                COMMENT  R2 HAS TYPE OF PREVIOUS NUMBER
15 017E  09 0840  0887                         R3 HAS TYPE OF NUMBER IN CURBUF;
15 017E  09 0840  0888          R4 := @PROGRAM(R6+2);  MVC(15,FIRSTNO,B4);
15 0188  09 0840  0889          CASE R2 OF
15 0188  09 0840  0890          BEGIN
15 0188  09 0840  0891             BEGIN  COMMENT  FIRST NUMBER IS INTEGER;
15 0190  09 0840  0892                CASE R3 OF
15 0190  09 0840  0893                BEGIN
15 0190  09 0840  0894                   BEGIN  R0 := FIRSTNOI + CURBUFI;
15 01A0  09 0840  0895                      CURBUFI := R0;
15 01A4  09 0840  0896                   END;
15 01A4  09 0840  0897                   BEGIN  CONVERT1;  F0 := F0 + CURBUFR;
15 01B0  09 0840  0898                      CURBUFR := F0;
15 01B4  09 0840  0899                   END;
15 01B4  09 0840  0900                   BEGIN  CONVERT1;  F01 := F01 + CURBUFLR;
15 01CC  09 0840  0901                      CURBUFLR := F01;
15 01C4  09 0840  0902                   END;
15 01C4  09 0840  0903                   BEGIN  CONVERT1;  CURBUFR := F0;  END;
15 01D0  09 0840  0904                   BEGIN  CONVERT1;  CURBUFLR := F01;  END;
15 01DC  09 0840  0905                END;
15 01F4  09 0840  0906             END;
15 01F4  09 0840  0907             BEGIN  COMMENT  FIRST IS REAL;
15 01F8  09 0840  0908                CASE R3 OF
15 01F8  09 0840  0909                BEGIN
15 01F8  09 0840  0910                BEGIN  CONVERT2;  F0 := F0 + FIRSTNOR;
15 0208  09 0840  0911                   CURBUFR := F0;  R1 := 2;
15 0210  09 0840  0912                END;
15 0210  09 0840  0913                BEGIN  F0 := FIRSTNOR + CURBUFR;
15 021C  09 0840  0914                   CURBUFR := F0;
15 0220  09 0840  0915                END;
15 0220  09 0840  0916                BEGIN  F01 := CURBUFLR;  F0 := F0 + FIRSTNOR;
```

```
15 022C  09 0840  0917                    CURBUFR := F0;  R1 := 2;
15 0234  09 0840  0918                 END;
15 0234  09 0840  0919                 BEGIN  F0 := FIRSTNOR;  CURBUFR := F0;  END;
15 0240  09 0840  0920                 BEGIN  F0 := FIRSTNOR;  CURBUFR := F0;
15 024C  09 0840  0921                    F01 := CURBUFLR(8);  CURBUFR(4) := F0;
15 0254  09 0840  0922                    R1 := 4;
15 0258  09 0840  0923                 END;
15 0258  09 0840  0924                 END;
15 0270  09 0840  0925              END;
15 0270  09 0840  0926           BEGIN  COMMENT  FIRST NUMBER IS LONG REAL;
15 0274  09 0840  0927              CASE R3 OF
15 0274  09 0840  0928              BEGIN
15 0274  09 0840  0929                 BEGIN  CONVERT2;  F01 := F01 + FIRSTNOLR;
15 0284  09 0840  0930                    CURBUFLR := F01;  R1 := 3;
15 028C  09 0840  0931                 END;
15 028C  09 0840  0932                 BEGIN  F01 := FIRSTNOLR;  F0 := F0 + CURBUFR;
15 0298  09 0840  0933                    CURBUFR := F0;
15 029C  09 0840  0934                 END;
15 029C  09 0840  0935                 BEGIN  F01 := CURBUFLR + FIRSTNOLR;
15 02A8  09 0840  0936                    CURBUFLR := F01;
15 02AC  09 0840  0937                 END;
15 02AC  09 0840  0938                 BEGIN  F01 := FIRSTNOLR;  CURBUFR := F0;  END;
15 02B8  09 0840  0939                 BEGIN  F01 := FIRSTNOLR;  CURBUFLR := F01;  END;
15 02C4  09 0840  0940              END;
15 02DC  09 0840  0941           END;
15 02DC  09 0840  0942           BEGIN  COMMENT  FIRST NUMBER IS COMPLEX;
15 02E0  09 0840  0943              CASE R3 OF
15 02E0  09 0840  0944              BEGIN
15 02E0  09 0840  0945                 BEGIN  CONVERT2;  F0 := F0 + FIRSTNOR;
15 02F0  09 0840  0946                    CURBUFR := F0;  F0 := FIRSTNOR(4);
15 02F8  09 0840  0947                    CURBUFR(4) := F0;  R1 := 4;
15 03C0  09 0840  0948                 END;
15 0300  09 0840  0949                 BEGIN  F0 := FIRSTNOR + CURBUFR;  CURBUFR := F0;
15 0310  09 0840  0950                    F0 := FIRSTNOR(4);  CURBUFR(4) := F0;  R1 := 4;
15 031C  09 0840  0951                 END;
15 031C  09 0840  0952                 BEGIN  F01 := CURBUFLR; F0 := F0 + FIRSTNOR;
15 0328  09 0840  0953                    CURBUFR := F0;  F0 := FIRSTNOR(4);
15 0330  09 0840  0954                    CURBUFR(4) := F0;  R1 := 4;
15 0338  09 0840  0955                 END;
15 0338  09 0840  0956                 BEGIN  F0 := CURBUFR + FIRSTNOR;  CURBUFR := F0;
15 0348  09 0840  0957                    F0 := CURBUFR(4) + FIRSTNOR(4);  CURBUFR(4):=F0;
15 0354  09 0840  0958                 END;
15 0354  09 0840  0959                 BEGIN  F01 := CURBUFLR;  F0 := F0 + FIRSTNOR;
15 0360  09 0840  0960                    CURBUFR := F0;  F01 := CURBUFLR(8);
15 0368  09 0840  0961                    F0 := F0 + FIRSTNOR(4);  CURBUFR(4) := F0;
15 0370  09 0840  0962                    R1 := 4;
15 0374  09 0840  0963                 END;
15 0374  09 0840  0964              END;
15 038C  09 0840  0965           END;
15 038C  09 0840  0966           BEGIN  COMMENT FIRST NUMBER IS LONG COMPLEX;
15 0390  09 0840  0967              CASE R3 OF
15 0390  09 0840  0968              BEGIN
15 0390  09 0840  0969                 BEGIN  CONVERT2;  F01 := F01 + FIRSTNOLR;
15 03A0  09 0840  0970                    CURBUFLR := F01;  F01 := FIRSTNOLR(8);
15 03A8  09 0840  0971                    CURBUFLR(8) := F01;  R1 := 5;
15 03B0  09 0840  0972                 END;
15 03B0  09 0840  0973                 BEGIN  F01 := FIRSTNOLR;  F0 := F0 + CURBUFR;
15 03BC  09 0840  0974                    CURBUFR := F0;
```

```
15 03C0  09 0840  0975                        F01 := FIRSTNOLR(8);  CURBUFR(4) := F0;  R1 := 4;
15 03CC  09 0840  0976                     END;
15 03CC  09 0840  0977                     BEGIN  F01 := FIRSTNOLR + CURBUFLR;
15 03D8  09 0840  0978                        CURBUFLR := F01;  F01 := FIRSTNOLR(8);
15 03E0  09 0840  0979                        CURBUFLR(8) := F01;  R1 := 5;
15 03E8  09 0840  0980                     END;
15 03E8  09 0840  0981                     BEGIN  F01 := FIRSTNOLR;  F0 := F0 + CURBUFR;
15 03F4  09 0840  0982                        CURBUFR := F0;
15 03F8  09 0840  0983                        F01 := FIRSTNOLR(8);  F0 := F0 + CURBUFR(4);
15 0400  09 0840  0984                        CURBUFR(4) := F0;
15 0404  09 0840  0985                     END;
15 0404  09 0840  0986                     BEGIN  F01 := CURBUFLR + FIRSTNOLR;
15 0410  09 0840  0987                        CURBUFLR := F01;
15 0414  09 0840  0988                        F01 := CURBUFLR(8) + FIRSTNOLR(8);
15 041C  09 0840  0989                        CURBUFLR(8) := F01;
15 0420  09 0840  0990                     END;
15 0420  09 0840  0991                  END;
15 0438  09 0840  0992               END;
15 0438  09 0840  0993            END;
15 0450  09 0840  0994         END;
15 0450  09 0840  0995      END;
15 0450  09 0840  0996      END;
15 0450  09 0840  0997      END;
15 0450  09 0840  0998      END;

                  SEGMENT 15  NAME = SEG#15      LENGTH = 0470  BASE REG = 15

08 0546  09 0840  0999
08 0546  09 0840  1000    PROCEDURE MOVETABLE(R10);
08 0546  09 0840  1001     BEGIN  R0 := 256;
08 054A  09 0840  1002       WHILE R3 >= R0 DO
08 0550  09 0840  1003       BEGIN  MVC(255,B1,B2); R1 := R1 + R0; R2 := R2 + R0;
08 055A  09 0840  1004          R3 := R3 - R0;
08 055C  09 0840  1005       END;
08 0560  09 0840  1006       IF R3 ¬= 0 THEN
08 0566  09 0840  1007       BEGIN  R3 := R3 - 1; EX(R3,MOVE12); R1 := R1 + R3 + 1;
08 0574  09 0840  1008       END;
08 0574  09 0840  1009     END;
08 0576  09 0840  1010
08 0576  09 0840  1011    SEGMENT PROCEDURE INITIALIZE(R9);
08 0576  09 0840  1012     BEGIN
16 0000  09 0840  1013       R0 := 6144; R1 := R0; R3 := AGETMAIN; BALR(R2,R3); COMMENT GETMAIN;
16 000C  09 0840  1014       CHAIN := R0;  STM(R0,R1,CHAININFO);   R0 := R0-R0;
16 0016  09 0840  1015       R7 := @SYMBUFFERS;  CURBUFBASE := R7;  MVI(#00,CURBUF);
16 0022  09 0840  1016       CURBUFTYPE := R0;  R7 := @B7(264);  NEXTBUFBASE := R7;
16 002E  09 0840  1017       MVI(#00,NEXTBUF);  NEXTBUFTYPE := R0;
16 0036  09 0840  1018       LM(R0,R1,COMMLIM); R11 := R0;  COMMENT OBTAIN COMMON LIMITS;
16 003C  09 0840  1019       R6 := R1 - R0 + 1;  COMMENT COMMON SIZE;
16 0044  09 0840  1020       R1 := R1 - 20; PROGLIMIT := R1;
16 004C  09 0840  1021       R0 := ISYMBOLINDEX-12 SHLL 16 OR 12; BLOCKLIST(0) := R0;
16 0060  09 0840  1022       R3 := 511; BLLIMIT := R3;
16 0068  09 0840  1023       R7 := @NAMETABLE; R2 := @NAMETFILL; R1 := R7;
16 0072  09 0840  1024       R3 := ISYMBOLINDEX; SYMBOLINDEX := R3; MOVETABLE;
16 0086  09 0840  1025       FOR R5 := 540 STEP 12 UNTIL 708 DO
16 008A  09 0840  1026       BEGIN  R0 := NAMETABLE(R5) AND #FFF OR #D0000;
16 009A  09 0840  1027          NAMETABLE(R5) := R0;
16 009E  09 0840  1028       END;
16 00AA  09 0840  1029       R6 := @B11(R6) - R7; IF R6 > #2AAA8 THEN R6 := #2AAA8;
```

```
16 00BC  09 0840   1030      COMMENT  R6 HAS SIZE OF DYNAMICALLY ALLOCATED COMMON;
16 00BC  09 0840   1031      R5 := R6 SHRL 4 * 3S AND #FFFFF8;  R7 := R7 + R5;
16 00CC  09 0840   1032      R0 := R5 - 16;  SCRATCHINDEX := R0;
16 00D6  09 0840   1033      REFRECBASE := R7; RFRCLISTBASE := R7; XC(1,B7,B7);
16 00E4  09 0840   1034      R5 := R6 SHRL 5 AND #FFFFFC;  R7 := R7 + R5;
16 00F0  09 0840   1035      IDDBASE := R7; IDDIRBASE := R7;
16 00F8  09 0840   1036      R1 := R7; R2 := @IDDIRFILL; R3 := IIDDIRINDEX;
16 0102  09 0840   1037      R0 := R3 - 4; IDDIRINDEX := R0; MOVETABLE;
16 0118  09 0840   1038      R5 := R5 + R5; IDDIRLIMIT := R5; R7 := R7 + R5;
16 0120  09 0840   1039      IDLISTBASE := R7;  IDLBASE := R7;
16 0128  09 0840   1040      R1 := R7; R2 := @IDLISTFILL;
16 012E  09 0840   1041      R3 := IIDLISTINDEX; IDLISTINDEX := R3; MOVETABLE;
16 0142  09 0840   1042      R5 := R5 + R5; IDLISTLIMIT := R5;
16 0148  09 0840   1043      R7 := @B7(R5+48); INPOINT := R7; R7 := R7 - 4; PRBASE := R7;
16 0158  09 0840   1044      RESET(NOGO); RESET(TRACE); SET(CHECKFLAG);
16 0164  09 0840   1045      SET(CARDFLAG);  XC(7,TRACEBITS,TRACEBITS);
16 016E  09 0840   1046      XC(15,FLAGS,FLAGS); XC(3,PARD,PARD); SET(LISTFLAG);
16 017E  09 0840   1047      R0 := _5S; ERRORCOUNT := R0; R0 := 195; ERRORLIMIT := R0;
16 018E  09 0840   1048      R0 := 0; STACKINDEX := R0; BLOCKNO := R0; CARDCOUNT := R0;
16 019E  09 0840   1049      RLISTPOINT := R0; DIMCNT := R0; TYPE := R0;
16 01AA  09 0840   1050      NESTLEVEL := R0;  RESET(LINEHOLD);
16 01B2  09 0840   1051      R0 := 1; RECORDNO := R0;
16 01BA  09 0840   1052      F01 := #4E0000000000000L; FCON1 := F01;
16 01C2  09 0840   1053      F01 := #4700000000000000L; FCON2 := F01;
16 01CA  09 0840   1054      R0 := @RESERVED2;  R1 := @RESERVED3;  R2 := @RESERVED4;
16 01D6  09 0840   1055      R3 := @RESERVED5;  R4 := @RESERVED6;  R5 := @RESERVED7;
16 01E2  09 0840   1056      R6 := @RESERVED8;  R7 := @RESERVED9;  STM(R0,R7,RESADDR(4));
16 01EE  09 0840   1057      R0 := @CODE2; R1 := @CODE3; R2 := @CODE4; R3 := @CODE5;
16 01FE  09 0840   1058      R4 := @CODE6; R5 := @CODE7; R6 := @CODE8; R7 := @CODE9;
16 020E  09 0840   1059      STM(R0,R7,CODEADDR(4));
16 0212  09 0840   1060      MVI(" ",INBUF);  MVC(130,INBUF(1),INBUF);
16 021C  09 0840   1061      XC(159,HASHID,HASHID); MVC(159,HASHID(160),HASHID);
16 0228  09 0840   1062      R7 := CURBUFBASE; R12 := IDDIRBASE; R3 := R3-R3;
16 0232  09 0840   1063      FOR R2 := 4 STEP 4 UNTIL IDDIRINDEX DO
16 0236  09 0840   1064      BEGIN  R4 := IDDIRLNG(R2);  R1 := IDDIRIND(R2) + IDLISTBASE;
16 0246  09 0840   1065         IC(R1,B1);  HASH;  R3 := R3 + 2;  R1 := R1 SHLL 1;
16 025E  09 0840   1066         R11 := CHAIN;  R4 := HASHID(R1);  CHAINID(R3) := R4;
16 026A  09 0840   1067         HASHID(R1) := R3;
16 026E  09 0840   1068      END;
16 027A  09 0840   1069      R6 := PRBASE;  R5 := 1;
16 0282  09 0840   1070    END;

                          SEGMENT 16  NAME = SEG#16      LENGTH = 02D8  BASE REG = 15

08 0576  09 0840   1071
08 0576  09 0840   1072    SEGMENT PROCEDURE  CLEANUP(R9);
08 0576  09 0840   1073     BEGIN
17 0000  09 0840   1074       IF LISTFLAG AND LINEHOLD THEN
17 0010  09 0840   1075       BEGIN  RESET(LINEHOLD);
17 0014  09 0840   1076          R0 := @INBUF;  PRINT;
17 0024  09 0840   1077       END;
17 0024  09 0840   1078       SET(ENDDOT);  R1 := STACKINDEX;
17 002C  09 0840   1079       WHILE R1 > 0 DO
17 0032  09 0840   1080       BEGIN  CLOSEBLOCK;  R4 :=  3;  ERROR;    COMMENT   ERROR NO  3;
17 004E  09 0840   1081          R1 := STACKINDEX;
17 0052  09 0840   1082       END;
17 0056  09 0840   1083       MVI(" ",OUTPUT);  MVC(130,OUTPUT(1),OUTPUT);
17 0060  09 0840   1084       R1 := 0;  R11 := NTBASE;  IF R1  <=  ERRORCOUNT THEN
```

```
17 0070  09 0840  1085      BEGIN  R4 := @ERRORS;  R3 := R4 + ERRORCOUNT;
17 007A  09 0840  1086         MVI("1",CARRCONT); MVC(17,OUTPUT,"CARD NO.        --"); R1 := 0;
17 0084  09 0840  1087         FOR R4 := R4 STEP 5 UNTIL R3 DO
17 0088  09 0840  1088         BEGIN  R1 := R1-R1;  IC(R1,B4);  MVC(3,OUTPUT(10),B4(1));
17 0098  09 0840  1089            CASE R1 OF
17 0098  09 0840  1090            BEGIN
17 0098  09 0840  1091               MVC(17,OUTPUT(20),"INCORRECT SPECIFTN");
17 00A6  09 0840  1092               MVC(17,OUTPUT(20),"INCORRECT CONSTANT");
17 00B0  09 0840  1093               MVC(17,OUTPUT(20),"MISSING END       ");
17 00BA  09 0840  1094               MVC(17,OUTPUT(20),"MISSING BEGIN     ");
17 00C4  09 0840  1095               MVC(17,OUTPUT(20),"MISSING )         ");
17 00CE  09 0840  1096               MVC(17,OUTPUT(20),"ILLEGAL CHARACTER ");
17 00D8  09 0840  1097               MVC(17,OUTPUT(20),"MISSING FINAL .   ");
17 00E2  09 0840  1098               MVC(17,OUTPUT(20),"STRING LNGTH ERROR");
17 00EC  09 0840  1099               MVC(17,OUTPUT(20),"BITS LENGTH ERROR ");
17 00F6  09 0840  1100               MVC(17,OUTPUT(20),"MISSING (         ");
17 0100  09 0840  1101               MVC(17,OUTPUT(20),"TOO MANY ERRORS   ");
17 010A  09 0840  1102               MVC(17,OUTPUT(20),"TABLE OVERFLOW    ");
17 0114  09 0840  1103               MVC(17,OUTPUT(20),"ID LENGTH > 256   ");
17 011E  09 0840  1104            END;
17 0156  09 0840  1105            R0 := @OUTPUT;  PRINT;
17 0166  09 0840  1106         END;
17 0170  09 0840  1107      END ELSE
17 0170  09 0840  1108      IF ¬ CHECKFLAG THEN
17 017C  09 0840  1109      BEGIN  R0 := @OUTPUT; OI("0",CARRCONT);
17 0184  09 0840  1110         MVC(23,OUTPUT,"NOCHECK OPTION SPECIFIED"); PRINT;
17 0196  09 0840  1111      END;
17 0196  09 0840  1112      R3 := BLOCKNO SHLL 2; BLOCKLISTSIZE := R3;
17 01A2  09 0840  1113      R3 := SYMBOLINDEX - 12;  NAMETABLESIZE := R3;
17 01AE  09 0840  1114      R1 := @NAMETABLE + SYMBOLINDEX + 19 AND #FFFFF8; R2 := REFRECBASE;
17 01C2  09 0840  1115      R3 := RLISTPOINT + 2; REFRECBASE := R1; MOVETABLE;
17 01DA  09 0840  1116      R1 := R1 + 3 AND #FFFFFC; R2 := IDDBASE;
17 01E6  09 0840  1117      R3 := IDDIRINDEX + 4; IDDBASE := R1; MOVETABLE;
17 01FE  09 0840  1118      R2 := IDLBASE;
17 0202  09 0840  1119      R3 := IDLISTINDEX; IDLBASE := R1; MOVETABLE;
17 0216  09 0840  1120      R1 := R1 + 3 AND #FFFFFC + 48; R2 := INPOINT;
17 0226  09 0840  1121       COMMENT PASS 2 LITERALS START 48 BYTES BEFORE PASS 1 OUTPUT;
17 0226  09 0840  1122      R3 := R6 + 11 AND #FFFFF8 - INPOINT; INPOINT := R1; MOVETABLE;
17 0244  09 0840  1123      R1 := R1 + 7 AND #FFFFF8; TREEBASE := R1;
17 0250  09 0840  1124      IC(R5,TRACEFLAG);  IF R5 = 8 THEN
17 025C  09 0840  1125      BEGIN  R0 := INPOINT; R1 := TREEBASE - R0;
17 0266  09 0840  1126         MVI("1",CARRCONT); DUMP; R14 := SAVE14;
17 027A  09 0840  1127      END;
17 027A  09 0840  1128      LM(R0,R1,CHAININFO); R3 := AFREEMAIN; BALR(R2,R3);
17 0284  09 0840  1129      TEST(NOPASSTWO);  IF  =  THEN R0 := 1 ELSE R0 := 0;
17 0298  09 0840  1130    END;

                         SEGMENT 17  NAME = SEG#17      LENGTH = 03F0  BASE REG = 15

08 0576  09 0840  1131
08 0576  09 0840  1132    BYTE SEGMENTEND;  COMMENT  *** LAST DECLARATION IN SEGMENT *** ;
08 0576  09 0841  1133
08 0576  09 0841  1134       SAVE14 := R14; RETADDR := R10;   INITIALIZE;
08 0588  09 0841  1135       R9 := _1;  FETCHCARD;  FETCHSYMBOL;
08 059A  09 0841  1136
08 059A  09 0841  1137       COMMENT  BEGIN MAIN ROUTINE;
08 059A  09 0841  1138
08 059A  09 0841  1139    L:  FETCHSYMBOL;  R3 := CURBUFTYPE;
```

```
08 05A2  09 0841  1140      IF R6 >= PROGLIMIT THEN STOP;  COMMENT PROGRAM TOO LONG;
08 05AE  09 0841  1141      CASE R3 OF BEGIN
08 05AE  09 0841  1142      BEGIN MATCHRESERVED;
08 05BA  09 0841  1143         IF R5=R3 THEN
08 05C0  09 0841  1144         BEGIN IDSEARCH; GOTO L;
08 05C8  09 0841  1145         END;
08 05C8  09 0841  1146      END;
08 05C8  09 0841  1147      BEGIN
08 05CC  09 0841  1148         R1 := CURBUFCOUNT;  NUMBEROPT;  CURBUFCOUNT := R1; NUMPOS := R6;
08 05E2  09 0841  1149         R1 := NUMBERCODE;  OUTCODE;  R1 := CURBUFCOUNT;  OUTCODE;
08 05F2  09 0841  1150         CASE R1 OF
08 05F2  09 0841  1151         BEGIN
08 05F2  09 0841  1152            BEGIN  R2 := CURBUFI;  OUTCODE4;  END;
08 0602  09 0841  1153            BEGIN  R2 := CURBUFI;  OUTCODE4;  END;
08 060E  09 0841  1154            BEGIN  R2 := CURBUFI(4);  OUTCODE4;
08 061A  09 0841  1155               R2 := CURBUFI(8);  OUTCODE4;
08 0622  09 0841  1156            END;
08 0622  09 0841  1157            BEGIN R2 := CURBUFI; OUTCODE4;  R2 := CURBUFI(4);  OUTCODE4;
08 0636  09 0841  1158            END;
08 0636  09 0841  1159            BEGIN  FOR R3 := 4 STEP 4 UNTIL 16 DO
08 063E  09 0841  1160               BEGIN  R2 := CURBUFI(R3);  OUTCODE4;  END;
08 0656  09 0841  1161            END;
08 0656  09 0841  1162         END;
08 066E  09 0841  1163         R2 := R6 + R5;  IMAGPOS := R2;  GOTO L;
08 067A  09 0841  1164      END;
08 067A  09 0841  1165      BEGIN
08 067E  09 0841  1166         MATCHSYMBOL;  IF R3 ¬= R5 THEN
08 0688  09 0841  1167         BEGIN  R4 := 6;  ERROR;  GOTO L;
08 0694  09 0841  1168         END;
08 0694  09 0841  1169      END;
08 0694  09 0841  1170      END;
08 06A4  09 0841  1171   L1: CASE R2 OF
08 06A4  09 0841  1172      BEGIN
08 06A4  09 0841  1173      OUTCODE; COMMENT  SYMBOLS NEEDING NO PROCESSING;                    1
08 06B0  09 0841  1174      BEGIN    COMMENT  BITS SIGN # ;                                     2
08 06B4  09 0841  1175         OUTCODE;  R1 := CURBUFCOUNT;  R0 := 0;  R2 := R0;
08 06C2  09 0841  1176         IF R1 = 0 OR R1 > 8 THEN
08 06D0  09 0841  1177         BEGIN  R4 := 9;  ERROR;  END ELSE
08 06D8  09 0841  1178         BEGIN  IC(R2,CURBUF(1));
08 06EC  09 0841  1179            FOR R3 := 2 STEP 1 UNTIL R1 DO
08 06E4  09 0841  1180            BEGIN  IC(R0,CURBUF(R3));  R2 := R2 SHLL 4 OR R0;
08 06F2  09 0841  1181            END;
08 06FC  09 0841  1182         END;
08 06FC  09 0841  1183         OUTCODE4;
08 070C  09 0841  1184      END;
08 0700  09 0841  1185      BEGIN    COMMENT  STRING QUOTE " ;                                  3
08 0704  09 0841  1186         R7 := NEXTBUFBASE;  OUTCODE;  R1 := NEXTBUFCOUNT;
08 0710  09 0841  1187         IF R1 = _1 THEN
08 0718  09 0841  1188         BEGIN  R4 := 8;  ERROR;  R1 := R1 - R1;  NEXTBUFCOUNT := R1;
08 0726  09 0841  1189         END;
08 0726  09 0841  1190         OUTCODE;  OUTSTRING;  ADVANCESYMBOL;
08 0738  09 0841  1191      END;
08 0738  09 0841  1192      BEGIN    COMMENT  :  ;                                              4
08 073C  09 0841  1193         R7 := NEXTBUFBASE;  CLI("=",NEXTBUF);
08 0744  09 0841  1194         IF = THEN
08 0748  09 0841  1195         BEGIN R1 := ASSIGNCODE;  OUTCODE;  ADVANCESYMBOL;
08 075A  09 0841  1196         END ELSE
08 075A  09 0841  1197         BEGIN  CLI(":",NEXTBUF);
```

```
08 0762  09 0841  1198          IF  =  THEN
08 0766  09 0841  1199          BEGIN  R1 := SPECCOLON;  OUTCODE;  ADVANCESYMBOL;
08 0778  09 0841  1200          END ELSE
08 0778  09 0841  1201          BEGIN  OUTCODE;  R1 := 0;  IC(R1,PROGRAMM3(R6-1));
08 0788  09 0841  1202             IF R1 = IDCODE THEN
08 0790  09 0841  1203             BEGIN  R1 := #100;  TYPE := R1;  R1 := 0;
08 079C  09 0841  1204                SIMTYPEINFO := R1;  IC(R2,PROGRAMM3(R6));
08 07A4  09 0841  1205                R2 := R2 SHLL 8;  IC(R2,PROGRAMM2(R6));  INSYMBOL;
08 07B0  09 0841  1206             END;
08 07B0  09 0841  1207          END;
08 07B0  09 0841  1208       END;
08 07B0  09 0841  1209    END;
08 07BC  09 0841  1210    BEGIN    COMMENT  *  ;                                    5
08 07B4  09 0841  1211       R7 := NEXTBUFBASE;  CLI("*",NEXTBUF);
08 07BC  09 0841  1212       IF = THEN
08 07C0  09 0841  1213       BEGIN  R1 := EXPONENT;  OUTCODE;  ADVANCESYMBOL;
08 07D2  09 0841  1214       END ELSE
08 07D2  09 0841  1215       BEGIN  OUTCODE;  R7 := CURBUFBASE;
08 07DE  09 0841  1216          CLI(#01,FARRAYFLAG);  IF  =  THEN
08 07E6  09 0841  1217          BEGIN  R1 := DIMCNT + 1;  DIMCNT := R1;  END;
08 07F2  09 0841  1218       END;
08 07F2  09 0841  1219    END;
08 07F2  09 0841  1220    BEGIN    COMMENT  (  ;                                    6
08 07F6  09 0841  1221       CLI(#01,ARRAYFLAG);
08 07FA  09 0841  1222       IF  =  THEN
08 07FE  09 0841  1223       BEGIN  RESET(ARRAYFLAG);  RESET(DECLARFLAG);
08 08C6  09 0841  1224          CLI(#01,PROCFLAG);  IF  =  THEN MVI(#01,FARRAYFLAG);
08 0812  09 0841  1225       END;
08 0812  09 0841  1226       OUTCODE;
08 0816  09 0841  1227    END;
08 0816  09 0841  1228    BEGIN    COMMENT  );                                      7
08 081A  09 0841  1229       OUTCODE;
08 081E  09 0841  1230       IF R5 <= PARD THEN
08 0826  09 0841  1231       BEGIN  CLI(#01,REFERFLAG);
08 082A  09 0841  1232          IF  =  THEN
08 082E  09 0841  1233          BEGIN  RESET(REFERFLAG);  R1 := RLISTPOINT + 2;
08 083A  09 0841  1234             RLISTPOINT := R1; R2 := R2-R2;
08 0840  09 0841  1235             R1 := R1 + RFRCLISTBASE; REFRCDLIST := R2;
08 0848  09 0841  1236             IF R1 >= IDDIRBASE THEN STOP;
08 0854  09 0841  1237          END ELSE
08 0854  09 0841  1238          BEGIN  CLI(#01,RECORDFLAG);  IF  =  THEN
08 0860  09 0841  1239             BEGIN  RESET(RECORDFLAG);
08 0864  09 0841  1240                RESET(DECLARFLAG);
08 0868  09 0841  1241             END ELSE
08 0868  09 0841  1242             BEGIN  CLI(#01,PROCFLAG);   IF  =  THEN
08 0874  09 0841  1243                BEGIN  CLI(#01,FARRAYFLAG);  IF  =  THEN
08 087C  09 0841  1244                   BEGIN  RESET(FARRAYFLAG);  R1 := DIMCNT;
08 0884  09 0841  1245                      R11 := NTBASE;
08 0888  09 0841  1246                      FOR R2 := SCRATCHINDEX + 12 STEP 12 UNTIL
08 0890  09 0841  1247                      FARRAYPNT DO STC(R1,TYPEINFO(R2+1));
08 08A4  09 0841  1248                      R1 := R1 - R1;  DIMCNT := R1;
08 08AA  09 0841  1249                   END ELSE
08 08AA  09 0841  1250                   BEGIN  CLOSEBLOCK;  RESET(PROCFLAG);
08 08B6  09 0841  1251                   END;
08 08B6  09 0841  1252                END;
08 08B6  09 0841  1253             END;
08 08B6  09 0841  1254          END;
08 08B6  09 0841  1255       END;
```

```
C8 08B6  09 0841  1256      END;
08 08B6  09 0841  1257      BEGIN    COMMENT  ,  ;                                          8
08 08BA  09 0841  1258         CLI(#01,DECLARFLAG);  IF  =  THEN R1 := SPECCOMMA;
08 08C6  09 0841  1259         OUTCODE;
08 08CA  09 0841  1260      END;
08 08CA  09 0841  1261      BEGIN    COMMENT  SEMICOLON;                                    9
08 08CE  09 0841  1262         OUTCODE; IF R5<=PARD THEN RESET(DECLARFLAG);
08 08DE  09 0841  1263         RESET(REFERFLAG);  RESET(ARRAYFLAG);  RESET(FARRAYFLAG);
08 08EA  09 0841  1264         R1 := 0;  SIMTYPEINFO := R1;  SETTYPE;  TYPE := R1;
08 08FA  09 0841  1265      END;
08 08FA  09 0841  1266      BEGIN    COMMENT  FOR;                                          10
08 08FE  09 0841  1267         OUTCODE;  OPENBLOCK;  R7 := NEXTBUFBASE;
08 090A  09 0841  1268         IF R5 = NEXTBUFTYPE THEN
08 0912  09 0841  1269         BEGIN  MATCHRESERVED;  IF R3 = 1 THEN
08 091E  09 0841  1270            BEGIN  IDSEARCH;  R3 := #601;  TYPE := R3;  INSYMBOL;
08 092E  09 0841  1271               ADVANCESYMBOL;  R3 := R3 - R3;  TYPE := R3;
08 093E  09 0841  1272            END;
08 093E  09 0841  1273         END;
08 093E  09 0841  1274         CLOSEBLOCK;  R7 := CURBUFBASE;
08 0946  09 0841  1275      END;
08 0946  09 0841  1276      BEGIN    COMMENT  LONG  ;                                       11
08 094A  09 0841  1277         R7 := NEXTBUFBASE;  R3 := NEXTBUFCOUNT;  R4 := NEXTBUFTYPE;
08 0956  09 0841  1278         CASE R4 OF
08 0956  09 0841  1279         BEGIN
08 0956  09 0841  1280         BEGIN  IF R3 = 3 THEN
08 0966  09 0841  1281            BEGIN  CLC(3,NEXTBUF,"REAL");  IF  =  THEN
08 0970  09 0841  1282               BEGIN  MVI(#01,DECLARFLAG);  R1 := 3;  SETTYPE;
08 097C  09 0841  1283                  TYPE := R1;  ADVANCESYMBOL;  R1 := SIMPLETYPE;
08 098E  09 0841  1284               END;
08 098E  09 0841  1285            END ELSE IF R3 = 6 THEN
08 099A  09 0841  1286            BEGIN CLC(6,NEXTBUF,"COMPLEX");  IF = THEN
08 09A4  09 0841  1287               BEGIN MVI(#01,DECLARFLAG);  R1 := 5;  SETTYPE;
08 09B0  09 0841  1288                  TYPE := R1;  ADVANCESYMBOL;  R1 := SIMPLETYPE;
08 09C2  09 0841  1289               END;
08 09C2  09 0841  1290            END;
08 09C2  09 0841  1291            OUTCODE;
08 09C6  09 0841  1292         END;
08 09C6  09 0841  1293         OUTCODE;
08 09CE  09 0841  1294         OUTCODE;
08 09D6  09 0841  1295         END;
08 09E6  09 0841  1296         R7 := CURBUFBASE;
08 09EA  09 0841  1297      END;
08 09EA  09 0841  1298      BEGIN    COMMENT  BEGIN ;                                       12
08 09EE  09 0841  1299         OUTCODE;  OPENBLOCK;
08 09F6  09 0841  1300      END;
08 09F6  09 0841  1301      BEGIN    COMMENT  END;                                          13
08 09FA  09 0841  1302         OUTCODE;
08 09FE  09 0841  1303         R7 := NEXTBUFBASE;
08 0AC2  09 0841  1304         IF R5 = NEXTBUFTYPE THEN
08 0AC6  09 0841  1305         BEGIN MATCHRESERVED;  IF R3 = 1 THEN
08 0A16  09 0841  1306            BEGIN ADVANCESYMBOL;  R7 := NEXTBUFBASE;
08 0A24  09 0841  1307            END;
08 0A24  09 0841  1308         END;
08 0A24  09 0841  1309         R1 := NEXTBUFTYPE; IF R1 = 3 THEN
08 0A30  09 0841  1310         BEGIN  CLI(".",NEXTBUF); IF = THEN SET(EFLAG) ELSE RESET(EFLAG);
08 0A44  09 0841  1311         END;
08 0A44  09 0841  1312         IF EOF OR EFLAG THEN
08 0A54  09 0841  1313         BEGIN  SET(ENDDOT); CLOSEBLOCK;
```

```
08 0A5C  09 0841  1314          IF ¬EFLAG THEN
08 0A64  09 0841  1315          BEGIN  R4 := 7; ERROR;
08 0A6C  09 0841  1316          END;
08 0A6C  09 0841  1317          GOTO ENDPROGRAM;
08 0A70  09 0841  1318       END;
08 0A70  09 0841  1319       CLOSEBLOCK;  R7 := CURBUFBASE;
08 0A78  09 0841  1320    END;
08 0A78  09 0841  1321    BEGIN    COMMENT  INTEGER,REAL,COMPLEX,LOGICAL;                14
08 0A7C  09 0841  1322             COMMENT  R1 CONTAINS CODE FOR TYPE;
08 0A7C  09 0841  1323       MVI(#01,DECLAREFLAG);
08 0A80  09 0841  1324       IF R5<PARD THEN SETTYPE;
08 0A8C  09 0841  1325       R0 := 0;  SIMTYPEINFO := R0;
08 0A94  09 0841  1326       TYPE := R1;  R1 := SIMPLETYPE;  OUTCODE;
08 0AAC  09 0841  1327    END;
08 0AA0  09 0841  1328    BEGIN    COMMENT  BITS;                                       15
08 0AA4  09 0841  1329       MVI(#01,DECLAREFLAG);  SETTYPE;  TYPE := R1;
08 0AB0  09 0841  1330       R1 := SIMPLETYPE;  OUTCODE;
08 0AB8  09 0841  1331       R7 := NEXTBUFBASE;  R1 := NEXTBUFTYPE;  CLI("(",NEXTBUF);
08 0AC4  09 0841  1332       IF  = AND R1 = 3 THEN
08 0ADC  09 0841  1333       BEGIN  R7 := CURBUFBASE;
08 0AD4  09 0841  1334          FETCHSYMBOL;  R7 := NEXTBUFBASE;  R2 := 32;
08 0AE0  09 0841  1335          IF R2 ¬= NEXTBUFI THEN
08 0AE8  09 0841  1336          BEGIN R4 := 1;  ERROR;
08 0AF0  09 0841  1337          END;
08 0AF0  09 0841  1338          R0 := R0 - R0;  R9 := @INBUF(84);  IC(R0,B8);
08 0AFA  09 0841  1339          WHILE R0 = " " DO NEXTCHAR;  R9 := R9 - R8;
08 0B0C  09 0841  1340          IF R0 = ")" THEN
08 0B14  09 0841  1341          BEGIN  R8 := R8 +R5;  R9 := R9 - R5;  ADVANCESYMBOL;
08 0B22  09 0841  1342          END ELSE
08 0B22  09 0841  1343          BEGIN  R4 := 5;  ERROR;  ADVANCESYMBOL;
08 0B38  09 0841  1344          END;
08 0B38  09 0841  1345          R7 := CURBUFBASE;
08 0B3C  09 0841  1346       END;
08 0B3C  09 0841  1347    END;
08 0B3C  09 0841  1348    BEGIN    COMMENT  REFERENCE;                                  16
08 0B40  09 0841  1349       OUTCODE;  R1 := RLISTPOINT+2;  SIMTYPEINFO := R1;  R1 := 9;
08 0B54  09 0841  1350       IF R5 < PARD THEN SETTYPE;
08 0B60  09 0841  1351       TYPE := R1;  R7 := NEXTBUFBASE;  CLI("(",NEXTBUF);
08 0B6C  09 0841  1352       IF  =  THEN
08 0B70  09 0841  1353       BEGIN  ADVANCESYMBOL;  MVI(#01,REFERFLAG);
08 0B7E  09 0841  1354       END ELSE
08 0B7E  09 0841  1355       BEGIN  R4 := 10;  ERROR;
08 0B8A  09 0841  1356       END;
08 0B8A  09 0841  1357       R7 := CURBUFBASE;  MVI(#01,DECLAREFLAG);
08 0B92  09 0841  1358    END;
08 0B92  09 0841  1359    BEGIN    COMMENT  VALUE;                                      17
08 0B96  09 0841  1360       OUTCODE;  R1 := #1000000 OR TYPE;  TYPE := R1;
08 0BA6  09 0841  1361    END;
08 0BA6  09 0841  1362    BEGIN    COMMENT  RESULT;                                     18
08 0BAA  09 0841  1363       OUTCODE;  R1 := #2000000 OR TYPE;  TYPE := R1;
08 0BBA  09 0841  1364     END;
08 0BBA  09 0841  1365     BEGIN      COMMENT  ARRAY;                                   19
08 0BBE  09 0841  1366       OUTCODE;  MVI(#01,ARRAYFLAG);
08 0BC6  09 0841  1367          CLI(#01,PROCFLAG);  IF  =  THEN
08 0BCE  09 0841  1368          BEGIN  R2 := SCRATCHINDEX;  FARRAYPNT := R2;  END;
08 0BD6  09 0841  1369       R1 := #000200 OR TYPE;  TYPE := R1;
08 0BE2  09 0841  1370    END;
08 0BE2  09 0841  1371    BEGIN    COMMENT  RECORD;                                     20
```

Moore Business Forms, Inc. sv

```
08 0BE6  09 0841  1372          OUTCODE;  R1 := RECORDNO + 1;  RECORDNO := R1;
08 0BF6  09 0841  1373          R1 := R1 SHLL 16 ++ #400;  TYPE := R1;  MVI(#01,RECORDFLAG);
08 0C06  09 0841  1374          MVI(#01,DECLARFLAG);
08 0C0A  09 0841  1375       END;
08 0C0A  09 0841  1376       BEGIN    COMMENT PROCEDURE;                                    21
08 0C0E  09 0841  1377          OUTCODE; MVI(#01,DECLARFLAG);
08 0C16  09 0841  1378          CLI(#01,PROCFLAG);
08 0C1A  09 0841  1379          IF  =  THEN
08 0C1E  09 0841  1380          BEGIN  R1 := #1300 OR TYPE;  TYPE := R1;
08 0C2A  09 0841  1381          END ELSE
08 0C2A  09 0841  1382          BEGIN  R1 := #300 OR TYPE;  TYPE := R1;  R7 := NEXTBUFBASE;
08 0C3E  09 0841  1383             IF R5 = NEXTBUFTYPE THEN
08 0C46  09 0841  1384             BEGIN  R7 := CURBUFBASE;  FETCHSYMBOL;  R7 := NEXTBUFBASE;
08 0C52  09 0841  1385                CLI("(",NEXTBUF);  IF ¬= THEN
08 0C5A  09 0841  1386                BEGIN  R7 := CURBUFBASE;  MATCHRESERVED;  IF R5 = R3 THEN
08 0C68  09 0841  1387                   BEGIN  OPENBLOCK; CLOSEBLOCK; IDSEARCH;
08 0C74  09 0841  1388                   END ELSE  GOTO L1;
08 0C74  09 0841  1389                END ELSE
08 0C74  09 0841  1390                BEGIN  R1 := BLOCKNO + 1 SHLL 16 OR TYPE;
08 0C88  09 0841  1391                   TYPE := R1;  R7 := CURBUFBASE;  MATCHRESERVED;
08 0C94  09 0841  1392                   IF R3 = 1 THEN IDSEARCH ELSE GOTO L1;
08 0CA0  09 0841  1393                   R1 := 0;  TYPE := R1;  OPENBLOCK;
08 0CAC  09 0841  1394                   MVI(#01,PROCFLAG);
08 0CB0  09 0841  1395                END;
08 0CBC  09 0841  1396             END;
08 0CB0  09 0841  1397             R7 := CURBUFBASE;
08 0CB4  09 0841  1398          END;
08 0CB4  09 0841  1399       END;
08 0CB4  09 0841  1400       BEGIN    COMMENT  STRING;                                      22
08 0CB8  09 0841  1401          MVI(#01,DECLARFLAG);
08 0CBC  09 0841  1402          IF R5 < PARD THEN SETTYPE;
08 0CC8  09 0841  1403          TYPE := R1;
08 0CCC  09 0841  1404          R1 := SIMPLETYPE;  OUTCODE;
08 0CD4  09 0841  1405          R7 := NEXTBUFBASE;  CLI("(",NEXTBUF);
08 0CDC  09 0841  1406          IF  =  THEN
08 0CE0  09 0841  1407          BEGIN  ADVANCESYMBOL;  R7 := NEXTBUFBASE;
08 0CEE  09 0841  1408             CLI(2,NEXTBUFTYPE(1));  IF  =  THEN
08 0CF6  09 0841  1409             BEGIN  R1 := NEXTBUFI - R5;  IF R1 < 256 AND R1 >= 0 THEN
08 0D0A  09 0841  1410                BEGIN  SIMTYPEINFO := R1;  GOTO PAREN;
08 0D12  09 0841  1411                END;
08 0D12  09 0841  1412             END;
08 0D12  09 0841  1413             R4 := 1;  ERROR;
08 0D1A  09 0841  1414    PAREN:   ADVANCESYMBOL;  R7 := NEXTBUFBASE;  CLI(")",NEXTBUF);
08 0D2C  09 0841  1415             IF  =  THEN ADVANCESYMBOL ELSE
08 0D3A  09 0841  1416             BEGIN  R4 := 5;  ERROR;  END;
08 0D46  09 0841  1417          END ELSE
08 0D46  09 0841  1418          BEGIN R1 := STRNGLENGTH;  SIMTYPEINFO := R1;
08 0D52  09 0841  1419          END;
08 0D52  09 0841  1420          R7 := CURBUFBASE;
08 0D56  09 0841  1421       END;
08 0D56  09 0841  1422       END;
08 0DB2  09 0841  1423       GOTO L;
08 0CB6  09 0841  1424    ENDPROGRAM:
08 0DB6  09 0841  1425       R1 := ENDOFPROGRAM; OUTCODE;  CLEANUP;  R10 := RETADDR;
08 0DCC  09 0841  1426    END;   COMMENT  END OF PASS 1;

                        SEGMENT 08  NAME = SEG#08      LENGTH = 0E58  BASE REG = 15
```

```
06 00A4  07 013C  1427
06 00A4  07 013C  1428  COMMENT  * * * PASS 2 LOCAL VARIABLES BEGIN HERE * * * ;
06 00A4  07 013C  1429
06 00A4  07 013C  1430  INTEGER SAVE14 SYN BASESAVE(16),     COMMENT DATA BASE ADDRESSES;
06 00A4  07 013C  1431          METABASE SYN BASESAVE(0);
06 00A4  07 013C  1432  INTEGER COMMONBASE SYN XFERVECTOR(0);   COMMENT COMMON BASE ADDRESS;
06 00A4  07 013C  1433  INTEGER COMMONLIMIT SYN XFERVECTOR(4);  COMMENT HIGH COMMON ADDRESS;
06 00A4  07 013C  1434  LONG REAL DEC SYN PKDEC;      COMMENT USED FOR DECIMAL OUTPUT;
06 00A4  07 013C  1435  INTEGER HEX SYN PKDEC;        COMMENT USED FOR HEXIDEC OUTPUT;
06 00A4  07 013C  1436  FUNCTION OI(4,#9600);
06 00A4  07 013C  1437
06 00A4  07 013C  1438  ARRAY 2  LONG REAL NUMVALUE;
06 00A4  07 0150  1439    INTEGER NUMVALUE4 SYN NUMVALUE;
06 00A4  07 0150  1440  SHORT INTEGER  BN;  COMMENT  BLOCK NUMBER;
06 00A4  07 0152  1441  SHORT INTEGER  BNC;  COMMENT  CURRENT BLOCK NUMBER;
06 00A4  07 0154  1442  SHORT INTEGER  SN;  COMMENT  SEGMENT NUMBER;
06 00A4  07 0156  1443  SHORT INTEGER  SNC;  COMMENT  CURRENT SEGMENT NUMBER;
06 00A4  07 0158  1444  SHORT INTEGER  HN;  COMMENT  HIERARCHY NUMBER;
06 00A4  07 015A  1445  SHORT INTEGER  DRELAD;  COMMENT  DATA RELATIVE ADDRESS;
06 00A4  07 015C  1446  SHORT INTEGER VARORIGIN;  COMMENT VARIABLE ORIGIN IN DATA SEGMENT;
06 00A4  07 015E  1447  SHORT INTEGER CARDNUMBER;
06 00A4  07 0160  1448  BYTE  SIMTYPE;
06 00A4  07 0161  1449  BYTE STRINGLENGTH;
06 00A4  07 0162  1450  SHORT INTEGER  VALUE;
06 00A4  07 0164  1451  SHORT INTEGER  FLAG ;
06 00A4  07 0166  1452  BYTE COMMONFLAG;  COMMENT SET IF COMMON OVERFLOWS;
06 00A4  07 0167  1453  BYTE UNDECLFLAG;
06 00A4  07 0168  1454  SHORT INTEGER TREELENGTH;
06 00A4  07 016A  1455  SHORT INTEGER RULENUMBER;      COMMENT  CONTAINS PRODUCTION NUMBER;
06 00A4  07 016C  1456  SHORT INTEGER DRELPOINT;
06 00A4  07 016E  1457  ARRAY  10  SHORT INTEGER DRELSAVE;
06 00A4  07 0182  1458  ARRAY 512  SHORT INTEGER BLOCKLIST2;    COMMENT  STATIC LINKS;
06 00A4  07 0582  1459  ARRAY 256  INTEGER SNLIST;
06 00A4  07 0984  1460    SHORT INTEGER STATLINK SYN SNLIST;
06 00A4  07 0984  1461    SHORT INTEGER OUTPOINT SYN SNLIST(2);
06 00A4  07 0984  1462  ARRAY   256  INTEGER CONSPOINTERSTACK ;
06 00A4  07 0D84  1463    SHORT INTEGER CTPOINTER SYN CONSPOINTERSTACK;
06 00A4  07 0D84  1464    SHORT INTEGER LITPOINTER SYN CONSPOINTERSTACK(2);
06 00A4  07 0D84  1465  ARRAY  2000  BYTE LITERALTABLE SYN B3;
06 00A4  07 0D84  1466  INTEGER  LITBASE;
06 00A4  07 0D88  1467  SHORT INTEGER CTPNT;     COMMENT  POINTER TO CONSTANTTABLE;
06 00A4  07 0D8A  1468  SHORT INTEGER LITPNT;    COMMENT  POINTER TO LITERALTABLE;
06 00A4  07 0D8C  1469  SHORT INTEGER CTORG;
06 00A4  07 0D8E  1470  SHORT INTEGER LITORG;
06 00A4  07 0D90  1471  INTEGER LITINITIAL;
06 00A4  07 0D94  1472  INTEGER  R1SAVE,R2SAVE,R3SAVE,R4SAVE,R5SAVE,R6SAVE;
06 00A4  07 0DAC  1473  ARRAY  16  BYTE TRANSTABLE; COMMENT TRANSLATION OF HEX DIGITS;
06 00A4  07 0DBC  1474  SHORT INTEGER OUTBASE;
06 00A4  07 0DBE  1475  INTEGER RETCODE;  COMMENT RETURN CODE FOR PHASE;
06 00A4  07 0DC4  1476  ARRAY  132  BYTE BUFFER;
06 00A4  07 0E48  1477
06 00A4  07 0E48  1478  DUMMY BASE R12;  COMMENT ALLOCATED FROM WORK SPACE;
06 00A4  18 0000  1479  INTEGER LABELADDR;    COMMENT  MUST DIRECTLY PRECEDE CONSTANTTABLE;
06 00A4  18 0004  1480  ARRAY 256  INTEGER CONSTANTTABLE;
06 00A4  18 0404  1481    SHORT INTEGER CINFO SYN CONSTANTTABLE;
06 00A4  18 0404  1482    BYTE CLENGTH        SYN CONSTANTTABLE;
06 00A4  18 0404  1483    BYTE CTYPE          SYN CONSTANTTABLE(1);
06 00A4  18 0404  1484    SHORT INTEGER CADDR SYN CONSTANTTABLE(2);
```

```
06 00A4   18 0404        1485   ARRAY 96  BYTE S;   COMMENT PARSING STACK;
06 00A4   18 0464        1486   ARRAY 96  BYTE S2;       COMMENT  STACK OF RELATIONS;
06 00A4   18 04C4        1487   ARRAY 96  LONG REAL V;   COMMENT  INTERPRETATION STACK;
06 00A4   18 07C8        1488     INTEGER V12 SYN V;
06 00A4   18 07C8        1489     SHORT INTEGER V1 SYN V;
06 00A4   18 07C8        1490     SHORT INTEGER V2  SYN V(2);           COMMENT BOTH TYPE FIELDS;
06 00A4   18 07C8        1491     BYTE  V21 SYN V(2);     COMMENT  TYPE FIELD;
06 00A4   18 07C8        1492     BYTE  V22 SYN V(3);     COMMENT  SIMPLE TYPE FIELD;
06 00A4   18 07C8        1493     SHORT INTEGER  V34 SYN V(  4);   COMMENT  BOTH REGISTER FIELDS;
06 00A4   18 07C8        1494     BYTE V3  SYN V(4);    COMMENT  GENERAL REGISTER FIELD;
06 00A4   18 07C8        1495     BYTE V4  SYN V(5);    COMMENT  FLOATING REGISTER FIELD;
06 00A4   18 07C8        1496     SHORT INTEGER V5  SYN  V(6);    COMMENT  OUTPUT POINTER FIELD;
06 00A4   18 07C8        1497   INTEGER PLIMIT;    COMMENT MAXIMUM P INDEX;
06 00A4   18 07CC        1498   ARRAY 0 LONG REAL PALIGN;  COMMENT ALIGN P;
06 00A4   18 07D0        1499   ARRAY 1 LOGICAL P;   COMMENT OUTPUT WORK AREA, SIZE IS DYNAMIC;
06 00A4   18 07D4        1500     BYTE OP SYN P;
06 00A4   18 07D4        1501     BYTE CONV  SYN P(1);
06 00A4   18 07D4        1502     SHORT INTEGER  POINTER  SYN P(2);
06 00A4   18 07D4        1503   CLOSE BASE;
06 00A4   07 0E48        1504
06 00A4   07 0E48        1505   SEGMENT BASE R10;   COMMENT TABLE OF SYNTACTIC SYMBOLS;
06 00A8   19 0000        1506   ARRAY   1904   BYTE METATABLE
06 00A8   19 0000        1507   =("             ",
06 00A8   19 0770        1508   "<T VAR ID>  <LABEL ID>  <T ARRAY ID><T FUNC ID> <RC CL ID>  ",
06 00A8   19 0770        1509   "<T FLD ID>  <CON ID>    <ST FUNC ID><FILE ID>   <ST PROC ID>",
06 00A8   19 0770        1510   "<SI VAR DC> <SI VAR DC*><SI TYPE >  <REF TYPE>  <FILE DC>   "
06 00A8   19 0770        1511   ,"<FILE HD**> <FILE HD*-> <FILE HD*>  <FILE DESC> <ARRAY DC>  "
06 00A8   19 0770        1512   ,"<ARRAY HD>  <BND LST HD><PROC DECL> <T PR BODY> <T PR HEAD> "
06 00A8   19 0770        1513   ,
06 00A8   19 0770        1514   "<T PR HEAD*><PROCEDURE> <FPAR HEAD> <FPAR HEAD*><FPAR HEAD->",
06 00A8   19 0770        1515   "<FBND LIST> <FBND HEAD> <F ARRAY HD><RC CL DC>  <RC HEAD>   ",
06 00A8   19 0770        1516   "<RC HEAD*>  <RECORD>    <T VAR>     <STR SEL HD><LENGTH>    ",
06 00A8   19 0770        1517   "<SI T VAR>  <T FLD HD>  <T ARRAY HD><T FUNC DES><APAR HEAD> ",
06 00A8   19 0770        1518   "<T EXP>     <T EXP*>    <IF CL>     <TRUE EXP>  <CASE HEAD> ",
06 00A8   19 0770        1519   "<CASE CL>   <SI T EXP>  <SI T EXP*> <T TERM>    <T TERM*>   ",
06 00A8   19 0770        1520   "<T FACT>    <T SECON>   <T PRIM>    <REL OP>    <EQL OP>    ",
06 00A8   19 0770        1521   "<LEFT PAR>  <RC DES HD> <PROGRAM>   <STATEMENT> <STATEMENT*>",
06 00A8   19 0770        1522   "<SI ST>     <BLOCK>     <BLOCKBODY> <BLOCKHEAD> <LABEL DEF> ",
06 00A8   19 0770        1523   "<T ASS ST>  <TRUE PART> <CASE SEQ>  <FOR CL>    <FOR HEAD>  ",
06 00A8   19 0770        1524   "<FOR LIST>  <FOR>       <STEPUNTIL> <WHILE CL>  <ST PROC HD>",
06 00A8   19 0770        1525   "<UNIT DES>  <UNIT HD>   <SI T EXP**>                        ",
06 00A8   19 0770        1526   "                                                           ",
06 00A8   19 0770        1527   "                                                           ",
06 00A8   19 0770        1528   "                                                           ",
06 00A8   19 0770        1529   "<ID>        ,           )           REFERENCE   ,           ",
06 00A8   19 0770        1530   "(           <I/O DEV>   FILE        ::          ARRAY       ",
06 00A8   19 0770        1531   "END         ;           PROCEDURE   VALUE       RESULT      ",
06 00A8   19 0770        1532   "*           RECORD      |           <T NUMBER>  IF          ",
06 00A8   19 0770        1533   "THEN        ELSE        CASE        OF          IS          ",
06 00A8   19 0770        1534   "+           -           OR          <STRING>    NULL        ",
06 00A8   19 0770        1535   "/           DIV         REM         AND         ¬           ",
06 00A8   19 0770        1536   "**  OR  SHL SHR         TRUE        FALSE       LONG        ",
06 00A8   19 0770        1537   "ABS         <BIT SEQ>   <           =           >           ",
06 00A8   19 0770        1538   ".           DO          GOTO        ON          OFF         ",
06 00A8   19 0770        1539   "BEGIN       PARALLEL    :           :=          FOR         ",
06 00A8   19 0770        1540   "STEP        UNTIL       WHILE       SHORT       ");
06 00A8   19 0770        1541   CLOSE BASE;
```

SEGMENT 19 NAME = SEG#19 LENGTH = 0770 BASE REG = 10

```
C6 00A8  07 0E48   1542
06 00A8  07 0E48   1543   SEGMENT BASE R14;  COMMENT READ-ONLY CONSTANTS;
06 00AC  20 0000   1544   ARRAY  900  BYTE OPTABL
06 00AC  20 0000   1545   =("        +        -        *        /        **       L:=      A:=      "
06 00AC  20 0384   1546   ,"S:=      R:=      F:=               STEPUNTIDIV      REM      <        "
06 00AC  20 0384   1547   ,"<=       >        >=       =        ¬=                L:=2     A:=2     "
06 00AC  20 0384   1548   ,"S:=2     R:=2                                 AP)      INDX     REFX     "
C6 00AC  20 0384   1549   ,"IFEXP                     UAR      DAR      BB       END      PCL      "
06 00AC  20 0384   1550   ,"SUBSTRIN|        AP,      R,       AR,      AR)      R)       LOGOR    "
06 00AC  20 0384   1551   ,"BITOR    LOGAND  BITAND   FORST    FORST2   FORLIST  FORCL    ENDFORLI"
06 00AC  20 0384   1552   ,"UJIFEXP UJ       CL       IFST     ::       IS       FILEINDX,        "
06 00AC  20 0384   1553   ,"WHILEOP WHILEST  IFJ      U-       ABS                        LOG¬     "
06 00AC  20 0384   1554   ,"BIT¬     ON      OFF      GOTO     :        STACKADD          CARD     "
06 00AC  20 0384   1555   ,"CASE                      BEGIN    PARBEGININUMBER  NUMBER   ID       "
06 00AC  20 0384   1556   ,"LABELID ARRAYID  FUNCID   RCCLID   FIELDID CONID    FILEID   PROCDC   "
06 00AC  20 0384   1557   ,"RCCLDC   SEG     BIT      STRING   TRUE     FALSE    WHILE    NULL     "
06 00AC  20 0384   1558   ,"NULLST   ARRAYDC AR*      STFUNCIDSTPROCID                     IF       "
06 00AC  20 0384   1559   );
06 00AC  20 0384   1560   SHORT INTEGER SITYPECODE  =#0D;
C6 00AC  20 0386   1561   SHORT INTEGER BNDLSTHD    =#16;
06 00AC  20 0388   1562   SHORT INTEGER TPROCBDY     =#18;
06 00AC  20 038A   1563   SHORT INTEGER TPROCHD     =#19;
06 00AC  20 038C   1564   SHORT INTEGER PROCNT      =#1B;
06 00AC  20 038E   1565   SHORT INTEGER TFUNCDES    =#2C;
06 00AC  20 0390   1566   SHORT INTEGER APARHEAD    =#2D;
06 00AC  20 0392   1567   SHORT INTEGER CASEHEADCODE=#32;
06 00AC  20 0394   1568   SHORT INTEGER BLOCKCODE   =#43;
06 00AC  20 0396   1569   SHORT INTEGER BBCODE      =#44;
06 00AC  20 0398   1570   SHORT INTEGER BHCODE      =#45;
C6 00AC  20 039A   1571   SHORT INTEGER CASESEQCODE =#49;
06 00AC  20 039C   1572   SHORT INTEGER FORCLCODE   =#4A;
06 00AC  20 039E   1573   SHORT INTEGER FORHEAD     = #4B;
06 00AC  20 03A0   1574   SHORT INTEGER FORLISTCODE = #4C;
06 00AC  20 03A2   1575   SHORT INTEGER FORNT      =#4D;
06 00AC  20 03A4   1576   SHORT INTEGER WHILECLCODE =#4F;
C6 00AC  20 03A6   1577   SHORT INTEGER IDCODE      =#65;
06 00AC  20 03A8   1578   SHORT INTEGER LPARENCODE  =#6A;
06 00AC  20 03AA   1579   SHORT INTEGER ENDCODE     =#6F;
06 00AC  20 03AC   1580   SHORT INTEGER SCOLCODE   =#70;
06 00AC  20 03AE   1581   SHORT INTEGER NUMBERCODE  =#77;
06 00AC  20 03B0   1582   SHORT INTEGER STRINGCODE  =#81;
06 00AC  20 03B2   1583   SHORT INTEGER BITCODE     =#8E;
06 00AC  20 03B4   1584   SHORT INTEGER ENDFILE     =#92;
06 00AC  20 03B6   1585   SHORT INTEGER GOTOCODE    =#94;
06 00AC  20 03B8   1586   SHORT INTEGER BEGINCODE   =#97;
06 00AC  20 03BA   1587   SHORT INTEGER FORCODE     =#9B;
06 00AC  20 03BC   1588   SHORT INTEGER FIRSTTERMINAL SYN IDCODE;
06 00AC  20 03BC   1589   SHORT INTEGER LSS=1;
06 00AC  20 03BE   1590   SHORT INTEGER GTR =2;
06 00AC  20 03C0   1591   SHORT INTEGER EQL =3;
06 00AC  20 03C2   1592   SHORT INTEGER PROCEDURETYPE =3;
C6 00AC  20 03C4   1593   ARRAY 3 SHORT INTEGER          COMMENT INSTRUCTIONS TO BE EXECUTED;
06 00AC  20 03C4   1594       COMPARE=(#D500S,@B3(1),@B4),      MOVE=(#D200S,@BUFFER(75),@B1),
06 00AC  20 03D0   1595       ERRMOVE=(#D200S,@BUFFER(27),@B1),
06 00AC  20 03D6   1596       ERRMOVE91=(#D200S,@BUFFER(91),@B1),
06 00AC  20 03DC   1597       MOVE31=(#D200S,@B3,@B1),
```

```
06 00AC   20 03E2   1598      MOVE14=(#D200S,@B1,@B4),          COMPARE24=(#D500S,@B2,@B4),
06 00AC   20 03EE   1599      TREEMOVE=(#D200S,@B2(4),@B4),     TREEXOR=(#D700S,@B4,@B4),
06 00AC   20 03FA   1600      MOVE24=(#D200S,@B2,@B4),          MOVENUMBER=(#D200S,@NUMVALUE,@B5);
06 00AC   20 0406   1601    SHORT INTEGER NOMORERULES =#FF;
06 00AC   20 0408   1602    ARRAY 10  BYTE LENGTHTABLE =(0,4,4,8,8,16,1,0,4,4);
06 00AC   20 0412   1603    ARRAY 17 SHORT INTEGER BITTABLE =
06 00AC   20 0434   1604     (#0,#1,#2,#4,#8,#10,#20,#40,#80,#100,#200,#400,#800,
06 00AC   20 0434   1605      #1000,#2000,#4000,#8000);
06 00AC   20 0434   1606    ARRAY 10 SHORT INTEGER INCREASE =
06 00AC   20 0448   1607     (0, 1, #100, #100, #200, #200, 1, 2, 1, 1);
06 00AC   20 0448   1608    LONG REAL MASK=#4020202020204040;
06 00AC   20 0450   1609    LONG REAL MASK2=#4020202021214040;
06 00AC   20 0458   1610    ARRAY  78  BYTE HEADER2
06 00AC   20 0458   1611       =("    LOC    IDLOC1     IDLOC2   "
06 00AC   20 04A6   1612       ,"SIMTYPEINFO        TYPEINFO   TYPE  SIMTYPE   ID");
06 00AC   20 04A6   1613    ARRAY  64  BYTE HEADER5 =(
06 00AC   20 04E6   1614      "           (HEX)    HN  SEG                  VR   RCCLNO   (HEX)");
06 00AC   20 04E6   1615    ARRAY  75  SHORT INTEGER ERTB
06 00AC   20 04E6   1616    =(#100,#265,0,#365,0,#345,0
06 00AC   20 057C   1617     ,#20D,#271,#26B,#100,#269,#267,0
06 00AC   20 057C   1618     ,#399,0,#267,#269,#265,0,#385,0
06 00AC   20 057C   1619     ,#265,#365,0,#365,#100,#270,#269,0
06 00AC   20 057C   1620     ,#374,#265,0,#365,#265,0,#245,0
06 00AC   20 057C   1621     ,#270,0,#265,#20D,0,#36D,#270,#290,0
06 00AC   20 057C   1622     ,#270,#267,0,0,#265,#26A,0
06 00AC   20 057C   1623     ,#365,#400,0,0,0,0
06 00AC   20 057C   1624     ,#293,0,#332,0,#349,0,#326,0
06 00AC   20 057C   1625     ,#343,#33A,#270,0,#265,#370,0);
06 00AC   20 057C   1626      BYTE CODE SYN ERTB;
06 00AC   20 057C   1627      BYTE NODE SYN ERTB(1);
06 00AC   20 057C   1628    ARRAY 60  SHORT INTEGER EMTB
06 00AC   20 057C   1629      =(#5E COMMENT  NONTERMINAL RULES START AT #5E;
06 00AC   20 05F4   1630      ,0,#36,#5E,6,#5E,#E,#16,#6,#1C,2
06 00AC   20 05F4   1631      ,#5E,#5E,2,#6C,#6C,#20,2,#28,#5E,#2C
06 00AC   20 05F4   1632      ,#5E,#5E,#2C,#32,#2C,#66,2,2,#36,#5E
06 00AC   20 05F4   1633      ,2,2,2,#66,2,#3C,#66,#5E,#5E,#42
06 00AC   20 05F4   1634      ,#42,#5E,2,2,2,#48,#4C,6,6,6
06 00AC   20 05F4   1635      ,#50,#50,#56,2,2,#42,2,2);
06 00AC   20 05F4   1636    ARRAY 160 SHORT INTEGER MTB =
06 00AC   20 0734   1637           (0,1,6,7,17,27,37,43,48,58,63,69,70,81,99,111,112,119,132,143,
06 00AC   20 0734   1638           156,157,169,186,187,188,203,209,220,221,237,286,296,309,321,322,
06 00AC   20 0734   1639           339,346,354,371,379,380,390,397,422,431,466,476,481,508,509,522,
06 00AC   20 0734   1640           533,538,579,584,619,624,641,646,647,648,655,668,669,670,675,685,
06 00AC   20 0734   1641           690,724,759,760,765,766,789,801,817,829,836,837,849,862,867,874,
06 00AC   20 0734   1642           879,880,881,882,883,884,885,886,887,888,889,890,891,892,893,894,
06 00AC   20 0734   1643           895,896,942,943,944,950,951,1004,1005,1011,1012,1013,1014,1015,
06 00AC   20 0734   1644           1021,1022,1023,1024,1030,1036,1041,1048,1049,1054,1061,1062,1063,
06 00AC   20 0734   1645           1069,1075,1076,1081,1086,1087,1088,1089,1090,1101,1102,1103,1108,
06 00AC   20 0734   1646           1113,1119,1125,1130,1140,1145,1155,1162,1163,1169,1170,1171,1181,
06 00AC   20 0734   1647           1182,1183,1184,1190,1197,1202,1208);
06 00AC   20 0734   1648
06 00AC   20 0734   1649    BYTE  PLUS          SYN    1  ;
06 00AC   20 0734   1650    BYTE  MINUS         SYN    2  ;
06 00AC   20 0734   1651    BYTE  TIMES         SYN    3  ;
06 00AC   20 0734   1652    BYTE  DIVIDE        SYN    4  ;
06 00AC   20 0734   1653    BYTE  EXPON         SYN    5;
06 00AC   20 0734   1654    BYTE  LCOLONEQ      SYN    6;
06 00AC   20 0734   1655     SHORT INTEGER LASSIGN = 6;
```

```
06 0CAC   20 0736        1656   BYTE  ACOLONEQ     SYN    7;
06 00AC   20 0736        1657    SHORT INTEGER AASSIGN = 7;
06 0CAC   20 0738        1658   BYTE  SCOLONEQ     SYN    8;
06 00AC   20 0738        1659    SHORT INTEGER SASSIGN = 8;
06 00AC   20 073A        1660   BYTE  RCOLONEQ     SYN    9;
06 00AC   20 073A        1661    SHORT INTEGER RASSIGN = 9;
06 00AC   20 073C        1662   BYTE  FCOLONEQ     SYN   10;
06 00AC   20 073C        1663   BYTE  STEPUNTIL    SYN   12;
06 00AC   20 073C        1664   BYTE  DIV          SYN   13;
06 00AC   20 073C        1665   BYTE  REM          SYN   14;
06 00AC   20 073C        1666   BYTE  LESS         SYN   15;
06 00AC   20 073C        1667   BYTE  LESSEQ       SYN   16;
06 00AC   20 073C        1668   BYTE  GREATER      SYN   17;
06 0CAC   20 073C        1669   BYTE  GTEQ         SYN   18;
06 00AC   20 073C        1670   BYTE  EQUAL        SYN   19;
06 0CAC   20 073C        1671   BYTE  UNEQ         SYN   20;
06 00AC   20 073C        1672   BYTE  LCOLONEQ2    SYN   22;
06 00AC   20 073C        1673   BYTE  ACOLONEQ2    SYN   23;
06 00AC   20 073C        1674   BYTE  SCOLONEQ2    SYN   24;
06 0CAC   20 073C        1675   BYTE  RCOLONEQ2    SYN   25;
06 00AC   20 073C        1676   BYTE  APPAREN      SYN   29;
06 00AC   20 073C        1677   BYTE  INDX         SYN   30;
06 0CAC   20 073C        1678   BYTE  REFX         SYN   31;
06 00AC   20 073C        1679   BYTE  IFEXP        SYN   32;
06 00AC   20 073C        1680   BYTE  SHL          SYN   35;
06 00AC   20 073C        1681   BYTE  SHR          SYN   36;
06 0CAC   20 073C        1682   BYTE  BBB          SYN   37;
06 00AC   20 073C        1683   BYTE  ENDD         SYN   38;
06 00AC   20 073C        1684   BYTE  PCL          SYN   39;
06 00AC   20 073C        1685   BYTE  SUBSTRING    SYN   40;
06 00AC   20 073C        1686   BYTE  BAR          SYN   41;
06 00AC   20 073C        1687   BYTE  APCOMMA      SYN   42;
06 00AC   20 073C        1688   BYTE  RCOMMA       SYN   43;
06 00AC   20 073C        1689   BYTE  ARCOMMA      SYN   44;
06 00AC   20 073C        1690   BYTE  ARPAREN      SYN   45;
06 00AC   20 073C        1691   BYTE  RPAREN       SYN   46;
06 00AC   20 073C        1692   BYTE  LOGOR        SYN   47;
06 00AC   20 073C        1693   BYTE  BITOR        SYN   48;
06 00AC   20 073C        1694   BYTE  LOGAND       SYN   49;
06 00AC   20 073C        1695   BYTE  BITAND       SYN   50;
06 00AC   20 073C        1696   BYTE  ITERST       SYN   51;
06 00AC   20 073C        1697   BYTE  ITERST2      SYN   52;
06 00AC   20 073C        1698   BYTE  FORLIST      SYN   53;
06 00AC   20 073C        1699   BYTE  FORCL        SYN   54;
06 0CAC   20 073C        1700   BYTE  ENDFORLIST   SYN   55;
06 00AC   20 073C        1701   BYTE  UJIFEXP      SYN   56;
06 0CAC   20 073C        1702   BYTE  UJ           SYN   57;
06 0CAC   20 073C        1703   BYTE  CLL          SYN   58;
06 0CAC   20 073C        1704   BYTE  IFST         SYN   59;
06 00AC   20 073C        1705   BYTE  COLON        SYN   60;
06 0CAC   20 073C        1706   BYTE  IS           SYN   61;
06 00AC   20 073C        1707   BYTE  FILEINDX     SYN   62;
06 00AC   20 073C        1708   BYTE  COMMA        SYN   63;
06 00AC   20 073C        1709   BYTE  WHILEOP      SYN   64;
06 00AC   20 073C        1710   BYTE  WHILEST      SYN   65;
06 00AC   20 073C        1711   BYTE  IFJ          SYN   66;
06 00AC   20 073C        1712                                COMMENT  UNARY OPERATORS;
06 00AC   20 073C        1713   BYTE  UMINUS       SYN   67;
```

```
06 00AC  20 073C  1714  BYTE  ABSS         SYN   68;
06 00AC  20 073C  1715  BYTE  LOGNOT       SYN   71;
06 00AC  20 073C  1716  BYTE  BITNOT       SYN   72;
06 00AC  20 073C  1717  BYTE  ON           SYN   73;
06 00AC  20 073C  1718  BYTE  OFF          SYN   74;
06 00AC  20 073C  1719  BYTE  GOTOO        SYN   75;
06 00AC  20 073C  1720  BYTE  LABELCOLON   SYN   76;
06 00AC  20 073C  1721  BYTE  STACKADDR    SYN   77;
06 00AC  20 073C  1722  BYTE  CARD         SYN   79;
06 00AC  20 073C  1723  BYTE  CASEE        SYN   80;
06 00AC  20 073C  1724                            COMMENT  TERMINAL NODES;
06 00AC  20 073C  1725  BYTE  BEGINN       SYN   83;
06 00AC  20 073C  1726  BYTE  PARBEGIN     SYN   84;
06 00AC  20 073C  1727  BYTE  INUMBER      SYN   85;
06 00AC  20 073C  1728  BYTE  NUMBER       SYN   86;
06 00AC  20 073C  1729  BYTE  ID           SYN   87;
06 00AC  20 073C  1730  BYTE  LABELID      SYN   88;
06 00AC  20 073C  1731  BYTE  ARRAYID      SYN   89;
06 00AC  20 073C  1732  BYTE  FUNCID       SYN   90;
06 00AC  20 073C  1733  BYTE  RCCLID       SYN   91;
06 00AC  20 073C  1734  BYTE  FIELDID      SYN   92;
06 00AC  20 073C  1735  BYTE  CONID        SYN   93;
06 00AC  20 073C  1736  BYTE  FILEID       SYN   94;
06 00AC  20 073C  1737  BYTE  PROCDC       SYN   95;
06 00AC  20 073C  1738  BYTE  RCCLDC       SYN   96;
06 00AC  20 073C  1739  BYTE  SEG          SYN   97;
06 00AC  20 073C  1740  BYTE  BITT         SYN   98;
06 00AC  20 073C  1741  BYTE  STRINGG      SYN   99;
06 00AC  20 073C  1742  BYTE  TRUE         SYN  100;
06 00AC  20 073C  1743  BYTE  FALSE        SYN  101;
06 00AC  20 073C  1744  BYTE  WHILEE       SYN  102;
06 00AC  20 073C  1745  BYTE  NUL          SYN  103;
06 00AC  20 073C  1746  BYTE  NULLST       SYN   104;
06 00AC  20 073C  1747  BYTE  ARRAYDC      SYN  105;
06 00AC  20 073C  1748  BYTE  ARSTAR       SYN  106;
06 00AC  20 073C  1749  BYTE  STFUNCID     SYN  107;
06 00AC  20 073C  1750  BYTE  STPROCID     SYN  108;
06 00AC  20 073C  1751  BYTE  IFF          SYN  111;
06 00AC  20 073C  1752
06 00AC  20 073C  1753  SHORT INTEGER  MULTDEF    SYN  1;
06 00AC  20 073C  1754  SHORT INTEGER  UNDECL     SYN  2;
06 00AC  20 073C  1755  SHORT INTEGER  SYNTAXERROR SYN 3;
06 00AC  20 073C  1756  SHORT INTEGER  RCCLERROR  SYN  4;
06 00AC  20 073C  1757  SHORT INTEGER  TYPE3ERROR   SYN 5;
06 00AC  20 073C  1758  SHORT INTEGER  APARERROR SYN  6;
06 00AC  20 073C  1759  SHORT INTEGER  ARRAYERROR SYN 7;
06 00AC  20 073C  1760  SHORT INTEGER DATAERROR  SYN  8;
06 00AC  20 073C  1761  SHORT INTEGER FIELDERROR   SYN  9;
06 00AC  20 073C  1762  SHORT INTEGER  LENGTHERROR SYN 10;
06 00AC  20 073C  1763  SHORT INTEGER  REFERROR  SYN  11;
06 00AC  20 073C  1764  SHORT INTEGER  HIERARCHYERROR SYN 12;
06 00AC  20 073C  1765  SHORT INTEGER  FILEERROR  SYN  13;
06 00AC  20 073C  1766  SHORT INTEGER REFRECERROR  SYN  14;
06 00AC  20 073C  1767  SHORT INTEGER  PROCERROR  SYN 15;
06 00AC  20 073C  1768  SHORT INTEGER  TYPEERROR  SYN 16;
06 00AC  20 073C  1769  SHORT INTEGER  RESULTERROR  SYN 17;
06 00AC  20 073C  1770  SHORT INTEGER  PPROCERROR SYN 18;
06 00AC  20 073C  1771  SHORT INTEGER  RELATIONERROR SYN 19;
```

```
06 00AC   20 073C   1772   SHORT INTEGER  ARRAY2ERROR  SYN  20;
06 00AC   20 073C   1773   SHORT INTEGER  CONSTANTERROR SYN 21;
06 00AC   20 073C   1774   SHORT INTEGER STRINGERROR    SYN 22;
06 00AC   20 073C   1775   SHORT INTEGER TABLEERROR SYN 23;
06 00AC   20 073C   1776   ARRAY 160 BYTE RMAP =
06 00AC   20 07DC   1777          (0,1,2,3,4,5,6,7,4,3,6,8,9,10,11,8,12,12,13,14,8,14,12,8,15,16,8,
06 00AC   20 07DC   1778          17,18,19,20,21,22,14,8,23,24,6,25,12,18,3,12,26,7,27,28,29,30,31,
06 00AC   20 07DC   1779          12,32,29,33,34,35,36,37,7,38,38,12,12,0,39,40,41,42,43,44,45,2,
06 00AC   20 07DC   1780          46,47,48,49,12,50,12,48,12,51,12,29,17(0),52,53,54,53,55,56,57,
06 00AC   20 07DC   1781          53,12,53,58,59,53,60,53,61,53,62,7,12,63,64,12,65,66,3(67),34,34,
06 00AC   20 07DC   1782          4(68),69,70,70,7,7,70,70,7,71,72,71,73,46,74,0,0,75,76,45,77,53,
06 00AC   20 07DC   1783          12,78,12,70);
06 00AC   20 07DC   1784   ARRAY 160 BYTE CMAP =
06 00AC   20 087C   1785          (0,1,2,1,1,3,1,4,4,1,5,6,7,8,9,6,4(7),6,7,7,6,10,3(7),11,5(12),6,
06 00AC   20 087C   1786          3(7),1,1,13,5(1),14,15,16,17,18,16,18,19,20,21,22,23,24,25,25,4,
06 00AC   20 087C   1787          19,0,26,27,5,28,29,29,30,31,17,5(5),32,5,5,1,1,33,17(0),34,35,36,
06 00AC   20 087C   1788          9,37,38,39,7,13,40,41,42,43,40,44,45,7,46,47,16,13,48,16,13,25,
06 00AC   20 087C   1789          49,49,25,19,19,4(50),51,52,52,5(4),53,54,53,55,56,5,0,0,57,58,59,
06 00AC   20 087C   1790          60,5,61,62,5,4);
06 00AC   20 087C   1791   ARRAY 1214 BYTE PRTB =
06 00AC   20 0D3A   1792          (255,0,1,41,75,2(255),0,3,38,70,1,3,106,43,80,255,0,4,44,83,1,4,
06 00AC   20 0D3A   1793          106,45,87,255,0,5,53,111,1,5,106,62,147,255,1,6,106,42,79,255,0,
06 00AC   20 0D3A   1794          7,58,133,255,0,8,58,129,1,8,106,61,146,255,0,9,81,201,255,1,10,
06 00AC   20 0D3A   1795          106,80,198,2(255),0,12,11,11,2,12,102,101,12,13,255,1,13,101,12,
06 00AC   20 0D3A   1796          12,2,13,110,101,21,26,2,13,113,101,27,39,255,1,14,103,13,14,2,14,
06 00AC   20 0D3A   1797          102,101,14,16,2(255),2,16,46,103,15,18,255,2,17,46,103,15,19,2,
06 00AC   20 0D3A   1798          17,46,105,16,20,255,1,18,103,15,17,1,18,105,17,21,255,2,19,106,
06 00AC   20 0D3A   1799          107,18,22,2,19,102,101,19,24,2(255),2,21,102,101,21,27,1,21,106,
06 00AC   20 0D3A   1800          22,28,255,4,22,46,109,46,103,20,25,4,22,46,109,46,105,22,29,
06 00AC   20 0D3A   1801          3(255),1,25,65,23,30,0,25,23,31,1,25,24,23,32,255,1,26,112,25,35,
06 00AC   20 0D3A   1802          255,0,27,26,36,2,27,28,103,26,37,2(255),0,29,28,40,2,29,102,101,
06 00AC   20 0D3A   1803          29,54,1,29,112,30,55,255,2,30,13,101,29,48,3,30,13,114,101,29,49,
06 00AC   20 0D3A   1804          3,30,13,115,101,29,50,3,30,13,113,101,29,51,4,30,13,114,115,101,
06 00AC   20 0D3A   1805          29,52,2,30,113,101,29,53,3,30,13,110,101,33,61,255,0,31,28,41,1,
06 00AC   20 0D3A   1806          31,112,30,56,255,2,32,116,103,31,57,2,32,116,105,32,59,255,1,33,
06 00AC   20 0D3A   1807          106,32,58,2,33,102,101,33,62,2(255),1,35,103,34,63,2,35,102,101,
06 00AC   20 0D3A   1808          35,65,1,35,112,36,67,255,2,36,13,101,35,66,255,3,37,106,13,101,
06 00AC   20 0D3A   1809          35,64,255,0,38,58,127,2,38,154,47,71,181,2,38,154,71,71,182,255,
06 00AC   20 0D3A   1810          3,39,46,40,103,38,71,2(255),0,41,38,69,1,41,106,39,73,255,2,42,
06 00AC   20 0D3A   1811          46,103,41,76,255,2,43,46,103,41,77,2,43,116,103,41,78,2,43,46,
06 00AC   20 0D3A   1812          105,43,81,2,43,116,105,43,82,255,0,44,58,128,0,44,66,164,255,2,
06 00AC   20 0D3A   1813          45,46,103,44,84,2,45,64,103,44,85,1,45,103,44,86,2,45,46,105,45,
06 00AC   20 0D3A   1814          88,2,45,64,105,45,89,1,45,105,45,90,255,0,46,24,33,1,46,122,49,
06 00AC   20 0D3A   1815          96,255,0,47,46,91,255,2,48,49,47,47,93,0,48,65,156,1,48,65,65,
06 00AC   20 0D3A   1816          157,1,48,72,65,158,2,48,72,65,65,159,2(255),2,50,46,103,47,94,2,
06 00AC   20 0D3A   1817          50,46,105,50,98,255,1,51,106,50,97,1,51,151,73,185,255,0,52,47,
06 00AC   20 0D3A   1818          92,255,2,53,60,83,52,101,2,53,59,83,52,102,2,53,125,5,52,103,0,
06 00AC   20 0D3A   1819          53,83,104,2,53,126,54,53,108,2,53,127,54,53,109,2,53,128,54,53,
06 00AC   20 0D3A   1820          110,255,0,54,53,105,255,0,55,54,115,2,55,116,56,55,117,2,55,131,
06 00AC   20 0D3A   1821          56,55,118,2,55,132,56,55,119,2,55,133,56,55,120,2,55,134,56,55,
06 00AC   20 0D3A   1822          121,255,0,56,55,116,255,0,57,56,122,2,57,136,58,57,125,2,57,137,
06 00AC   20 0D3A   1823          58,57,126,255,0,58,57,124,3(255),2,61,46,103,58,130,255,2,62,46,
06 00AC   20 0D3A   1824          103,53,112,2,62,46,105,62,148,3(255),0,65,64,150,255,0,66,65,151,
06 00AC   20 0D3A   1825          1,66,122,72,183,255,0,67,66,162,255,2,68,46,111,24,34,1,68,111,
06 00AC   20 0D3A   1826          67,167,2,68,64,111,67,168,1,68,112,68,170,2,68,64,112,68,171,1,
06 00AC   20 0D3A   1827          68,70,68,172,255,0,69,68,169,2,69,11,112,69,175,2,69,20,112,69,
06 00AC   20 0D3A   1828          176,2,69,23,112,69,177,2,69,34,112,69,178,2,69,15,112,69,179,
06 00AC   20 0D3A   1829          2(255),0,71,66,163,2(255),1,73,111,65,160,2,73,64,111,65,161,2,
```

```
06 00AC  20 0D3A  1830          73,64,112,73,186,1,73,112,73,187,255,1,74,147,65,152,2,74,147,65,
06 00AC  20 0D3A  1831          65,153,255,2,75,78,46,74,188,0,75,74,189,1,75,105,76,192,255,1,
06 00AC  20 0D3A  1832          76,46,74,190,2,76,46,105,76,193,255,2,77,154,47,75,191,2(255),1,
06 00AC  20 0D3A  1833          79,147,65,154,2,79,147,65,65,155,255,2,80,46,103,66,166,2,80,46,
06 00AC  20 0D3A  1834          105,80,199,255,0,81,38,72,255,2,82,46,103,81,200,255,0,83,52,100,
06 00AC  20 0D3A  1835          18(255),0,101,1,1,0,101,2,2,0,101,3,3,0,101,4,4,0,101,5,5,0,101,
06 00AC  20 0D3A  1836          6,6,0,101,7,7,0,101,8,8,0,101,9,9,0,101,10,10,1,101,153,70,180,
06 00AC  20 0D3A  1837          3(255),1,104,101,14,15,2(255),2,106,13,101,29,42,3,106,13,114,
06 00AC  20 0D3A  1838          101,29,43,3,106,13,115,101,29,44,4,106,13,114,115,101,29,45,3,
06 00AC  20 0D3A  1839          106,13,113,101,29,46,2,106,113,101,29,47,3,106,13,110,101,33,60,
06 00AC  20 0D3A  1840          0,106,61,145,2(255),1,108,101,19,23,5(255),1,113,101,27,38,
06 00AC  20 0D3A  1841          4(255),1,117,101,37,68,255,1,118,119,40,74,255,0,119,58,137,255,
06 00AC  20 0D3A  1842          2,120,46,121,48,95,2(255),0,122,72,184,255,2,123,46,124,51,99,
06 00AC  20 0D3A  1843          3(255),1,126,54,53,106,255,1,127,54,53,107,2(255),0,129,53,113,
06 00AC  20 0D3A  1844          255,0,130,53,114,5(255),1,135,56,56,123,1,135,144,60,144,3(255),
06 00AC  20 0D3A  1845          0,138,58,131,255,0,139,58,132,255,1,140,58,58,134,255,1,141,58,
06 00AC  20 0D3A  1846          58,136,255,0,142,58,138,255,0,143,59,139,1,143,144,59,140,255,0,
06 00AC  20 0D3A  1847          144,60,143,255,0,145,59,141,1,145,144,59,142,255,2,146,67,146,63,
06 00AC  20 0D3A  1848          149,2(255),1,148,2,66,165,3(255),0,151,69,173,1,151,152,69,174,
06 00AC  20 0D3A  1849          4(255),1,155,101,77,194,255,2,156,46,157,78,195,255,0,157,78,196,
06 00AC  20 0D3A  1850          255,1,158,46,79,197,255,1,159,58,58,135,255);
06 00AC  20 0D3A  1851  ARRAY   5050  BYTE    MATRIX =
06 00AC  20 20F4  1852          (77(0),2,11(0),2,6(0),2,3(0),3(2),0,0,2,2,0,0,2,2,0,7(2),0,2,
06 00AC  20 20F4  1853          3(0),3(2),37(0),2,2,3(0),2,2,5(0),2,28(0),2,11(0),2,6(0),2,3(0),
06 00AC  20 20F4  1854          2,2,3,0,0,2,2,0,0,2,2,0,7(2),0,2,3(0),3(2),14(0),2,11(0),2,6(0),
06 00AC  20 20F4  1855          2,3(0),2,2,3,0,0,2,2,0,0,2,2,0,7(2),0,2,4(0),2,2,14(0),2,11(0),2,
06 00AC  20 20F4  1856          6(0),2,3(0),2,2,3,0,0,2,2,3(0),2,0,2,2,0,2,0,2,2,0,2,4(0),2,2,
06 00AC  20 20F4  1857          39(0),3,38(0),2,11(0),2,6(0),2,3(0),2,2,3(0),2,2,0,0,2,2,0,7(2),
06 00AC  20 20F4  1858          0,2,4(0),2,2,43(0),3,56(0),3,6(0),2,55(0),3,5(0),3,0,0,3,3,54(0),
06 00AC  20 20F4  1859          3,3,28(0),1,0,1,1,9(0),3,1,1,0,7(1),8(0),1,1,3(0),1,8(0),1,0,1,0,
06 00AC  20 20F4  1860          1,48(0),3,3,61(0),3,0,0,3,67(0),2,22(0),1,0,3(1),4(0),3,3(0),
06 00AC  20 20F4  1861          3(1),0,7(1),0,0,3,1,1,0,1,0,1,1,3(0),1,3(0),2,4(0),1,0,1,0,1,
06 00AC  20 20F4  1862          5(0),1,17(0),3,1,25(0),1,3(0),2,57(0),3,62(0),3,2,5(0),3,29(0),3,
06 00AC  20 20F4  1863          1,33(0),3,56(0),2,5(0),3,66(0),3,53(0),3,3,5(0),3,29(0),3,1,
06 00AC  20 20F4  1864          67(0),2,11(0),2,6(0),2,3(0),2,2,3(0),2,2,0,0,2,2,0,7(2),0,2,3(0),
06 00AC  20 20F4  1865          3,2,2,0,0,1,0,1,1,9(0),3,1,1,0,7(1),8(0),1,1,3(0),1,6(0),3,0,1,0,
06 00AC  20 20F4  1866          1,0,1,13(0),1,0,3(1),8(0),3,1,1,0,7(1),0,3,3(1),0,1,0,1,1,0,3,3,
06 00AC  20 20F4  1867          1,8(0),1,0,1,0,1,5(0),1,19(0),3,22(0),3,3,3(0),3,2,3(0),1,0,3,
06 00AC  20 20F4  1868          7(0),2,5(0),3,14(0),2,18(0),2,3(0),2,2,3(0),2,2,3(0),2,0,2,7(0),
06 00AC  20 20F4  1869          2,4(0),2,2,0,0,1,0,3(1),8(0),3(1),3,7(1),0,0,3,1,1,0,1,0,1,1,0,2,
06 00AC  20 20F4  1870          2,1,0,0,2,2,4(0),3(1),0,1,5(0),1,7(0),1,0,1,1,10(0),3,1,0,7(1),
06 00AC  20 20F4  1871          8(0),1,1,3(0),1,8(0),1,0,1,0,1,50(0),3,18(0),3,19(0),2,11(0),3,
06 00AC  20 20F4  1872          6(0),2,3(0),2,2,3(0),2,2,3(0),2,0,2,3,0,1,0,1,1,0,2,4(0),2,2,
06 00AC  20 20F4  1873          14(0),2,11(0),2,6(0),2,3(0),2,2,3(0),2,2,3(0),2,0,2,2,0,2,0,2,2,
06 00AC  20 20F4  1874          0,2,4(0),2,2,14(0),2,11(0),2,6(0),2,3(0),2,2,3(0),2,2,0,0,3,2,0,
06 00AC  20 20F4  1875          2,2,3,2,0,2,2,0,2,4(0),2,2,14(0),2,11(0),2,6(0),2,3(0),2,2,3(0),
06 00AC  20 20F4  1876          2,2,0,0,2,2,0,4(2),0,2,2,0,2,4(0),2,2,14(0),2,11(0),2,6(0),2,
06 00AC  20 20F4  1877          3(0),2,2,3(0),2,2,0,0,2,2,0,4(2),3,2,2,0,2,4(0),2,2,0,0,1,0,1,1,
06 00AC  20 20F4  1878          14(0),6(1),8(0),3,1,3(0),1,8(0),1,0,1,0,1,48(0),3,3,3(0),3,3,
06 00AC  20 20F4  1879          57(0),2,2,3(0),2,2,57(0),2,2,3(0),2,2,5(0),3,51(0),2,2,3(0),2,2,
06 00AC  20 20F4  1880          5(0),2,6(0),3,9(0),1,0,3(1),8(0),3,1,1,0,7(1),0,3,3(1),3,1,0,1,1,
06 00AC  20 20F4  1881          3(0),1,0,0,3,3,4(0),1,0,1,0,1,5(0),1,7(0),2,0,3(2),3,3(1),4(0),
06 00AC  20 20F4  1882          3(2),0,7(2),0,6(2),0,2,2,3(0),2,0,0,2,2,1,3(0),2,0,2,0,2,5(0),2,
06 00AC  20 20F4  1883          7(0),2,0,3(2),8(0),3(2),0,7(2),0,6(2),0,2,2,3(0),2,0,0,2,2,4(0),
06 00AC  20 20F4  1884          2,0,2,0,2,5(0),2,7(0),1,3(0),1,10(0),1,10(0),3,1,1,0,1,0,0,1,0,2,
06 00AC  20 20F4  1885          2,3(0),2,2,14(0),1,7(0),1,3(0),1,10(0),1,9(0),3,3(1),0,1,0,0,1,
06 00AC  20 20F4  1886          6(0),3,3,14(0),1,62(0),3,39(0),3,4(0),3,18(0),2,4(0),1,1,61(0),3,
06 00AC  20 20F4  1887          16(0),2,11(0),2,6(0),2,3(0),2,2,3(0),2,2,0,0,2,2,0,7(2),0,2,3(0),
```

```
06 00AC   20 20F4          1888          3(2),12(0),3(2),11(0),2,6(0),2,0,0,4(2),0,0,2,2,0,0,2,2,0,7(2),0,
06 00AC   20 20F4          1889          2,0,0,3,3(2),35(0),3,42(0),2,11(0),2,6(0),2,0,2,0,3(2),0,7(2),0,
06 00AC   20 20F4          1890          7(2),0,2,3(0),3(2),0,0,2,0,3(2),8(0),3(2),0,7(2),0,4(2),0,2,0,2,
06 00AC   20 20F4          1891          2,0,3(2),6(0),2,0,2,0,2,0,2,5(0),2,7(0),2,0,3(2),0,0,3,1,4(0),
06 00AC   20 20F4          1892          3(2),0,7(2),0,4(2),0,2,0,2,2,0,3(2),3,3(0),3,0,2,0,2,0,2,0,2,
06 00AC   20 20F4          1893          5(0),2,42(0),2,2,62(0),2,2,3(0),2,2,5(0),2,6(0),2,9(0),2,0,8(2),
06 00AC   20 20F4          1894          3(0),3(2),0,7(2),0,6(2),0,2,2,3(0),2,0,0,3(2),3(0),2,0,2,0,2,
06 00AC   20 20F4          1895          5(0),2,40(0),3,9(0),3,20(0),1,0,0,1,17(0),3,1,1,9(0),1,0,3,3,1,
06 00AC   20 20F4          1896          8(0),1,3(0),1,59(0),3,17(0),2,0,3(2),8(0),11(2),0,0,3(2),0,2,0,2,
06 00AC   20 20F4          1897          2,0,3(2),0,0,2,2,4(0),3(2),0,2,5(0),2,7(0),2,0,3(2),9(0),2,2,0,
06 00AC   20 20F4          1898          7(2),0,0,3(2),0,2,0,2,2,0,3(2),0,0,2,2,4(0),2,0,2,0,2,5(0),2,
06 00AC   20 20F4          1899          44(0),2,18(0),2,9(0),3,30(0),1,30(0),1,0,0,1,15(0),3,4(1),9(0),1,
06 00AC   20 20F4          1900          3(0),1,8(0),1,3(0),1,13(0),1,0,0,1,17(0),3,1,1,9(0),1,3(0),1,
06 00AC   20 20F4          1901          8(0),1,3(0),1,13(0),1,0,0,1,17(0),3,1,1,9(0),1,3(0),1,8(0),1,
06 00AC   20 20F4          1902          3(0),1,0,0,3,10(0),1,0,0,1,19(0),3,9(0),1,3(0),1,8(0),1,17(0),2,
06 00AC   20 20F4          1903          0,2,2,14(0),6(2),8(0),2,2,3(0),2,8(0),2,0,2,0,2,0,0,3,10(0),2,0,
06 00AC   20 20F4          1904          2,2,14(0),6(2),8(0),2,2,3(0),2,8(0),2,0,2,0,2,40(0),3,1,27(0),1,
06 00AC   20 20F4          1905          8(0),3,31(0),1,30(0),2,0,7(2),4(0),3(2),0,7(2),0,6(2),0,2,2,3(0),
06 00AC   20 20F4          1906          2,0,0,3(2),3(0),2,0,2,0,2,5(0),2,3,6(0),2,0,7(2),4(0),3(2),0,
06 00AC   20 20F4          1907          7(2),0,6(2),0,2,2,3(0),2,0,0,3(2),3(0),2,0,2,0,2,5(0),2,7(0),1,0,
06 00AC   20 20F4          1908          1,1,10(0),3,1,0,7(1),6(0),3,0,1,1,3(0),1,8(0),1,0,1,0,1,13(0),2,
06 00AC   20 20F4          1909          0,2,2,9(0),3(2),0,7(2),8(0),2,2,3(0),2,8(0),2,0,2,0,2,11(0));
06 00AC   20 20F4          1910     CLOSE BASE;

                        SEGMENT 20  NAME = SEG#20       LENGTH = 20F4  BASE REG = 14

06 0CAC   07 0E48          1911
06 00AC   07 0E48          1912     DUMMY BASE R11;   COMMENT COMPILER COMMON FORMAT;
06 00AC   21 0000          1913     INTEGER COMMTIME;
06 00AC   21 0004          1914     SHORT INTEGER COMMLINE, COMMPAGE;
06 00AC   21 0008          1915     ARRAY 3 LOGICAL COMMFLAGS;
06 00AC   21 0014          1916       BYTE NOGO SYN COMMFLAGS(0);
06 00AC   21 0014          1917       BYTE OUTFLAG SYN COMMFLAGS(1);
06 00AC   21 0014          1918       BYTE CHECKFLAG SYN COMMFLAGS(2);
06 00AC   21 0014          1919       BYTE CARDFLAG SYN COMMFLAGS(3);
06 00AC   21 0014          1920       ARRAY 2 LOGICAL TRACEBITS SYN COMMFLAGS(4);
06 00AC   21 0014          1921     SHORT INTEGER BLOCKLISTSIZE, NAMETABLESIZE;
06 00AC   21 0018          1922     INTEGER REFRECBASE, IDDIRBASE, IDLISTBASE, INPOINT;
06 00AC   21 0028          1923     INTEGER TREELINK, TREEBASE, TREETOP;
06 00AC   21 0034          1924      INTEGER TREEORG SYN TREETOP;
06 00AC   21 0034          1925     ARRAY 512 LOGICAL BLOCKLIST;
06 00AC   21 0834          1926       SHORT INTEGER  BLENGTH SYN BLOCKLIST(0);  COMMENT LENGTH OF NAMETAB;
06 00AC   21 0834          1927       SHORT INTEGER NPOINT  SYN BLOCKLIST(2);  COMMENT POINTER TO NAMETAB;
06 00AC   21 0834          1928     COMMENT  *** SIZE OF OTHER TABLES IS DYNAMIC *** ;
06 00AC   21 0834          1929     ARRAY 3 LOGICAL NAMETABLE;
06 00AC   21 0840          1930       SHORT INTEGER  IDLOC1  SYN NAMETABLE;
06 00AC   21 0840          1931       SHORT INTEGER  IDLOC2 SYN  NAMETABLE(2);
06 00AC   21 0840          1932       BYTE  HIERARCHY SYN NAMETABLE(2);
06 00AC   21 0840          1933       BYTE  PROGSEG    SYN  NAMETABLE(3);
06 00AC   21 0840          1934       SHORT INTEGER  SIMTYPEINFO SYN NAMETABLE (4);
06 00AC   21 0840          1935       SHORT INTEGER  TYPEINFO    SYN NAMETABLE (6);
06 00AC   21 0840          1936       SHORT INTEGER  TYPES       SYN NAMETABLE (8);
06 00AC   21 0840          1937       BYTE     TYPE         SYN NAMETABLE (8);
06 00AC   21 0840          1938       BYTE  SIMPLETYPE      SYN NAMETABLE (9);
06 00AC   21 0840          1939       SHORT INTEGER    IDNO        SYN NAMETABLE (10);
06 00AC   21 0840          1940       BYTE     VR           SYN NAMETABLE (6);
06 00AC   21 0840          1941       BYTE     DIMEN        SYN NAMETABLE (7);
06 00AC   21 0840          1942       BYTE  RCCLNUMBER     SYN NAMETABLE(7);
```

```
06 00AC   21 0840          1943   ARRAY 1  SHORT INTEGER REFRECLIST SYN B5;
06 00AC   21 0840          1944   ARRAY  1  LOGICAL IDDIR SYN B3;
06 00AC   21 0840          1945     SHORT INTEGER IDLENGTH SYN IDDIR;
06 00AC   21 0840          1946     SHORT INTEGER IDPOINT SYN IDDIR(2);
06 00AC   21 0840          1947   ARRAY  1  BYTE IDLIST SYN B3;
06 00AC   21 0840          1948   LOGICAL TREE SYN 0; COMMENT PASS TWO OUTPUT;
06 00AC   21 0840          1949   BYTE PROGRAM SYN 0; COMMENT PASS ONE OUTPUT;
06 00AC   21 0840          1950   CLOSE BASE;
06 00AC   07 0E48          1951
06 00AC   07 0E48          1952   INTEGER INPOINTSAVE;
06 00AC   07 0E4C          1953   SHORT INTEGER PATCHPOINT;
06 00AC   07 0E4E          1954   ARRAY 64  SHORT INTEGER PATCHLIST;
06 00AC   07 0ECE          1955
06 00AC   07 0ECE          1956
06 00AC   07 0ECE          1957   PROCEDURE   ERROR (R10);
06 00AC   07 0ECE          1958     BEGIN  LONG REAL DEC;
06 00B0   07 0ED8          1959       INTEGER R1SAVE,R2SAVE,R3SAVE,R4SAVE;
06 00B0   07 0EE8          1960       INTEGER TEMP2;
06 00B0   07 0EEC          1961       BYTE STACKFLAG;
06 00B0   07 0EED          1962       PROCEDURE IDWRITE(R4);
06 00B0   07 0EED          1963       BEGIN  R3 := IDDIRBASE;  R1 := R1 SHLL 2;  R2 := IDLENGTH(R1);
06 00C0   07 0EED          1964         IF R2 > 12 THEN R2 := 12;  R1 := IDPOINT(R1);  R3 := IDLISTBASE;
06 00D4   07 0EED          1965         R1 := @IDLIST(R1);
06 00D8   07 0EED          1966       END;
06 00DA   07 0EED          1967
06 00DA   07 0EED          1968     MVC(130,BUFFER(1),BUFFER);  STM(R1,R4,R1SAVE);  R3 := R0;
06 00E6   07 0EED          1969     R0 := @BUFFER; RESET(STACKFLAG);
06 00EE   07 0EED          1970     CASE R3 OF
06 00EE   07 0EED          1971       BEGIN
06 00EE   07 0EED          1972         BEGIN
06 00F6   07 0EED          1973           MVC(24,BUFFER,"MULTIPLY-DEFINED SYMBOL -");  IDWRITE;
06 0100   07 0EED          1974           EX(R2,ERRMOVE);
06 0104   07 0EED          1975         END;
06 0104   07 0EED          1976         BEGIN
06 0108   07 0EED          1977           IF R1 ¬= 0 THEN
06 010E   07 0EED          1978           BEGIN  MVC(17,BUFFER,"UNDEFINED SYMBOL -");  IDWRITE;
06 0118   07 0EED          1979             EX(R2,ERRMOVE);
06 011C   07 0EED          1980           END  ELSE GOTO L;
06 011C   07 0EED          1981         END;
06 011C   07 0EED          1982         BEGIN   MVC(12,BUFFER,"SYNTAX ERROR,");
06 0126   07 0EED          1983           SET(STACKFLAG);  R6SAVE := R6;
06 012E   07 0EED          1984         END;
06 012E   07 0EED          1985         MVC(34,BUFFER,"IDENTIFIER MUST BE RECORD CLASS ID,");
06 0138   07 0EED          1986         MVC(20,BUFFER,"MISMATCHED PARAMETER,");
06 0142   07 0EED          1987         MVC(37,BUFFER,"INCORRECT NUMBER OF ACTUAL PARAMETERS,");
06 014C   07 0EED          1988         MVC(19,BUFFER,"INCORRECT DIMENSION,");
06 0156   07 0EED          1989         MVC(18,BUFFER,"DATA AREA EXCEEDED,");
06 0160   07 0EED          1990         MVC(26,BUFFER,"INCORRECT NUMBER OF FIELDS,");
06 016A   07 0EED          1991         MVC(27,BUFFER,"INCOMPATIBLE STRING LENGTHS,");
06 0174   07 0EED          1992         MVC(23,BUFFER,"INCOMPATIBLE REFERENCES,");
06 017E   07 0EED          1993         MVC(22,BUFFER,"BLOCKS NESTED TOO DEEP,");
06 0188   07 0EED          1994         MVC(20,BUFFER,"INCORRECT I/O DEVICE,");
06 0192   07 0EED          1995         MVC(36,BUFFER,"REFERENCE MUST REFER TO RECORD CLASS,");
06 019C   07 0EED          1996         MVC(36,BUFFER,"EXPRESSION MISSING IN PROCEDURE BODY,");
06 01A6   07 0EED          1997         BEGIN  MVC(21,BUFFER,"INCORRECT SIMPLE TYPE ");
06 01B0   07 0EED          1998           R2 := RULENUMBER;  CVD(R2,DEC);  MVC(5,BUFFER(22),MASK2);
06 01BE   07 0EED          1999           ED(5,BUFFER(22),DEC(5));
06 01C4   07 0EED          2000         END;
```

```
06 01C4  07 0EED  2001        MVC(32,BUFFER,"RESULT PARAMETER MUST BE <T VAR>,");
06 01CE  07 0EED  2002        MVC(32,BUFFER,"PROCEDURE HEAD LACKS SIMPLE TYPE,");
06 01D8  07 0EED  2003        BEGIN  IC(R3,S(R8));  R3 := R3 * 12S + METABASE;
06 01E8  07 0EED  2004          MVC(11,BUFFER,B3);
06 01EE  07 0EED  2005          IF R6 = IDCODE THEN
06 01F6  07 0EED  2006          BEGIN  R1 := VALUE; MVC(11,BUFFER(27),"           "); IDWRITE;
06 0204  07 0EED  2007            EX(R2,ERRMOVE);
06 0208  07 0EED  2008            R2 := @BUFFER(R2+28);  MVI(",",B2);
06 0210  07 0EED  2009          END ELSE
06 0210  07 0EED  2010          BEGIN  R3 := R6 * 12S + METABASE;
06 021E  07 0EED  2011            MVC(11,BUFFER(27),B3);  MVI(",",BUFFER(39));
06 0228  07 0EED  2012          END;
06 0228  07 0EED  2013          MVC(11,BUFFER(13),"UNRELATED TO");
06 022E  07 0EED  2014          GOTO LL;
06 0232  07 0EED  2015        END;
06 0232  07 0EED  2016        MVC(22,BUFFER,"ARRAY USED INCORRECTLY,");
06 023C  07 0EED  2017        MVC(31,BUFFER,"TOO MANY CONSTANTS IN PROCEDURE,");
06 0246  07 0EED  2018        MVC(23,BUFFER,"INCORRECT STRING LENGTH,");
06 0250  07 0EED  2019        MVC(23,BUFFER,"COMPILER TABLE OVERFLOW,");
06 025A  07 0EED  2020      END;
06 02BA  07 0EED  2021      MVC(16,BUFFER(67),"CURRENT SYMBOL IS");
06 02C0  07 0EED  2022      R1 := R6SAVE;
06 02C4  07 0EED  2023      IF R1 = IDCODE THEN
06 02CC  07 0EED  2024      BEGIN  R1 := VALUE;  IDWRITE;  EX(R2,ERRMOVE91);
06 02D8  07 0EED  2025      END ELSE
06 02D8  07 0EED  2026      BEGIN  R2 := R1 * 12S + METABASE;
06 02E6  07 0EED  2027        MVC(11,BUFFER(91),B2);
06 02EC  07 0EED  2028      END;
06 02EC  07 0EED  2029  LL:
06 02EC  07 0EED  2030      R2 := CARDNUMBER;  CVD(R2,DEC);
06 02F4  07 0EED  2031      MVC(15,BUFFER(41),"CARD NUMBER IS  ");
06 02FA  07 0EED  2032      MVC(5,BUFFER(56),#4020202021214040L);
06 0300  07 0EED  2033      ED(5,BUFFER(56),DEC(5));  MVI(".",BUFFER(62));  PRINT;
06 030E  07 0EED  2034      R3 := 0;  IC(R3,OUTFLAG);
06 0316  07 0EED  2035      IF STACKFLAG AND R3 >=3 THEN
06 0326  07 0EED  2036      BEGIN  MVI(" ",BUFFER(41));
06 032A  07 0EED  2037        MVC(62,BUFFER(42),BUFFER(41));
06 0330  07 0EED  2038        MVC(14,BUFFER,"STACK CONTAINS:");  PRINT;
06 033A  07 0EED  2039        MVC(14,BUFFER,BUFFER(41));  R3 := @S(R7);
06 0344  07 0EED  2040        TEMP2 := R3;
06 0348  07 0EED  2041        FOR R3 := @ S STEP 1 UNTIL TEMP2 DO
06 034C  07 0EED  2042        BEGIN  R2 := 0;  IC(R2,B3);  R2 := R2 * 12S + METABASE;
06 0360  07 0EED  2043          MVC(11,BUFFER(2),B2);  PRINT;
06 036A  07 0EED  2044        END;
06 0376  07 0EED  2045      END;
06 0376  07 0EED  2046  L:  R0 := 0;
06 037A  07 0EED  2047      LM(R1,R4,R1SAVE);  MVI(" ",BUFFER);
06 0382  07 0EED  2048      SET(NOGO);
06 0386  07 0EED  2049    END;
06 0388  07 0EED  2050
06 0388  07 0EED  2051  PROCEDURE  REFCOMPATIBLE  (R10);
06 0388  07 0EED  2052  COMMENT  COMPARES REFERENCE INFO IN R1 TO REFERENCE INFO IN R4;
06 0388  07 0EED  2053  BEGIN  INTEGER  RASAVE;
06 0388  07 0EF4  2054    RASAVE := R10;  R1 := R1 AND R4;
06 038E  07 0EF4  2055    IF = THEN
06 0392  07 0EF4  2056    BEGIN R0 := @REFERROR; ERROR;
06 039A  07 0EF4  2057    END;
06 039A  07 0EF4  2058    R10 := RASAVE;
```

```
06 039E  07 0EF4  2059  END;
06 03A0  07 0EF4  2060
06 03A0  07 0EF4  2061  PROCEDURE CNVRTASSN(R10);
06 03A0  07 0EF4  2062  COMMENT   LEFT SIMPLE TYPE IN R0, ITS SIMTYPEINFO IN R4, RIGHT
06 03A0  07 0EF4  2063     SIMPLE TYPE IN V(J+1);
06 03A0  07 0EF4  2064    BEGIN   INTEGER  RASAVE;
06 03A0  07 0EF8  2065      RASAVE := R10;  R1 := 0;  IC(R1,V22(R8+8));
06 03AC  07 0EF8  2066      IF R0 ¬= R1 THEN
06 03B2  07 0EF8  2067      BEGIN
06 03B2  07 0EF8  2068        IF R0 >5 THEN  BEGIN  R0 := @TYPEERROR;  ERROR;
06 03C2  07 0EF8  2069                       END  ELSE
06 03C2  07 0EF8  2070        IF R0 = 1 THEN  BEGIN  R0 := @TYPEERROR;  ERROR;
06 03D6  07 0EF8  2071                        END  ELSE
06 03D6  07 0EF8  2072        IF R0 = 0 THEN
06 03E0  07 0EF8  2073        BEGIN  R0 := @PPROCERROR;  ERROR;
06 03E8  07 0EF8  2074        END ELSE
06 03E8  07 0EF8  2075        IF R0 < = 3 THEN
06 03F4  07 0EF8  2076          BEGIN  IF R1 > 3 THEN  BEGIN R0 := @TYPEERROR;  ERROR;
06 0404  07 0EF8  2077                                 END  ELSE STC(R0,CONV(R9));
06 040C  07 0EF8  2078          END  ELSE
06 040C  07 0EF8  2079        IF R1 > 5 THEN  BEGIN  R0 := @TYPEERROR;  ERROR;
06 0420  07 0EF8  2080                        END  ELSE  STC(R0,CONV(R9));
06 0428  07 0EF8  2081        R0 := @ACOLONEQ;
06 042C  07 0EF8  2082      END  ELSE
06 042C  07 0EF8  2083      IF R1 = 7 THEN
06 0438  07 0EF8  2084      BEGIN   R1 := V1(R8+8);
06 043C  07 0EF8  2085        IF R1 > R4 THEN  BEGIN  R0 := @LENGTHERROR;  ERROR;
06 044A  07 0EF8  2086                         END;
06 044A  07 0EF8  2087        R0 := @SCOLONEQ;
06 044E  07 0EF8  2088      END  ELSE IF R1 = 9 THEN
06 045A  07 0EF8  2089      BEGIN   R1 := V1(R8+8);  REFCOMPATIBLE;
06 0462  07 0EF8  2090        R0 := @RCOLONEQ;
06 0466  07 0EF8  2091      END  ELSE  IF R1 = 6 THEN  R0 := @LCOLONEQ ELSE R0 := @ACOLONEQ;
06 047E  07 0EF8  2092      R10 := RASAVE;
06 0482  07 0EF8  2093    END;
06 0484  07 0EF8  2094
06 0484  07 0EF8  2095  PROCEDURE  SAMETYPE(R10);
06 0484  07 0EF8  2096  COMMENT  CHECKS THAT R0 AND V22(J+1) ARE SAME TYPE AND ARE STRING OR
06 0484  07 0EF8  2097           REFERENCE COMPATIBLE.   R4 CONTAINS SIMTYPEINFO FOR R0;
06 0484  07 0EF8  2098    BEGIN  INTEGER RASAVE;
06 0484  07 0EFC  2099      RASAVE := R10;  IC(R1,V22(R8+8));
06 048C  07 0EFC  2100      IF R0 ¬= R1 THEN
06 0492  07 0EFC  2101        BEGIN  R0 := @TYPEERROR;    ERROR;
06 049A  07 0EFC  2102        END   ELSE   IF R1 = 7 THEN
06 04A6  07 0EFC  2103        BEGIN   R1 := V1(R8+8);
06 04AA  07 0EFC  2104          IF R1 ¬= R4 THEN
06 04B0  07 0EFC  2105          BEGIN   R0 := @LENGTHERROR;    ERROR;
06 04B8  07 0EFC  2106          END;
06 04B8  07 0EFC  2107        END  ELSE  IF R1 = 9 THEN
06 04C4  07 0EFC  2108        BEGIN   R1 := V1(R8+8);   REFCOMPATIBLE;
06 04CC  07 0EFC  2109        END;
06 04CC  07 0EFC  2110      R10 := RASAVE;
06 04D0  07 0EFC  2111    END;
06 04D2  07 0EFC  2112  PROCEDURE APAREXPCHECK (R5);
06 04D2  07 0EFC  2113  COMMENT   SHOULD BE IN EXECUTE2;
06 04D2  07 0EFC  2114  COMMENT  POINTER TO NAMETABLE IN R2;
06 04D2  07 0EFC  2115    BEGIN  IC(R0,SIMPLETYPE(R2));
06 04D6  07 0EFC  2116      R3 := 0;  IC(R3,VR(R2));
```

```
06 04DE  07 0EFC  2117      R4 := SIMTYPEINFO(R2);
06 04E2  07 0EFC  2118      IF R3 ¬= 0 THEN
06 04E8  07 0EFC  2119        BEGIN
06 04E8  07 0EFC  2120          CASE R3 OF
06 04E8  07 0EFC  2121          BEGIN
06 04E8  07 0EFC  2122            CNVRTASSN;
06 04F4  07 0EFC  2123            BEGIN    COMMENT RESULT;
06 04F8  07 0EFC  2124              IC(R3,OP(R9));  R3 := R3 AND #7F;
06 0500  07 0EFC  2125              IF R3 ¬=30 AND R3 ¬= 31 AND R3 ¬= 40 AND R3 ¬= 62 AND
06 0520  07 0EFC  2126                R3 ¬= 87 AND R3 ¬= 89 THEN
06 0530  07 0EFC  2127              BEGIN R0 := @RESULTERROR;  ERROR;
06 0538  07 0EFC  2128              END ELSE
06 0538  07 0EFC  2129              BEGIN R3 := R0;  IC(R0,V22(R8+8));  STC(R3,V22(R8+8));
06 0546  07 0EFC  2130                R3 := R4; R4 := V1(R8+8);  V1(R8+8) := R3;
06 0550  07 0EFC  2131                CNVRTASSN; R0 := 0; STC(R0,CONV(R9));
06 055C  07 0EFC  2132              END;
06 055C  07 0EFC  2133            END;
06 055C  07 0EFC  2134            BEGIN  COMMENT  VALUE-RESULT;
06 0560  07 0EFC  2135              IC(R3,OP(R9));  R3 := R3 AND #7F;
06 0568  07 0EFC  2136              IF R3 ¬= 30 AND R3 ¬= 31 AND R3 ¬= 40 AND R3 ¬= 62 AND
06 0588  07 0EFC  2137              R3 ¬= 87 AND R3 ¬= 89 THEN
06 0598  07 0EFC  2138              BEGIN  R0 := @RESULTERROR;  ERROR;
06 05A0  07 0EFC  2139              END;
06 05A0  07 0EFC  2140              IC(R1,V22(R8+8));
06 05A4  07 0EFC  2141              IF R0 = 2 AND R1 = 3 THEN NULL ELSE
06 05B4  07 0EFC  2142              IF R0 = 3 AND R1 = 2 THEN NULL ELSE
06 05C8  07 0EFC  2143              IF R0 = 4 AND R1 = 5 THEN NULL ELSE
06 05DC  07 0EFC  2144              IF R0 = 5 AND R1 = 4 THEN NULL ELSE
06 05F0  07 0EFC  2145              SAMETYPE;
06 05F8  07 0EFC  2146            END;
06 05F8  07 0EFC  2147          END;
06 0608  07 0EFC  2148          IC(R3,V21(R8+8));
06 060C  07 0EFC  2149          IF R3¬=0 AND R3 ¬= #13 THEN
06 061A  07 0EFC  2150          BEGIN R3 := POINTER(R9);  R3 := TYPEINFO(R3);
06 0622  07 0EFC  2151            IF R3 ¬= 0 THEN
06 0628  07 0EFC  2152            BEGIN  R0 := @TYPE3ERROR; ERROR;
06 0630  07 0EFC  2153            END;
06 0630  07 0EFC  2154          END;
06 0630  07 0EFC  2155        END  ELSE
06 0630  07 0EFC  2156        BEGIN  IC(R3,TYPE(R2));  SAMETYPE;  R1 := 0;
06 064C  07 0EFC  2157          IC(R1,V21(R8+8));  R1 := R1 OR #10;
06 0648  07 0EFC  2158          IF R1 ¬= R3 THEN
06 064E  07 0EFC  2159          BEGIN   IF R3 = #10 AND R1 = #13 THEN
06 065E  07 0EFC  2160            BEGIN  R1 := POINTER(R9); R1 := TYPEINFO(R1);
06 0666  07 0EFC  2161              IF R1 ¬= 0 THEN
06 066C  07 0EFC  2162              BEGIN  R0 := @TYPE3ERROR; ERROR;
06 0674  07 0EFC  2163              END;
06 0674  07 0EFC  2164            END ELSE
06 0674  07 0EFC  2165            BEGIN  R0 := @TYPE3ERROR;    ERROR;
06 0680  07 0EFC  2166            END;
06 0680  07 0EFC  2167          END  ELSE IF R1 = #12 THEN
06 068C  07 0EFC  2168          BEGIN  IC(R0,DIMEN(R2));   R1 := V34(R8+8);
06 0694  07 0EFC  2169            IF R1 = 0 THEN
06 069A  07 0EFC  2170            BEGIN  COMMENT INSERT DIMEN IN NAMETABLE;
06 069A  07 0EFC  2171              R1 := POINTER(R9);  STC(R0,DIMEN(R1));
06 06A2  07 0EFC  2172            END ELSE
06 06A2  07 0EFC  2173            IF R0 ¬= R1 THEN
06 06AC  07 0EFC  2174            BEGIN  R0 := @ARRAYERROR;   ERROR;
```

```
06 06B4  07 0EFC  2175          END;
06 06B4  07 0EFC  2176        END;
06 06B4  07 0EFC  2177      END;
06 06B4  07 0EFC  2178    END;
06 06B6  07 0EFC  2179
06 06B6  07 0EFC  2180    PROCEDURE  REGPATH          (R10);
06 06B6  07 0EFC  2181      BEGIN
06 06B6  07 0EFC  2182        R2 := V34(R8);  R3 := V34(R7);
06 06BE  07 0EFC  2183        IF R2<R3 THEN
06 06C4  07 0EFC  2184        BEGIN  V34(R8) := R3;  R0 := R0 OR #80;
06 06CC  07 0EFC  2185        END ELSE
06 06CC  07 0EFC  2186        IF R2 = R3 THEN
06 06D6  07 0EFC  2187        BEGIN  R1 := 0;  IC(R1,V22(R8));  R1 := R1 SHLL 1;
06 06E2  07 0EFC  2188          R2 := R2 + INCREASE(R1);  V34(R8) := R2;
06 06EA  07 0EFC  2189          IF R0 = AASSIGN OR R0 = LASSIGN OR R0 = SASSIGN OR R0 = RASSIGN
06 0702  07 0EFC  2190           THEN R0 := R0 OR #80;  COMMENT RESOLVE ASSIGNMENT TO RIGHT;
06 070E  07 0EFC  2191        END;
06 070E  07 0EFC  2192      END;
06 0710  07 0EFC  2193
06 0710  07 0EFC  2194    PROCEDURE CARDOUT(R10);
06 0710  07 0EFC  2195      BEGIN
06 0710  07 0EFC  2196        R9 := R9 + 4;  R0 := @CARD;  STC(R0,OP(R9));  V5(R8) := R9;
06 0720  07 0EFC  2197        R2 := CARDNUMBER; POINTER(R9) := R2;
06 0728  07 0EFC  2198      END;
06 072A  07 0EFC  2199
06 072A  07 0EFC  2200    PROCEDURE LITERAL (R10);
06 072A  07 0EFC  2201    COMMENT  UPON ENTRY, POINTER TO LITERAL IS IN R4, SIMPLE TYPE IS IN R0,
06 072A  07 0EFC  2202          LENGTH-1 OF STRING IS IN R5.  UPON RETURN, POINTER TO CONSTANT-
06 072A  07 0EFC  2203          TABLE IS IN R5;
06 072A  07 0EFC  2204    BEGIN  INTEGER R1SAVE,R2SAVE,R3SAVE;
06 072A  07 0F08  2205      STM(R1,R3,R1SAVE);  R1 := R0;  R3 := LITBASE;
06 0734  07 0F08  2206      CASE R1 OF
06 0734  07 0F08  2207      BEGIN
06 0734  07 0F08  2208        R5 := 3; R5 := 3; R5 := 7; R5 := 7; R5 := 15; R5 := 0;
06 0768  07 0F08  2209        BEGIN  R5 := R5 SHLL 8;  R0 := R0 OR R5; R5 := R5 SHRL 8;
06 0776  07 0F08  2210        END;
06 0776  07 0F08  2211        R5 := 3;
06 077E  07 0F08  2212      END;
06 07A2  07 0F08  2213      R1 := CTORG;
06 07A6  07 0F08  2214      WHILE R1 < CTPNT DO
06 07AE  07 0F08  2215      BEGIN
06 07AE  07 0F08  2216        IF R0 ¬= CINFO(R1) THEN R1 := R1 + 4 ELSE
06 07BA  07 0F08  2217        BEGIN   R2 := CADDR(R1);  R2 := @LITERALTABLE(R2);
06 07C6  07 0F08  2218          R2 := R2 + LITORG;
06 07CA  07 0F08  2219          EX(R5,COMPARE24); IF = THEN
06 07D2  07 0F08  2220            BEGIN R5 := R1 - CTORG;  GOTO L;
06 07DC  07 0F08  2221            END  ELSE R1 := R1 + 4;
06 07E4  07 0F08  2222        END;
06 07E4  07 0F08  2223      END;
06 07E8  07 0F08  2224      R1 := CTPNT;
06 07EC  07 0F08  2225      IF R1 >= 1024 THEN
06 07F4  07 0F08  2226      BEGIN  R0 := @CONSTANTERROR;  ERROR;
06 07FC  07 0F08  2227      END ELSE
06 07FC  07 0F08  2228      BEGIN
06 0800  07 0F08  2229        CINFO(R1) := R0;  R0 := R0 AND #FF;  R2 := LITPNT;
06 080C  07 0F08  2230        IF R0 = 3 OR R0 = 5 THEN R2 := R2 + 7 SHRL 3 SHLL 3 ELSE
06 0828  07 0F08  2231        IF R0 ¬= 7 THEN R2 := R2 + 3 SHRL 2 SHLL 2;
06 0840  07 0F08  2232        R0 := R2 - LITORG;  CADDR(R1) := R0;  R0 := R1;
```

```
06 084C  07 0F08  2233      R1 := @LITERALTABLE(R2);  EX(R5,MOVE14);  R2 := @B2(R5+1);
C6 C858  07 0F08  2234      LITPNT := R2;  R5 := R0 - CTORG;  R0 := R0 + 4;  CTPNT := R0;
06 086A  07 0F08  2235    END;
06 086A  07 0F08  2236  L:R0 := 0;  LM(R1,R3,R1SAVE);
06 C872  07 0F08  2237  END;
06 C874  07 0F08  2238
06 C874  07 0F08  2239  SEGMENT PROCEDURE WRITETREE(R10);
06 C874  07 0F08  2240  COMMENT  WRITES TREE, CONSTANT POINTER TABLE, AND CONSTANT TABLE;
06 0874  07 0F08  2241    BEGIN   INTEGER  R1SAVE,R2SAVE,R3SAVE,R4SAVE,TEMP;
22 0000  07 0F1C  2242      STM(R1,R4,R1SAVE);    MVC(130,BUFFER(1),BUFFER);
22 000A  07 0F1C  2243      R1 := R1 + R4 - 4;  TEMP := R1;  R0 := @BUFFER;  OI("0",CARRCONT);
22 001C  07 0F1C  2244      MVC(16,BUFFER(1),"PROGRAM SEGMENT  ");  MVC(5,BUFFER(18),MASK2);
22 0028  07 0F1C  2245      R3 := POINTER(R4);   IC(R3,PROGSEG(R3)); R3 := R3 AND #FF;
22 0034  07 0F1C  2246      CVD(R3,DEC);  ED(5,BUFFER(18),DEC(5));  PRINT;
22 004A  07 0F1C  2247      MVC(36,BUFFER(1),"     LOC  FLAG  OPCODE  CONV  POINTER"); PRINT;
22 005C  07 0F1C  2248      MVC(130,BUFFER(1),BUFFER);  MVI("0",BUFFER(4));  R3 := 0;
22 006A  07 0F1C  2249      MVC(2,BUFFER(5),BUFFER(4)); UNPK(4,2,BUFFER(34),R1SAVE(2));
22 0C76  07 0F1C  2250      MVI(" ",BUFFER(38));  TR(3,BUFFER(34),TRANSTABLE(_240));
22 0080  07 0F1C  2251      PRINT;  R1 := 0;
22 0090  07 0F1C  2252      FOR R4 := R4 STEP 4 UNTIL TEMP DO
22 0090  07 0F1C  2253      BEGIN
22 0094  07 0F1C  2254        R3 := R3 + 4;    HEX := R3;  UNPK(4,2,BUFFER(4),HEX(2));
22 00A2  07 0F1C  2255        MVI(" ",BUFFER(8));  TR(3,BUFFER(4),TRANSTABLE(_240));
22 00AC  07 0F1C  2256        IC(R1,OP(R4));
22 00B0  07 0F1C  2257        IF R1 > #7F THEN
22 0088  07 0F1C  2258        BEGIN  R2 := 1; R1 := R1 AND #7F;
22 00C0  07 0F1C  2259        END ELSE R2 := 0;
22 00C8  07 0F1C  2260        R2 := R2 OR #F0;  STC(R2,BUFFER(13));   R2 := R1 SHLL 3;
22 00D6  07 0F1C  2261        R2 := @OPTABL(R2);   MVC(7,BUFFER(16),B2);
22 00E0  07 0F1C  2262        MVI(" ",BUFFER(25)); MVC(4,BUFFER(26),BUFFER(25));
22 00EA  07 0F1C  2263        R2 := @BEGINN;
22 00EE  07 0F1C  2264        IF R1=R2 THEN
22 00F4  07 0F1C  2265        BEGIN  R1 := P(R4) SHRL 12 AND #FFF;
22 0100  07 0F1C  2266          IF R1 ¬= 0 THEN
22 0106  07 0F1C  2267          BEGIN  CVD (R1,DEC); MVC(5,BUFFER(25),MASK);
22 0110  07 0F1C  2268            ED(5,BUFFER(25),DEC(5));
22 0116  07 0F1C  2269          END;
22 0116  07 0F1C  2270          R2 := POINTER(R4) AND #FFF;
22 011E  07 0F1C  2271        END ELSE
22 011E  07 0F1C  2272        BEGIN  IC(R1,CONV(R4));
22 0126  07 0F1C  2273          IF R1 ¬= 0 THEN
22 012C  07 0F1C  2274          BEGIN  CVD (R1,DEC); MVC(5,BUFFER(25),MASK);
22 0136  07 0F1C  2275            ED(5,BUFFER(25),DEC(5));
22 013C  07 0F1C  2276          END;
22 013C  07 0F1C  2277          R2 := POINTER(R4);
22 0140  07 0F1C  2278        END;
22 0140  07 0F1C  2279        HEX := R2;  UNPK(4,2,BUFFER(34),HEX(2));
22 014A  07 0F1C  2280        MVI(" ",BUFFER(38)); TR(3,BUFFER(34),TRANSTABLE(_240));
22 0154  07 0F1C  2281        PRINT;  R1 := 0;
22 0164  07 0F1C  2282      END;
22 017C  07 0F1C  2283      MVC(38,BUFFER(1),BUFFER);  PRINT;
22 0182  07 0F1C  2284      MVC(15,BUFFER(1),"LITERAL ORIGIN -");
22 0168  07 0F1C  2285      UNPK(4,2,BUFFER(18),LABELADDR(2));
22 018E  07 0F1C  2286      TR(3,BUFFER(18),TRANSTABLE(_240));  MVI(" ",BUFFER(22));  PRINT;
22 01A4  07 0F1C  2287      MVC(20,BUFFER(1),"LITERAL POINTER TABLE");  PRINT;
22 01B6  07 0F1C  2288      MVC(28,BUFFER(1),"   LOC  LENGTH  TYPE  POINTER");   PRINT;
22 01C8  07 0F1C  2289      MVC(28,BUFFER(1),BUFFER);  R4 := CTORG;  R1 := 0;
22 01D6  07 0F1C  2290      R2 := R4 + CTPNT - CTORG - 4; TEMP := R2;  R3 := 0;
```

```
22 01EC  07 0F1C  2291        FOR R4 := R4 STEP 4 UNTIL TEMP DO
22 01EC  07 0F1C  2292        BEGIN HEX := R3;  UNPK(4,2,BUFFER(3),HEX(2));MVI(" ",BUFFER(7));
22 01FE  07 0F1C  2293          TR(3,BUFFER(3),TRANSTABLE(_240));  R2 := CADDR(R4); HEX := R2;
22 020C  07 0F1C  2294          UNPK(4,2,BUFFER(26),HEX(2));
22 0212  07 0F1C  2295          TR(3,BUFFER(26),TRANSTABLE(_240));  MVI(" ",BUFFER(30));
22 021C  07 0F1C  2296          MVC(5,BUFFER(15),MASK2);  IC(R1,CTYPE(R4)); CVD(R1,DEC);
22 022A  07 0F1C  2297          ED(5,BUFFER(15),DEC(5));
22 0230  07 0F1C  2298          IF R1 = 7 THEN MVC(5,BUFFER(9),MASK2 )ELSE
22 023E  07 0F1C  2299          MVC(5,BUFFER(9),MASK); IC(R1,CLENGTH(R4));   CVD(R1,DEC);
22 0250  07 0F1C  2300          ED(5,BUFFER(9),DEC(5));  PRINT; R1 := 0;   R3 := R3 + 4;
22 026A  07 0F1C  2301        END;
22 0276  07 0F1C  2302        MVC(30,BUFFER(1),BUFFER);   PRINT;    R3 := LITBASE;
22 028C  07 0F1C  2303        MVC(12,BUFFER(1),"LITERAL TABLE"); PRINT;   R4 := LITORG;
22 02A2  07 0F1C  2304        R0 := @ LITERALTABLE(R4);  R1 := LITPNT - LITORG;  DUMP;  R0 := 0;
22 02BE  07 0F1C  2305        R14 := SAVE14;  LM(R1,R4,R1SAVE);
22 02C6  07 0F1C  2306      END;

                  SEGMENT 22  NAME = SEG#22      LENGTH = 0370  BASE REG = 15

06 0874  07 0F1C  2307
06 0874  07 0F1C  2308    SEGMENT PROCEDURE PASS2(R10);
06 0874  07 0F1C  2309    BEGIN  INTEGER RASAVE;
23 0000  07 0F20  2310
23 0000  07 0F20  2311    PROCEDURE  DECLARETEST (R10 );
23 0000  07 0F20  2312    COMMENT  DECLARETEST FINDS THE ENTRY IN THE NAME TABLE (IF ONE EXISTS)
23 0000  07 0F20  2313      FOR THE ID INDICATED IN V1(I)(CHECKS WHETHER IT IS MULTIPLY DECLARED,
23 0000  07 0F20  2314      INSERTS THE RELATIVE NAME TABLE ADDRESS IN V1(J);
23 0000  07 0F20  2315
23 0000  07 0F20  2316      BEGIN
23 0004  07 0F20  2317          SHORT INTEGER TEMP;     INTEGER  RASAVE;
23 0004  07 0F28  2318          INTEGER  R1SAVE,R2SAVE,R3SAVE,R4SAVE;
23 0004  07 0F38  2319          STM(R1,R4,R1SAVE);
23 0008  07 0F38  2320          R1 := V1(R7);
23 000C  07 0F38  2321          R2:= BNC SHLA  1;
23 0014  07 0F38  2322    LOOP:  R3 := NPOINT(R2);
23 0018  07 0F38  2323          R4:= BLENGTH(R2)+  R3;
23 001E  07 0F38  2324          WHILE R3 < R4  DO
23 0024  07 0F38  2325             IF R1=IDNO(R3) THEN GOTO  SUCCESS  ELSE R3 := R3 + 12;
23 0034  07 0F38  2326          IF R2 ¬= 0 THEN
23 003A  07 0F38  2327          BEGIN  R2 := R2 SHRL 1;  R2 := BLOCKLIST2(R2) SHLL 1;
23 0046  07 0F38  2328          END ELSE
23 0046  07 0F38  2329          BEGIN R0 := @UNDECL;  RASAVE := R10;  ERROR;  R10 := RASAVE;
23 0062  07 0F38  2330            SET(UNDECLFLAG);  V1(R8) := R2;  R1 := R6SAVE;
23 006E  07 0F38  2331            IF R1 = LPARENCODE THEN R2 := #301 ELSE
23 007A  07 0F38  2332            BEGIN  R2 := R7 SHRL 3;  IC(R1,S(R2-1));
23 0088  07 0F38  2333              IF R1 = GOTOCODE THEN R2 := #100 ELSE R2 := 1;
23 009C  07 0F38  2334            END;
23 009C  07 0F38  2335            V2(R8) := R2; GOTO L;
23 00A4  07 0F38  2336          END;
23 00A4  07 0F38  2337          GOTO LOOP;
23 00A8  07 0F38  2338    SUCCESS: V1(R8):= R3; COMMENT POINTER TO NAME TABLE;
23 00AC  07 0F38  2339            TEMP:=R4; R4:=TYPES(R3);
23 00B4  07 0F38  2340            R4 := R4 AND #EFFF;   COMMENT MASK OUT FPAR BIT;
23 00B8  07 0F38  2341            V2(R8):=R4;
23 00BC  07 0F38  2342            R3 := @B3(12);        COMMENT CHECK FOR MULTIPLE DECLARATION;
23 00C0  07 0F38  2343            WHILE  R3< TEMP DO
23 00C8  07 0F38  2344              IF  R1=IDNO(R3) THEN
23 00D0  07 0F38  2345                 BEGIN
```

```
23 00D0  07 0F38  2346                  R0 := @MULTDEF;  RASAVE := R10;  ERROR;  R10 := RASAVE;
23 00E8  07 0F38  2347                  GOTO L;
23 00EC  07 0F38  2348                END
23 00EC  07 0F38  2349               ELSE R3 := @B3(12);
23 00F8  07 0F38  2350  L:     LM(R1,R4,R1SAVE);
23 00FC  07 0F38  2351    END;
23 00FE  07 0F38  2352
23 00FE  07 0F38  2353  PROCEDURE   BLOCKSTEP (R10);
23 00FE  07 0F38  2354    BEGIN
23 00FE  07 0F38  2355        R1 := BN + 2;              BN := R1;  COMMENT   STEP BN;
23 010A  07 0F38  2356        R2 := BNC; BLOCKLIST2(R1) := R2;      COMMENT   CHAIN IN BLOCKLIST2;
23 0112  07 0F38  2357        BNC := R1;                            COMMENT   RESET BNC;
23 0116  07 0F38  2358    END;
23 0118  07 0F38  2359
23 0118  07 0F38  2360  PROCEDURE BLOCKEXIT (R5);
23 0118  07 0F38  2361    BEGIN   R1 := BNC;
23 011C  07 0F38  2362      IF R1 ¬= 2 AND R6 ¬= ENDFILE THEN
23 012C  07 0F38  2363      BEGIN R1 := BLOCKLIST2(R1);  BNC := R1;
23 0134  07 0F38  2364      END;
23 0134  07 0F38  2365      R2 := V2(R8);
23 0138  07 0F38  2366      IF R2 = #FF THEN
23 0140  07 0F38  2367      BEGIN  R1 := HN+1;  HN := R1;
23 014C  07 0F38  2368        R2 := DRELAD;  R1 := V1(R8) + OUTBASE;
23 0158  07 0F38  2369        R1 := POINTER(R1) + OUTBASE - 4;
23 0164  07 0F38  2370        IF R2 > #FFF THEN
23 016C  07 0F38  2371        BEGIN  R0 := @DATAERROR;  ERROR;
23 017C  07 0F38  2372        END;
23 017C  07 0F38  2373      R2 := R2 OR P(R1);  P(R1) := R2;
23 0184  07 0F38  2374      R1 := VARORIGIN-4;  VARORIGIN := R1;
23 0190  07 0F38  2375      R1 := V34(R8);  DRELAD := R1;        COMMENT RESTORE DRELAD;
23 0198  07 0F38  2376      END;
23 0198  07 0F38  2377    END;
23 019A  07 0F38  2378  PROCEDURE    TYPEINTEGER  (R5);
23 019A  07 0F38  2379  COMMENT  TYPEINTEGER CHECKS THAT R0 AND R1 ARE BOTH TYPE INTEGER;
23 019A  07 0F38  2380    IF R0 ¬= 1 THEN
23 01A2  07 0F38  2381    BEGIN   R0 := @TYPEERROR;   ERROR;
23 01B2  07 0F38  2382    END  ELSE  IF R0 ¬= R1 THEN
23 01BC  07 0F38  2383    BEGIN  R0 := @TYPEERROR;   ERROR;
23 01CC  07 0F38  2384    END;
23 01CE  07 0F38  2385
23 01CE  07 0F38  2386
23 01CE  07 0F38  2387  PROCEDURE  OUTID (R10);
23 01CE  07 0F38  2388  COMMENT  OUTID PUTS TERMINAL NODE AND POINTER TO NAME TABLE IN OUTPUT
23 01CE  07 0F38  2389     STRING.  TAKES TERMINAL NODE FROM R0.  LEAVES POINTER TO NAMETABLE
23 01CE  07 0F38  2390     IN R1;
23 01CE  07 0F38  2391  BEGIN
23 01CE  07 0F38  2392     R9 := @B9(4);    COMMENT STEP OUTPUT POINTER;
23 01D2  07 0F38  2393     STC(R0,OP(R9));
23 01D6  07 0F38  2394     R1:=V1(R8);
23 01DA  07 0F38  2395     POINTER(R9):=R1;
23 01DE  07 0F38  2396     V5(R8):=R9;       COMMENT SAVE OUTPUT POINTER;
23 01E2  07 0F38  2397  END;
23 01E4  07 0F38  2398
23 01E4  07 0F38  2399  PROCEDURE  OUTOP (R10);
23 01E4  07 0F38  2400  COMMENT  OUTOP PUTS OPERATOR AND POINTER TO FIRST ARGUMENT IN OUTPUT
23 01E4  07 0F38  2401     STRING.  TAKES  OPERATOR FROM R0;
23 01E4  07 0F38  2402  BEGIN
23 01E4  07 0F38  2403     R9 := @B9(4);    COMMENT STEP OUTPUT POINTER;
```

```
23 01E8  07 0F38  2404      IF R9 >= PLIMIT THEN
23 01FC  07 0F38  2405      BEGIN  R0 := @TABLEERROR;  ERROR;  GOTO EXIT;
23 0204  07 0F38  2406      END;
23 0204  07 0F38  2407      STC(R0,OP(R9));
23 0208  07 0F38  2408      R1 := V5(R8) - OUTBASE;  POINTER(R9) := R1;
23 0214  07 0F38  2409      V5(R8):=R9;        COMMENT SAVE OUTPUT POINTER;
23 0218  07 0F38  2410    END;
23 021A  07 0F38  2411
23 021A  07 0F38  2412    PROCEDURE BOOLVALUE(R10);
23 021A  07 0F38  2413    COMMENT  VALUE IN R0, POINTER IN R1;
23 021A  07 0F38  2414      BEGIN  R9 := @B9(4);  STC(R0,OP(R9));  COMMENT OUTPUT OPERATOR;
23 0222  07 0F38  2415             POINTER(R9) := R1;
23 0226  07 0F38  2416             R0 := 6; V2(R8):=R0;   COMMENT SET VALUE STACK;
23 022E  07 0F38  2417             R0 := 0;  V34(R8) := R0;
23 0236  07 0F38  2418             V5(R8):=R9;
23 023A  07 0F38  2419      END;
23 023C  07 0F38  2420
23 023C  07 0F38  2421    PROCEDURE REFBIND(R10);
23 023C  07 0F38  2422    BEGIN  INTEGER  RASAVE,R4SAVE,R5SAVE;
23 023C  07 0F44  2423      SHORT INTEGER B5 SYN #5000;
23 023C  07 0F44  2424      R3 := SIMTYPEINFO(R1);  RASAVE := R10;
23 0244  07 0F44  2425      IF R3 = 0 THEN
23 024A  07 0F44  2426      BEGIN  R0 := @REFRECERROR;  ERROR;
23 025A  07 0F44  2427      END  ELSE
23 025A  07 0F44  2428      BEGIN   STM(R4,R5,R4SAVE);  F01 := V(R8);
23 0266  07 0F44  2429        R4 := 0;  R5 := REFRECBASE;  R5 := @REFRECLIST(R3);  R3 := B5;
23 0276  07 0F44  2430        WHILE R3 ¬= 0 DO
23 027C  07 0F44  2431        BEGIN  V1(R7) := R3;  DECLARETEST;  R3 := V1(R8);
23 0288  07 0F44  2432          IC(R0,RCCLNUMBER(R3));
23 028C  07 0F44  2433          R3 := R0 SHLL 1;  R3 := BITTABLE(R3);  R4 := R4 OR R3;
23 0298  07 0F44  2434          R5 := R5 + 2;  R3 := B5;
23 02A0  07 0F44  2435        END;
23 02A4  07 0F44  2436        SIMTYPEINFO(R1) := R4;  V(R8) := F01;  LM(R4,R5,R4SAVE);
23 02B0  07 0F44  2437      END;
23 02B0  07 0F44  2438      R10 := RASAVE;
23 02B4  07 0F44  2439    END;
23 02B6  07 0F44  2440
23 02B6  07 0F44  2441    PROCEDURE ALIGNSIMPVAR(R10);
23 02B6  07 0F44  2442    COMMENT  ALIGNSIMPVAR ADJUSTS R4 TO THE APPROPRIATE HALF-WORD,WORD,OR
23 02B6  07 0F44  2443             DOUBLE-WORD  BOUNDARY - AS DETERMINED BY THE SIMPLE TYPE IN
23 02B6  07 0F44  2444             R3.       R3  IS NOT DESTROYED  ;
23 02B6  07 0F44  2445      IF R3 = 3 OR R3 = 5 THEN R4 := R4 + 7 SHRL 3 SHLL 3 ELSE
23 02D2  07 0F44  2446      IF R3 ¬= 7 AND R3 ¬= 6 THEN R4 := R4 + 3 SHRL 2 SHLL 2;
23 02F4  07 0F44  2447
23 02F4  07 0F44  2448    PROCEDURE MAXREG(R10);
23 02F4  07 0F44  2449    COMMENT  SETS REGISTER COUNTS TO MAXIMUM OF V(R8) AND V(8)(R8);
23 02F4  07 0F44  2450      BEGIN
23 02F4  07 0F44  2451        IC(R0,V3(R8+8));  R1 := 0;  IC(R1,V3(R8));
23 0300  07 0F44  2452        IF R1 < R0 THEN STC(R0,V3(R8));
23 030A  07 0F44  2453        IC(R0,V4(R8+8));  IC(R1,V4(R8));
23 0312  07 0F44  2454        IF R1 < R0 THEN STC(R0,V4(R8));
23 031C  07 0F44  2455      END;
23 031E  07 0F44  2456
23 031E  07 0F44  2457    PROCEDURE CHECKSPACE(R10);
23 031E  07 0F44  2458    COMMENT  INPUT AS MOVETREE. CHECKS FOR SPACE, COMPACTS IF POSSIBLE;
23 031E  07 0F44  2459      BEGIN  LOGICAL SAVERA;  SAVERA := R10;
23 0322  07 0F48  2460        R10 := @B2(R1+3) - COMMONLIMIT; IF > THEN
23 032E  07 0F48  2461        BEGIN  COMMENT  R10 = ADDITIONAL SPACE REQUIRED;
```

```
23 032E  07 0F48  2462        R10 := NEG R10 + INPOINT - INPOINTSAVE - LITPNT; IF < THEN
23 0340  07 0F48  2463        BEGIN  COMMENT SPACE NOT AVAILABLE;
23 0340  07 0F48  2464          SET(COMMONFLAG);  R0 := @TABLEERROR; ERROR;
23 0354  07 0F48  2465        END ELSE
23 0354  07 0F48  2466        BEGIN  ARRAY 4 LOGICAL SAVE03;  STM(R0,R3,SAVE03);
23 035C  07 0F58  2467          R3 := INPOINTSAVE + LITPNT;  R1 := INPOINT;
23 0368  07 0F58  2468          R0 := R1 - R3 AND #3;  R3 := R3 + R0;  COMMENT ALIGN DELTA;
23 0372  07 0F58  2469          INPOINT := R3;  COMMENT R1, R3 = SOURCE, DEST;
23 0376  07 0F58  2470          R0 := R1 - R3; R10 := R2 - R1; COMMENT R0,R10 = DELTA,BYTE CNT;
23 037E  07 0F58  2471          WHILE R10 > 256 DO
23 0386  07 0F58  2472          BEGIN  MVC(255,B3,B1);  R10 := R10 - 256;
23 0390  07 0F58  2473            R1 := @B1(256);  R3 := @B3(256);
23 0398  07 0F58  2474          END;
23 039C  07 0F58  2475          IF R10 > 0 THEN
23 03A2  07 0F58  2476          BEGIN  R10 := R10 - 1;  EX(R10,MOVE31);
23 03AA  07 0F58  2477          END;
23 03AA  07 0F58  2478          R2 := R2 - R0;  R1 := TREEBASE - R0; TREEBASE := R1;
23 03B6  07 0F58  2479          R1 := @TREELINK;  COMMENT ADJUST TREE POINTERS;
23 03EA  07 0F58  2480          WHILE R1 > 0 DO
23 03C0  07 0F58  2481          BEGIN  R3 := MEM(R1) - R0; MEM(R1) := R3;  R1 := R3;
23 03CC  07 0F58  2482          END;
23 03D0  07 0F58  2483          LM(R0,R1,SAVE03);  R3 := SAVE03(12);
23 03D8  07 0F58  2484        END;
23 03D8  07 0F58  2485      END;
23 03D8  07 0F58  2486      R10 := SAVERA;
23 03DC  07 0F58  2487    END;
23 03DE  07 0F58  2488
23 03DE  07 0F58  2489  PROCEDURE MOVETREE(R10);
23 03DE  07 0F58  2490  COMMENT  BYTE COUNT IN R1, SOURCE IN R4, DESTINATION IN R2;
23 03DE  07 0F58  2491    BEGIN  INTEGER R1SAVE, R4SAVE, RASAVE;  R4SAVE := R4;
23 03E2  07 0F64  2492      RASAVE := R10;  CHECKSPACE;  R10 := RASAVE;
23 03EE  07 0F64  2493      IF NOGO THEN R2 := @B2(R1+4) ELSE
23 03FA  07 0F64  2494      BEGIN  R1SAVE := R1;  R4 := R4SAVE;
23 0406  07 0F64  2495        TREE(R2) := R1;   COMMENT  INSERT LENGTH;
23 040A  07 0F64  2496        WHILE R1 > 256 DO
23 0412  07 0F64  2497        BEGIN  MVC(255,B2(4),B4); R2 := @B2(256); R4 := @B4(256);
23 0420  07 0F64  2498          R1 := R1 - 256;
23 0424  07 0F64  2499        END;
23 0428  07 0F64  2500        IF R1 ¬= 0 THEN
23 042E  07 0F64  2501        BEGIN  R1 := R1 - 1; EX(R1,TREEMOVE); R2 := @B2(R1+1);
23 043A  07 0F64  2502        END;
23 043A  07 0F64  2503        R2 := @B2(4);   COMMENT  SET TO NEXT FREE WORD;
23 043E  07 0F64  2504        R1 := R1SAVE;
23 0442  07 0F64  2505      END;
23 0442  07 0F64  2506      R4 := R4SAVE;  WHILE R1 > 256 DO
23 044E  07 0F64  2507      BEGIN MVI(0,B4); MVC(254,B4(1),B4); R4 := @B4(256); R1 := R1-256;
23 0460  07 0F64  2508      END;
23 0464  07 0F64  2509      IF R1 ¬= 0 THEN
23 046A  07 0F64  2510      BEGIN  R1 := R1-1;  EX(R1,TREEXOR);
23 0472  07 0F64  2511      END;
23 0472  07 0F64  2512    END;
23 0474  07 0F64  2513
23 0474  07 0F64  2514   SEGMENT PROCEDURE   EXECUTE1(R6);
23 0474  07 0F64  2515  BEGIN
24 0000  07 0F64  2516
24 0000  07 0F64  2517  PROCEDURE    BNDPR(R10);
24 0000  07 0F64  2518  COMMENT  BNDPR CHECKS THAT BOTH BOUNDS ARE INTEGER, INCREASES THE
24 0000  07 0F64  2519           DIMENSION COUNT, AND OUTPUTS COLON;
```

```
24 0000  07 0F64  2520    BEGIN  INTEGER  RASAVE;
24 0004  07 0F68  2521      RASAVE := R10;
24 0008  07 0F68  2522      IC(R0,V22(R8+8));          COMMENT GET SIMPLE TYPE OF LOWER BOUND;
24 000C  07 0F68  2523      IC(R1,V22(R8+24));         COMMENT  GET SIMPLE TYPE OF UPPER BOUND;
24 0010  07 0F68  2524      TYPEINTEGER;
24 001C  07 0F68  2525      R9 := @B9(4);               COMMENT  STEP OUTPUT POINTER;
24 0020  07 0F68  2526       R0 := @COLON;
24 0024  07 0F68  2527      STC(R0,OP(R9));  R1 := V5(R8+8) - OUTBASE;  POINTER(R9) := R1;
24 0034  07 0F68  2528      IC(R0,V4(R8));  R0 := R0 + 1;  STC(R0,V4(R8));
24 0040  07 0F68  2529      R10 := RASAVE;
24 0044  07 0F68  2530    END;
24 0046  07 0F68  2531
24 0046  07 0F68  2532  PROCEDURE  FPARLOCATE(R5);
24 0046  07 0F68  2533  COMMENT   DRELAD IS INITIALIZED, SPACE IS ALLOTED FOR PD, AND RELATIVE
24 0046  07 0F68  2534     ADDRESSES ARE ASSIGNED TO FORMAL PARAMETERS;
24 0046  07 0F68  2535    BEGIN  INTEGER  R5SAVE,TEMP;
24 0046  07 0F70  2536      R5SAVE := R5;
24 004A  07 0F70  2537      R1 := VARORIGIN+4;  VARORIGIN := R1;
24 0056  07 0F70  2538      R1 := HN-1;  HN := R1;          COMMENT  STEP  HN;
24 0062  07 0F70  2539      IF R1 < = 5 THEN
24 006A  07 0F70  2540      BEGIN  R0 := @HIERARCHYERROR;  ERROR;
24 007A  07 0F70  2541      END;
24 007A  07 0F70  2542      R2 := V1(R8);   STC(R1,HIERARCHY(R2));
24 0082  07 0F70  2543      R3 := SNC SHLA 2;
24 008A  07 0F70  2544      R1 := CTPNT;  CTPOINTER(R3) := R1;
24 0092  07 0F70  2545      R1 := R1 + 3 SHRA 2 SHLA 2;  CTORG := R1;
24 00A2  07 0F70  2546      R4 := @CONSTANTTABLE(R1);   MVC(11,B4,CONSTANTTABLE);
24 00AC  07 0F70  2547      R1 := R1 + 12;     CTPNT := R1;
24 00B4  07 0F70  2548      R1 := LITPNT;  LITPOINTER(R3) := R1;  R1 := R1 + 7 SHRA 3 SHLA 3;
24 00C8  07 0F70  2549      LITORG := R1;  R3 := LITBASE;  R4 := @LITERALTABLE(R1);
24 00D4  07 0F70  2550      MVC(3,B4,LITERALTABLE);  R1 := R1 + 4;  LITPNT := R1;
24 00E2  07 0F70  2551
24 00E2  07 0F70  2552      R3 := SN+1;  SN := R3;  R4 := SNC;  SNC := R3;
24 00F6  07 0F70  2553      STC(R3,PROGSEG(R2));  R3 := R3 SHLA 2;  STATLINK(R3) := R4;
24 0102  07 0F70  2554      R4 := OUTBASE;  OUTPOINT(R3) := R4;  R4 := R9 - 8;  OUTBASE := R4;
24 0114  07 0F70  2555      R4 := DRELAD;  V34(R8) := R4;
24 011C  07 0F70  2556      R4 := LABELADDR;  V2(R8) := R4;  R4 := 24;  LABELADDR := R4;
24 012C  07 0F70  2557      R2 := TYPEINFO(R2) SHLA 2;    COMMENT  GET BLOCKNUMBER OF FPARS;
24 0134  07 0F70  2558      IF R2 ¬= 0 THEN
24 013A  07 0F70  2559      BEGIN
24 013A  07 0F70  2560        R1 := NPOINT(R2); R3 := BLENGTH(R2);  R2 := R3 + R1;  TEMP := R2;
24 014A  07 0F70  2561        R2 := 0;  R3 := R3/12 SHLL 3;  R4 := VARORIGIN + R3;
24 015C  07 0F70  2562        R3 := R3 SHRL 1 + LABELADDR;  LABELADDR := R3;
24 0168  07 0F70  2563        R2 := VARORIGIN;  R3 := 0;  R5 := HN;
24 0174  07 0F70  2564        WHILE R1 < TEMP DO
24 017C  07 0F70  2565         BEGIN
24 017C  07 0F70  2566          IDLOC1(R1) := R5;       COMMENT  INSERT HN;
24 0180  07 0F70  2567          IC(R0,VR(R1));         COMMENT  GET VALUE INFO;
24 0184  07 0F70  2568          IC(R3,TYPE(R1));
24 0188  07 0F70  2569          IF R0 ¬= 0 THEN
24 018E  07 0F70  2570            BEGIN  R3 := 0;  STC(R3,TYPE(R1));
24 0196  07 0F70  2571            END ELSE IF R3 ¬= #12 THEN IDLOC2(R1) := R2;
24 01A6  07 0F70  2572          R1 := R1 + 12;  R2 := R2 + 8;
24 01AE  07 0F70  2573         END;
24 01B2  07 0F70  2574      END  ELSE  R4 := VARORIGIN;
24 01BA  07 0F70  2575      DRELAD := R4;  R5 := R5SAVE;
24 01C2  07 0F70  2576    END;
24 01C4  07 0F70  2577
```

```
24 01C4  07 0F70  2578  PROCEDURE PROCEDCLOSE (R5);
24 01C4  07 0F70  2579        BEGIN
24 01C4  07 0F70  2580          R1 := BNC;                  COMMENT  RESTORE BNC;
24 01C8  07 0F70  2581          R1 := BLOCKLIST2(R1);   BNC := R1;
24 01D0  07 0F70  2582          R2 := HN+1;  HN := R2;   COMMENT  RESTORE  HN;
24 01DC  07 0F70  2583          R2 := VARORIGIN-4;  VARORIGIN := R2;
24 01E8  07 0F70  2584          R2 := V5(R8) - 4;  R2 := POINTER(R2);
24 01F4  07 0F70  2585          R1 := DRELAD;
24 01F8  07 0F70  2586          IF R1 > #FFF THEN
24 0200  07 0F70  2587          BEGIN  R0 := @DATAERROR; ERROR;
24 0210  07 0F70  2588          END;
24 0210  07 0F70  2589          IDLOC1(R2) := R1;     COMMENT  INSERT LOCAL STACK ORIGIN;
24 0214  07 0F70  2590          R1 := V34(R8);  DRELAD := R1;
24 021C  07 0F70  2591          R4 := V5(R8) - 4;  R0 := @PCL;  OUTOP;
24 0234  07 0F70  2592          R1 := SNC SHLA 2;  R2 := STATLINK(R1);  SNC := R2;
24 0244  07 0F70  2593          R2 := OUTPOINT(R1);  OUTBASE := R2;
24 024C  07 0F70  2594          R2 := TREEORG;  R1 := R9 - R4 + 4;
24 0258  07 0F70  2595          R9 := TREELENGTH + R1 + 4;  TREELENGTH := R9;
24 0266  07 0F70  2596          R9 := R4;  R3 := 0;  IC(R3,OUTFLAG); IF R3 > 5 THEN WRITETREE;
24 0282  07 0F70  2597          R2 := POINTER(R4);  IC(R0,PROGSEG(R2));
24 028A  07 0F70  2598          R3 := R1;  R1 := 8; R2 := TREEORG; CHECKSPACE;  R1 := R3;
24 02A2  07 0F70  2599          IF ¬NOGO THEN
24 02AA  07 0F70  2600          BEGIN  ARRAY 2 INTEGER SAVE45;  STM(R4,R5,SAVE45);
24 02AE  07 0F78  2601            R5 := TREELINK; B2(0) := R5; TREELINK := R2;
24 02BA  07 0F78  2602            MVC(7,B2(4),"        ");
24 02C0  07 0F78  2603            R4 := POINTER(R4); COMMENT PROCDC;  R4 := IDNO(R4) SHLL 2;
24 02CC  07 0F78  2604            R3 := IDDIRBASE;  R5 := IDLENGTH(R4); IF R5 > 7 THEN R5 := 7;
24 02DC  07 0F78  2605            R4 := IDPOINT(R4); R3 := IDLISTBASE;
24 02E8  07 0F78  2606            R4 := @IDLIST(R4); EX(R5,TREEMOVE);  LM(R4,R5,SAVE45);
24 02F4  07 0F78  2607          END;
24 02F4  07 0F78  2608          R2 := @B2(12); R4 := @P(R4);  MOVETREE;  TREEORG := R2;
24 030C  07 0F78  2609          POINTER(R9) := R0;  R0 := @SEG;  STC(R0,OP(R9));
24 0318  07 0F78  2610          R2 := TREEORG;  R1 := CTPNT - CTORG + 4;  R4 := CTORG;
24 032C  07 0F78  2611          R4 := @CONSTANTTABLE(R4-4);  R3 := B4;
24 0334  07 0F78  2612          MVC(3,B4,LABELADDR);  LABELADDR := R4;  MOVETREE;
24 034A  07 0F78  2613          R4 := LABELADDR;    B4 := R3;
24 0352  07 0F78  2614          R1 := V2(R8);  LABELADDR := R1;
24 035A  07 0F78  2615          R1 := LITPNT - LITORG;  R4 := LITORG;  R3 := LITBASE;
24 036A  07 0F78  2616          R4 := @LITERALTABLE(R4);  MOVETREE;
24 037A  07 0F78  2617          R2 := R2 + 3 SHRA 2 SHLA 2;  TREEORG := R2;
24 038A  07 0F78  2618          R1 := SNC SHLA 2;  R2 := CTPOINTER(R1);  CTPNT := R2;
24 039A  07 0F78  2619          R2 := LITPOINTER(R1);  LITPNT := R2;
24 03A2  07 0F78  2620          R1 := STATLINK(R1) SHLA 2;
24 03AA  07 0F78  2621          R2 := CTPOINTER(R1) + 3 SHRA 2 SHLA 2;  CTORG := R2;
24 03BE  07 0F78  2622          R2 := LITPOINTER(R1) + 7 SHRA 3 SHLA 3;  LITORG := R2;
24 03D2  07 0F78  2623        END;
24 03D4  07 0F78  2624
24 03D4  07 0F78  2625  PROCEDURE VALUEP(R5);
24 03D4  07 0F78  2626    BEGIN
24 03D4  07 0F78  2627      DECLARETEST;  R1 := V1(R8);  R3 := TYPES(R1);
24 03E8  07 0F78  2628      IF R3 ¬= 9 THEN
24 03F0  07 0F78  2629      BEGIN  R4 := DRELAD;  ALIGNSIMPVAR;  IDLOC2(R1) := R4;
24 0404  07 0F78  2630        IF R3 ¬= 7 THEN IC(R3,LENGTHTABLE(R3) )ELSE
24 0410  07 0F78  2631        R3 := SIMTYPEINFO(R1) + 1;
24 041C  07 0F78  2632        R4 := R4 + R3;  DRELAD := R4;
24 0422  07 0F78  2633      END;
24 0422  07 0F78  2634      R0 := 1;  V34(R8) := R0;
24 042A  07 0F78  2635    END;
```

```
24 042C  07 0F78          2636
24 042C  07 0F78          2637  PROCEDURE RELADDRESS(R5);
24 042C  07 0F78          2638    IF R1 ¬= 0 THEN     COMMENT ROUTINE EXECUTED ONLY IF ID DEFINED;
24 0432  07 0F78          2639    BEGIN
24 0432  07 0F78          2640      R4 := HN;  IDLOC1(R1) := R4;  R4 := DRELAD;  R3 := TYPES(R1);
24 0442  07 0F78          2641      IF R3 ¬= 9 THEN
24 044A  07 0F78          2642      BEGIN
24 044A  07 0F78          2643      R0 := R3;  ALIGNSIMPVAR;  IDLOC2(R1) := R4;
24 045C  07 0F78          2644      IF R3 ¬= 7 THEN IC(R3,LENGTHTABLE(R3) )ELSE
24 0468  07 0F78          2645         R3 := SIMTYPEINFO(R1) + 1;
24 0474  07 0F78          2646      R3 := R3 + R4; DRELAD := R3;
24 047A  07 0F78          2647      END;
24 047A  07 0F78          2648    END;
24 047C  07 0F78          2649
24 047C  07 0F78          2650  PROCEDURE FIELDDC(R5);
24 047C  07 0F78          2651    BEGIN
24 047C  07 0F78          2652      R4 := V2(R8);
24 0480  07 0F78          2653      DECLARETEST;  R1 := V1(R8);  R3 := 0;  IC(R3,SIMPLETYPE(R1));
24 0498  07 0F78          2654      IF R3 = 3 OR R3 = 5 THEN
24 04A8  07 0F78          2655      BEGIN  R4 := R4 + 7 AND #FF8; IDLOC2(R1) := R4;
24 04B4  07 0F78          2656         IC(R3,LENGTHTABLE(R3));  R4 := R4 + R3;
24 04BA  07 0F78          2657      END;
24 04BA  07 0F78          2658      V2(R8) := R4;
24 04BE  07 0F78          2659    END;
24 04C0  07 0F78          2660
24 04C0  07 0F78          2661       RULENUMBER := R5;
24 04C4  07 0F78          2662      CASE R5 OF
24 04C4  07 0F78          2663      BEGIN
24 04C4  07 0F78          2664                                                                    1
24 04C4  07 0F78          2665         COMMENT   <T VAR ID> ::= <ID>,ALSO SELECTION OF APPROPRIATE
24 04C4  07 0F78          2666           ID RULE;
24 04C4  07 0F78          2667         BEGIN
24 04CC  07 0F78          2668           DECLARETEST;  IC(R0,V21(R8));
24 04DC  07 0F78          2669           IF R0 > #F THEN R2 := #404 ELSE R2 := 0;   R0 := R0 AND #F;
24 04F4  07 0F78          2670           V34(R8) := R2;        COMMENT  SET REGISTER COUNTS;
24 04F8  07 0F78          2671           IF      R0 ¬= 0 THEN
24 04FE  07 0F78          2672              BEGIN
24 04FE  07 0F78          2673              R0 := R0 - 1 SHLA 2;
24 0506  07 0F78          2674              R3 := R3SAVE;
24 050A  07 0F78          2675              R3 := R3 + R0;      R3SAVE := R3;
24 0510  07 0F78          2676              SET(FLAG);   GOTO  DONE;
24 0518  07 0F78          2677              END;
24 0518  07 0F78          2678           R0 := @ID;  OUTID;        COMMENT "ID"AND POINTER IN OUTPUT;
24 0528  07 0F78          2679           R1 := SIMTYPEINFO(R1);   V1(R8) := R1;
24 0530  07 0F78          2680  DONE:  END;
24 0530  07 0F78          2681                                                                    2
24 0530  07 0F78          2682         COMMENT    <LABEL ID> ::= <ID>;
24 0530  07 0F78          2683         BEGIN
24 0534  07 0F78          2684           R0 := @LABELID;OUTID; COMMENT "LABELID" AND POINTER IN OUTPUT;
24 0544  07 0F78          2685         END;
24 0544  07 0F78          2686                                                                    3
24 0544  07 0F78          2687         COMMENT    <T ARRAY ID> ::= <ID>;
24 0544  07 0F78          2688         BEGIN
24 0548  07 0F78          2689           R0 := @ARRAYID;OUTID; COMMENT "ARRAYID" AND POINTER IN OUTPUT;
24 0558  07 0F78          2690         END;
24 0558  07 0F78          2691                                                                    4
24 0558  07 0F78          2692         COMMENT    <T FUNC ID> ::= <ID>;
24 0558  07 0F78          2693         BEGIN
```

```
24 055C  07 0F78  2694          RO := @FUNCID;OUTID; COMMENT  "FUNCID" AND POINTER IN OUTPUT;
24 056C  07 0F78  2695        END;
24 056C  07 0F78  2696                                                                        5
24 056C  07 0F78  2697        COMMENT    <RC CL ID> ::= <ID>;
24 056C  07 0F78  2698        BEGIN
24 0570  07 0F78  2699          RO := @RCCLID; OUTID;  COMMENT "RCCLID" AND POINTER IN OUTPUT;
24 0580  07 0F78  2700          IC(RO,RCCLNUMBER(R1));  STC(RO,V22(R8));
24 0588  07 0F78  2701        END;
24 0588  07 0F78  2702                                                                        6
24 0588  07 0F78  2703        COMMENT    <T FLD ID> ::= <ID>;
24 0588  07 0F78  2704        BEGIN
24 058C  07 0F78  2705          RO := @FIELDID;OUTID; COMMENT "FIELDID" AND POINTER IN OUTPUT;
24 059C  07 0F78  2706          IC(RO,RCCLNUMBER(R1));  STC(RO,V21(R8));
24 05A4  07 0F78  2707          R1 := SIMTYPEINFO(R1);   V1(R8) := R1;
24 05AC  07 0F78  2708          R1 := 1;  STC(R1,V3(R8));  COMMENT SET REGISTER COUNT;
24 05B4  07 0F78  2709        END;
24 05B4  07 0F78  2710                                                                        7
24 05B4  07 0F78  2711        COMMENT    <CON ID> ::= <ID>;
24 05B4  07 0F78  2712        BEGIN
24 05B8  07 0F78  2713          RO := @CONID; OUTID;  COMMENT "CONID" AND POINTER IN OUTPUT;
24 05C8  07 0F78  2714          RO := 0;  STC(RO,V21(R8));  COMMENT  SET TYPE TO 0;
24 05D0  07 0F78  2715        END;
24 05D0  07 0F78  2716                                                                        8
24 05D0  07 0F78  2717        COMMENT    <ST FUNC ID>   ::=  <ID>;
24 05D0  07 0F78  2718        BEGIN
24 05D4  07 0F78  2719          RO := @STFUNCID;  OUTID;
24 05E4  07 0F78  2720          COMMENT  **********  SET REGISTER COUNTS;
24 05E4  07 0F78  2721        END;
24 05E4  07 0F78  2722                                                                        9
24 05E4  07 0F78  2723        COMMENT    <FILE ID>    ::=  <ID>;
24 05E4  07 0F78  2724        NULL;
24 05E8  07 0F78  2725                                                                       10
24 05E8  07 0F78  2726        COMMENT    <ST PROC ID>   ::=   <ID>;
24 05E8  07 0F78  2727        BEGIN
24 05EC  07 0F78  2728          RO := @STPROCID;  OUTID;
24 05FC  07 0F78  2729        END;
24 05FC  07 0F78  2730                                                                       11
24 05FC  07 0F78  2731        COMMENT    <SI VAR DC>  ::= <SI VAR DC*>;
24 05FC  07 0F78  2732        NULL;
24 0600  07 0F78  2733                                                                       12
24 0600  07 0F78  2734        COMMENT    <SI VAR DC*> ::= <SI TYPE>  <ID>;
24 0600  07 0F78  2735        BEGIN
24 0604  07 0F78  2736          DECLARETEST;  R1 :=V1(R8);  RELADDRESS;
24 0618  07 0F78  2737        END;
24 0618  07 0F78  2738                                                                       13
24 0618  07 0F78  2739        COMMENT    <SI VAR DC*> ::= <SI VAR DC*> ,, <ID> ;
24 0618  07 0F78  2740        BEGIN
24 061C  07 0F78  2741          R1 := V1(R8)+12;  V1(R8) := R1;  RELADDRESS;
24 062C  07 0F78  2742        END;
24 062C  07 0F78  2743                                                                       14
24 062C  07 0F78  2744        COMMENT    <SI TYPE>   ::=   <REF TYPE>    );
24 062C  07 0F78  2745        BEGIN
24 0630  07 0F78  2746          RO := 9;  V2(R8) := RO;   COMMENT SET SI TYPE TO "REFERENCE";
24 0638  07 0F78  2747        END;
24 0638  07 0F78  2748                                                                       15
24 0638  07 0F78  2749        COMMENT    <REF TYPE>    ::=  REFERENCE   <ID>;
24 0638  07 0F78  2750        BEGIN
24 063C  07 0F78  2751          DECLARETEST;  IC(RO,V21(R8));
```

```
24 064C  07 0F78  2752          IF RO ¬= 4 THEN
24 0654  07 0F78  2753          BEGIN  RO := @RCCLERROR;  ERROR;
24 0664  07 0F78  2754          END;
24 0664  07 0F78  2755        END;
24 0664  07 0F78  2756                                                          16
24 0664  07 0F78  2757        COMMENT    <REF TYPE>   ::=   <REF TYPE>   ,,   <ID>;
24 0664  07 0F78  2758        BEGIN
24 0668  07 0F78  2759          DECLARETEST;  IC(RO,V21(R8));
24 0678  07 0F78  2760          IF RO ¬= 4 THEN
24 0680  07 0F78  2761          BEGIN  RO := @RCCLERROR;  ERROR;
24 0690  07 0F78  2762          END;
24 069C  07 0F78  2763        END;
24 0690  07 0F78  2764                                                          17
24 0690  07 0F78  2765        COMMENT    <FILE DC> ::=  <FILE HD*>  )   ;
24 0690  07 0F78  2766        NULL;
24 0694  07 0F78  2767                                                          18
24 0694  07 0F78  2768        COMMENT    <FILE DC> ::=  <FILE HD**>   <T EXP>   )  ;
24 0694  07 0F78  2769        NULL;
24 0698  07 0F78  2770                                                          19
24 0698  07 0F78  2771        COMMENT    <FILE DC> ::=  <FILE HD*->   <T EXP>   )  ;
24 0698  07 0F78  2772        NULL;
24 069C  07 0F78  2773                                                          20
24 069C  07 0F78  2774        COMMENT    <FILE HD**> ::= <FILE HD*->  <T EXP>   ,  ;
24 069C  07 0F78  2775        NULL;
24 06A0  07 0F78  2776                                                          21
24 06AC  07 0F78  2777        COMMENT    <FILE HD*-> ::= <FILE HD*> ,   ;
24 06A0  07 0F78  2778        NULL;
24 06A4  07 0F78  2779                                                          22
24 06A4  07 0F78  2780        COMMENT    <FILE HD*>   ::=  <FILE DESC>    (   <I/O DEV>;
24 06A4  07 0F78  2781        NULL;
24 06A8  07 0F78  2782                                                          23
24 06A8  07 0F78  2783        COMMENT    <FILE DESC>  ::=  FILE  <ID>;
24 06A8  07 0F78  2784        NULL;  COMMENT  * * * * * * * * * * *;
24 06AC  07 0F78  2785                                                          24
24 06AC  07 0F78  2786        COMMENT    <FILE DESC>   ::=  <FILE DESC>  ,,  <ID>   ;
24 06AC  07 0F78  2787        NULL;  COMMENT  * * * * * * * * * * *;
24 06B0  07 0F78  2788                                                          25
24 06B0  07 0F78  2789        COMMENT    <ARRAY DC> ::= <BND LST HD> <T EXP> :: <T EXP>   ) ;
24 06B0  07 0F78  2790        BEGIN  BYTE FLAG;
24 06B4  07 0F79  2791          BNDPR;  RO := @ARPAREN;  OUTOP;  R2 := V1(R8);  R4 := 0;
24 06D0  07 0F79  2792          IC(RO,SIMPLETYPE(R2));
24 06D4  07 0F79  2793          IF RO ¬= 9 THEN
24 06DC  07 0F79  2794          BEGIN  RESET(FLAG);  R5 := DRELAD + 3 SHRL 2 SHLL 2;
24 06F0  07 0F79  2795          END ELSE SET(FLAG);
24 06F8  07 0F79  2796          R1 := V2(R8);  BNC := R1;   COMMENT  RESTORE BNC;
24 0700  07 0F79  2797          IC(R4,V3(R8));  V2(R8) := R4;  IC(R4,V4(R8));
24 070C  07 0F79  2798          R1 := R4*12S + 8;
24 C716  07 0F79  2799          FOR R3 := 1 STEP 1 UNTIL V2(R8) DO
24 C71A  07 0F79  2800          BEGIN    IC(RO,DIMEN(R2));
24 0722  07 0F79  2801            IF RO = 0 THEN STC(R4,DIMEN(R2) )ELSE
24 C72C  07 0F79  2802            IF RO ¬= R4 THEN
24 0736  07 0F79  2803            BEGIN   RO := @ARRAYERROR; ERROR;
24 0746  07 0F79  2804            END;
24 0746  07 0F79  2805            IF ¬FLAG THEN
24 074E  07 0F79  2806            BEGIN  IDLOC2(R2) := R5;  R5 := R5 + R1;  DRELAD := R5;
24 C758  07 0F79  2807            END ELSE IDLOC2(R2) := R1;
24 C76C  07 0F79  2808            R5 := HN;  IDLOC1(R2) := R5;  R5 := DRELAD; R2 := R2 + 12;
24 0770  07 0F79  2809          END;
```

```
24 077C  07 0F79  2810          END;
24 077C  07 0F79  2811                                                                  26
24 077C  07 0F79  2812          COMMENT    <ARRAY HD>   ::=  <SI TYPE>  ARRAY  <ID>  ;
24 077C  07 0F79  2813          BEGIN
24 0780  07 0F79  2814            R9 := R9 + 4;  R0 := @ARRAYDC;  STC(R0,OP(R9));  CARDOUT;
24 0798  07 0F79  2815            DECLARETEST;  R0 := 1;  STC(R0,V3(R8));
24 07AC  07 0F79  2816            R1 := V1(R8);  POINTER(R9-4) := R1;
24 07B4  07 0F79  2817          END;
24 07B4  07 0F79  2818                                                                  27
24 07B4  07 0F79  2819          COMMENT    <ARRAY HD>   ::=  <ARRAY HD>   ,,  <ID>  ;
24 07B4  07 0F79  2820          BEGIN
24 07B8  07 0F79  2821            IC(R0,V3(R8));  R0 := R0 + 1;  STC(R0,V3(R8));
24 07C4  07 0F79  2822          END;
24 07C4  07 0F79  2823                                                                  28
24 07C4  07 0F79  2824          COMMENT    <BND LST HD> ::= <ARRAY HD>  (  ;
24 07C4  07 0F79  2825          BEGIN
24 07C8  07 0F79  2826            R0 := 0;  STC(R0,V4(R8));  COMMENT  INITIALIZE DIMEN COUNT;
24 07D0  07 0F79  2827            R1 := BNC;  V2(R8) := R1;  R1 := BLOCKLIST2(R1);  BNC := R1;
24 07E0  07 0F79  2828            IC(R0,V3(R8));  STC(R0,CONV(R9-4));
24 07E8  07 0F79  2829          END;
24 07E8  07 0F79  2830                                                                  29
24 07E8  07 0F79  2831          COMMENT  <BND LST HD> ::= <BND LST HD> <T EXP> :: <T EXP>  ,  ;
24 07E8  07 0F79  2832          BEGIN
24 07EC  07 0F79  2833            BNDPR;    R0 := @ARCOMMA;  OUTOP;
24 0800  07 0F79  2834          END;
24 0800  07 0F79  2835                                                                  30
24 0800  07 0F79  2836          COMMENT    <PROC DECL>  ::=  <T PR HEAD> <STATEMENT*>  ;
24 0800  07 0F79  2837          BEGIN
24 0804  07 0F79  2838            R1 := V1(R8);
24 C8C8  07 0F79  2839            IF R1 ¬= 0 THEN
24 080E  07 0F79  2840            BEGIN  IC(R0,SIMPLETYPE(R1));  IF R0 ¬= 0 THEN
24 0818  07 0F79  2841              BEGIN  R0 := @PROCERROR;  ERROR;
24 0828  07 0F79  2842              END;
24 0828  07 0F79  2843              PROCEDCLOSE;
24 082C  07 0F79  2844            END;
24 082C  07 0F79  2845          END;
24 082C  07 0F79  2846                                                                  31
24 082C  07 0F79  2847          COMMENT    <PROC DECL>  ::=  <T PR HEAD> ;
24 082C  07 0F79  2848          BEGIN
24 0830  07 0F79  2849            R0 := @NULLST; R9 := R9 + 4;  STC(R0,OP(R9));
24 083C  07 0F79  2850            R1 := V1(R8);
24 0840  07 0F79  2851            IF R1 ¬= 0 THEN
24 0846  07 0F79  2852            BEGIN  IC(R0,SIMPLETYPE(R1));
24 084A  07 0F79  2853              IF R0 ¬= 0 THEN
24 0850  07 0F79  2854              BEGIN  R0 := @PROCERROR;  ERROR;
24 0860  07 0F79  2855              END;
24 0860  07 0F79  2856              PROCEDCLOSE;
24 0864  07 0F79  2857            END;
24 0864  07 0F79  2858          END;
24 0864  07 0F79  2859                                                                  32
24 0864  07 0F79  2860          COMMENT    <PROC DECL>  ::= <T PROC HEAD>  <T PROC BODY>  ;
24 0864  07 0F79  2861          BEGIN
24 0868  07 0F79  2862            R4 := V1(R8);  IC(R0,SIMPLETYPE(R4));  R4 := SIMTYPEINFO(R4);
24 0874  07 0F79  2863            R2 := @ENDD;  IC(R1,OP(R9));
24 087C  07 0F79  2864            IF R1 = R2 THEN
24 0882  07 0F79  2865            BEGIN  R9 := R9 - 4;  CNVRTASSN;  R9 := R9 + 4;
24 0896  07 0F79  2866            END ELSE  CNVRTASSN;
24 08A6  07 0F79  2867            R1 := V1(R8);
```

```
24 08AA  07 0F79  2868          IF R1 ¬= 0 THEN
24 08BC  07 0F79  2869          BEGIN  R2 := V1(R7);  STC(R2,CONV(R9+4));  PROCEDCLOSE;
24 08BC  07 0F79  2870          END;
24 08BC  07 0F79  2871        END;
24 08BC  07 0F79  2872                                                                  33
24 08BC  07 0F79  2873        COMMENT    <T PROC BODY>  ::= <T EXP> ;
24 08BC  07 0F79  2874        BEGIN    R0 := 0;  V34(R8) := R0;
24 08C8  07 0F79  2875        END;
24 08C8  07 0F79  2876                                                                  34
24 08C8  07 0F79  2877        COMMENT    <T PROC BODY>  ::= <BLOCKBODY>  <T EXP>   END   ;
24 08C8  07 0F79  2878        BEGIN
24 08CC  07 0F79  2879          BLOCKEXIT;
24 08D8  07 0F79  2880          R0 := @COMMA;  OUTOP;  R0 := @ENDD;  OUTID;
24 08F8  07 0F79  2881          R1 := V12(R8+8);   V12(R8) := R1;
24 0900  07 0F79  2882        END;
24 0900  07 0F79  2883                                                                  35
24 0900  07 0F79  2884        COMMENT    <T PR HEAD>  ::=  <T PR HEAD*>  .,      ;
24 0900  07 0F79  2885        NULL;
24 0904  07 0F79  2886                                                                  36
24 0904  07 0F79  2887        COMMENT    <T PR HEAD*> ::= <PROCEDURE>  ;
24 0904  07 0F79  2888        NULL;
24 0908  07 0F79  2889                                                                  37
24 0908  07 0F79  2890        COMMENT    <T PR HEAD*> ::= <PROCEDURE>   <FPAR HEAD>  )   ;
24 0908  07 0F79  2891        BEGIN
24 090C  07 0F79  2892          R1 := V1(R8);  R2 := TYPEINFO(R1) SHLL 2;
24 0918  07 0F79  2893          R1 := NPOINT(R2);  R3 := BLENGTH(R2);  R3 := R3 + R1;
24 0922  07 0F79  2894          R4 := DRELAD;
24 0926  07 0F79  2895          WHILE R1 < R3 DO
24 092C  07 0F79  2896          BEGIN  R2 := TYPES(R1);
24 0930  07 0F79  2897            IF R2 = 9 THEN
24 0938  07 0F79  2898            BEGIN  R4 := R4 + 3 SHRL 2 SHLL 2;  IDLOC2(R1) := R4;
24 0948  07 0F79  2899              R4 := R4 + 4;
24 094C  07 0F79  2900            END;
24 094C  07 0F79  2901            R1 := R1 + 12;
24 0950  07 0F79  2902          END;
24 0954  07 0F79  2903          DRELAD := R4;
24 0958  07 0F79  2904        END;
24 0958  07 0F79  2905                                                                  38
24 0958  07 0F79  2906        COMMENT    <PROCEDURE>  ::=  PROCEDURE  <ID>   ;
24 0958  07 0F79  2907        BEGIN
24 095C  07 0F79  2908          DECLARETEST;   R0 := @PROCDC;  OUTID;
24 0978  07 0F79  2909          CARDOUT;
24 0984  07 0F79  2910          IF R1 ¬= 0 THEN
24 098A  07 0F79  2911          BEGIN  BLOCKSTEP;  FPARLOCATE;
24 099A  07 0F79  2912          END;
24 099A  07 0F79  2913        END;
24 099A  07 0F79  2914                                                                  39
24 099A  07 0F79  2915        COMMENT    <PROCEDURE>  ::=  <SI TYPE>  PROCEDURE  <ID> ;
24 099A  07 0F79  2916        BEGIN
24 099E  07 0F79  2917          DECLARETEST;   R0 := @PROCDC;  OUTID;
24 09BA  07 0F79  2918          CARDOUT;
24 09C6  07 0F79  2919          IF R1 ¬= 0 THEN
24 09CC  07 0F79  2920          BEGIN  BLOCKSTEP;  FPARLOCATE;
24 09DC  07 0F79  2921          END;
24 09DC  07 0F79  2922        END;
24 09DC  07 0F79  2923                                                                  40
24 09DC  07 0F79  2924        COMMENT    <FPAR HEAD>  ::=  <FPAR HEAD*>  ;
24 09DC  07 0F79  2925        NULL;
```

```
24 09E0  07 0F79  2926                                                          41
24 09E0  07 0F79  2927        COMMENT    <FPAR HEAD>   ::=   <FBND LIST>  ;
24 09E0  07 0F79  2928        NULL;
24 09E4  07 0F79  2929                                                          42
24 09E4  07 0F79  2930        COMMENT    <FPAR HEAD*> ::=  (   <SI TYPE>   <ID>   ;
24 09E4  07 0F79  2931        BEGIN
24 09E8  07 0F79  2932          R0 := 0;  V34(R8) := R0;
24 09F0  07 0F79  2933        END;
24 09F0  07 0F79  2934                                                          43
24 09F0  07 0F79  2935        COMMENT    <FPAR HEAD*> ::=  (   <SI TYPE>   VALUE  <ID> ;
24 09F0  07 0F79  2936        VALUEP;
24 09F8  07 0F79  2937                                                          44
24 09F8  07 0F79  2938        COMMENT    <FPAR HEAD*> ::=  (   <SI TYPE>   RESULT    <ID>  ;
24 09F8  07 0F79  2939        VALUEP;
24 0A00  07 0F79  2940                                                          45
24 0A00  07 0F79  2941        COMMENT    <FPAR HEAD*> ::=  (   <SI TYPE>  VALUE  RESULT <ID> ;
24 0A00  07 0F79  2942        VALUEP;
24 0A08  07 0F79  2943                                                          46
24 0A08  07 0F79  2944        COMMENT    <FPAR HEAD*> ::=  (   <SI TYPE>   PROCEDURE  <ID> ;
24 0A08  07 0F79  2945        BEGIN
24 0A0C  07 0F79  2946          R0 := 0; V34(R8) := R0;
24 0A14  07 0F79  2947        END;
24 0A14  07 0F79  2948                                                          47
24 0A14  07 0F79  2949        COMMENT    <FPAR HEAD*> ::=  (   PROCEDURE  <ID>  ;
24 0A14  07 0F79  2950        BEGIN
24 0A18  07 0F79  2951          R0 := 0; V34(R8) := R0;
24 0A20  07 0F79  2952        END;
24 0A20  07 0F79  2953                                                          48
24 0A20  07 0F79  2954        COMMENT    <FPAR HEAD*> ::= <FPAR HEAD-> <SI TYPE>  <ID>  ;
24 0A20  07 0F79  2955        BEGIN
24 0A24  07 0F79  2956          R0 := 0; V34(R8) := R0;
24 0A2C  07 0F79  2957        END;
24 0A2C  07 0F79  2958                                                          49
24 0A2C  07 0F79  2959        COMMENT    <FPAR HEAD*> ::= <FPAR HEAD-> <SI TYPE>   VALUE  <ID>;
24 0A2C  07 0F79  2960        VALUEP;
24 0A34  07 0F79  2961                                                          50
24 0A34  07 0F79  2962        COMMENT    <FPAR HEAD*> ::= <FPAR HEAD-> <SI TYPE>  RESULT  <ID>;
24 0A34  07 0F79  2963        VALUEP;
24 0A3C  07 0F79  2964                                                          51
24 0A3C  07 0F79  2965        COMMENT    <FPAR HEAD*>  ::= <FPAR HEAD-><SI TYPE> PROCEDURE<ID>;
24 0A3C  07 0F79  2966        BEGIN
24 0A40  07 0F79  2967          R0 := 0; V34(R8) := R0;
24 0A48  07 0F79  2968        END;
24 0A48  07 0F79  2969                                                          52
24 0A48  07 0F79  2970        COMMENT    <FPAR HEAD*> ::=<FPAR HEAD-><SI TYPE>VALUE RESULT<ID>;
24 0A48  07 0F79  2971        VALUEP;
24 0A50  07 0F79  2972                                                          53
24 0A50  07 0F79  2973        COMMENT    <FPAR HEAD*> ::= <FPAR HEAD->  PROCEDURE  <ID>;
24 0A50  07 0F79  2974        BEGIN
24 0A54  07 0F79  2975          R0 := 0;  V34(R8) := R0;
24 0A5C  07 0F79  2976        END;
24 0A5C  07 0F79  2977                                                          54
24 0A5C  07 0F79  2978        COMMENT    <FPAR HEAD*> ::= <FPAR HEAD*> ,, <ID>;
24 0A5C  07 0F79  2979        BEGIN
24 0A60  07 0F79  2980          R1 := V34(R8);
24 0A64  07 0F79  2981          IF R1 ¬= 0 THEN VALUEP;
24 0A6E  07 0F79  2982        END;
24 0A6E  07 0F79  2983                                                          55
```

```
24 0A6E  07 0F79  2984          COMMENT    <FPAR HEAD->  ::= <FPAR HEAD*>  .,     ;
24 0A6E  07 0F79  2985          NULL;
24 0A72  07 0F79  2986                                                              56
24 0A72  07 0F79  2987          COMMENT    <FPAR HEAD-> ::= <FBND LIST>  .,    ;
24 0A72  07 0F79  2988          NULL;
24 0A76  07 0F79  2989                                                              57
24 0A76  07 0F79  2990          COMMENT    <FBND LIST>  ::= <FBND HEAD>  *  )  ;
24 0A76  07 0F79  2991          BEGIN
24 0A7A  07 0F79  2992            R4 := DRELAD + 3 SHRA 2 SHLA 2; R2:=V1(R8); IC(R0,DIMEN(R2));
24 0A92  07 0F79  2993            R1 := R0 * 12S +4;
24 0A9C  07 0F79  2994            FOR R3 := 1 STEP 1 UNTIL V2(R8) DO
24 0AA0  07 0F79  2995            BEGIN IDLOC2(R2) := R4;
24 0AA8  07 0F79  2996              R2 := R2 + 12;  R4 := R4 + R1;
24 0AAE  07 0F79  2997            END;
24 0ABA  07 0F79  2998            DRELAD := R4;
24 0ABE  07 0F79  2999          END;
24 0ABE  07 0F79  3000                                                              58
24 0ABE  07 0F79  3001          COMMENT    <FBND HEAD>  ::= <F ARRAY HD>   (     ;
24 0ABE  07 0F79  3002           NULL;
24 0AC2  07 0F79  3003                                                              59
24 0AC2  07 0F79  3004          COMMENT    <FBND HEAD>  ::= <FBND HEAD>  *     , ;
24 0AC2  07 0F79  3005           NULL;
24 0AC6  07 0F79  3006                                                              60
24 0AC6  07 0F79  3007          COMMENT    <F ARRAY HD> ::= (  <SI TYPE>   ARRAY   <ID>    ;
24 0AC6  07 0F79  3008          BEGIN
24 0ACA  07 0F79  3009            DECLARETEST;   R1 := 1;  V2(R8) := R1;
24 0ADE  07 0F79  3010          END;
24 0ADE  07 0F79  3011                                                              61
24 0ADE  07 0F79  3012          COMMENT    <F ARRAY HD> ::= <FPAR HEAD-> <SI TYPE>  ARRAY <ID> ;
24 0ADE  07 0F79  3013          BEGIN
24 0AE2  07 0F79  3014            DECLARETEST;   R1 := 1;  V2(R8) := R1;
24 0AF6  07 0F79  3015          END;
24 0AF6  07 0F79  3016                                                              62
24 0AF6  07 0F79  3017          COMMENT    <F ARRAY HD> ::= <F ARRAY HD>  ,,  <ID>  ;
24 0AF6  07 0F79  3018          BEGIN
24 0AFA  07 0F79  3019            R1 := V2(R8)+1;  V2(R8) := R1;
24 0B06  07 0F79  3020          END;
24 0B06  07 0F79  3021                                                              63
24 0B06  07 0F79  3022          COMMENT    <RC CL DC>   ::=   <RC HEAD>     );
24 0B06  07 0F79  3023          BEGIN
24 0B0A  07 0F79  3024            R1 := V5(R8);   R4 := V2(R8);
24 0B12  07 0F79  3025            SIMTYPEINFO(R1) := R4;  COMMENT INSERT RECORD LENGTH;
24 0B16  07 0F79  3026          END;
24 0B16  07 0F79  3027                                                                64
24 0B16  07 0F79  3028          COMMENT    <RC HEAD>   ::=  <RECORD>  (  <SI TYPE>  <ID>  ;
24 0B16  07 0F79  3029          FIELDDC;
24 0B1E  07 0F79  3030                                                                65
24 0B1E  07 0F79  3031          COMMENT    <RC HEAD>   ::=  <RC HEAD>  ,,  <ID>  ;
24 0B1E  07 0F79  3032          FIELDDC;
24 0B26  07 0F79  3033                                                                66
24 0B26  07 0F79  3034          COMMENT    <RC HEAD>  ::=  <RC HEAD*>  <SI TYPE>  <ID>  ;
24 0B26  07 0F79  3035          FIELDDC;
24 0B2E  07 0F79  3036                                                                67
24 0B2E  07 0F79  3037          COMMENT    <RC HEAD*>   ::=  <RC HEAD>  .,    ;
24 0B2E  07 0F79  3038          NULL;
24 0B32  07 0F79  3039                                                                68
24 0B32  07 0F79  3040          COMMENT    <RECORD>   ::=  RECORD  <ID>  ;
24 0B32  07 0F79  3041          BEGIN
```

```
24 0B36  07 0F79  3042          DECLARETEST;
24 0B42  07 0F79  3043          R0 := @RCCLDC;  OUTID;  COMMENT  OUTPUT "RCCLDC";
24 0B52  07 0F79  3044          R3 := HN;  IDLOC1(R1) := R3;
24 0B5A  07 0F79  3045          V5(R8) := R1;  COMMENT  POINTER TO NAMETABLE IN V5(J);
24 0B5E  07 0F79  3046          R4 := 4;  R1 := R1 + 12;  IC(R0,TYPE(R1));  R2 := 0;
24 0B6E  07 0F79  3047          WHILE  R0 = 5 DO
24 0B76  07 0F79  3048          BEGIN  IC(R0,SIMPLETYPE(R1));
24 0B7A  07 0F79  3049            IF R0 = 9 THEN
24 0B82  07 0F79  3050            BEGIN  IDLOC2(R1) := R4; R4 := R4+ 4;
24 0B8A  07 0F79  3051            END;
24 0B8A  07 0F79  3052            IDLOC1(R1) := R3;   R2 := R2 + 1;
24 0B92  07 0F79  3053            R1 := R1 + 12;   IC(R0,TYPE(R1));
24 0B9A  07 0F79  3054          END;
24 0B9E  07 0F79  3055          R3 := R1-12;  R1 := V5(R8);  STC(R2,VR(R1));
24 0BAC  07 0F79  3056          IF R4 = 4 THEN R4 := 1;
24 0BB8  07 0F79  3057          FOR R1 := R1 + 12 STEP 12 UNTIL R3 DO
24 0BBC  07 0F79  3058          BEGIN  IC(R0,SIMPLETYPE(R1));
24 0BC4  07 0F79  3059            IF R0 = 6 THEN
24 0BCC  07 0F79  3060            BEGIN  IDLOC2(R1) := R4;  R4 := R4 + 1;
24 0BD4  07 0F79  3061            END ELSE
24 0BD4  07 0F79  3062            IF R0 = 7 THEN
24 0BEC  07 0F79  3063            BEGIN  IDLOC2(R1) := R4;  R4 := R4 + SIMTYPEINFO(R1) + 1;
24 0BEC  07 0F79  3064            END;
24 0BEC  07 0F79  3065          END;
24 0BF6  07 0F79  3066          R4 := R4 + 3 SHRL 2 SHLL 2;
24 0C02  07 0F79  3067          FOR R1 := V5(R8) + 12 STEP 12 UNTIL R3 DO
24 0C0A  07 0F79  3068          BEGIN  IC(R0,SIMPLETYPE(R1));
24 0C12  07 0F79  3069            IF R0 = 1 OR R0 = 2 OR R0 = 4 OR R0 = 8 THEN
24 0C32  07 0F79  3070            BEGIN  IDLOC2(R1) := R4;  R4 := R4 + 4;
24 0C3A  07 0F79  3071            END;
24 0C3A  07 0F79  3072            IF R0 = 4 THEN R4 := R4 + 4;
24 0C46  07 0F79  3073          END;
24 0C50  07 0F79  3074          V2(R8) := R4;
24 0C54  07 0F79  3075        END;
24 0C54  07 0F79  3076    END;
24 0D68  07 0F79  3077  END;

                        SEGMENT 24  NAME = SEG#24      LENGTH = 0DC0  BASE REG = 15

23 0474  07 0F79  3078
23 0474  07 0F79  3079   SEGMENT PROCEDURE   EXECUTE2(R6);
23 0474  07 0F79  3080  BEGIN
25 0000  07 0F79  3081
25 0000  07 0F79  3082  PROCEDURE  ARITHCHECK(R5);
25 0000  07 0F79  3083  COMMENT  FIRST ARGUMENT IS IN R1, SECOND IS IN V(I);
25 0000  07 0F79  3084    BEGIN   IC(R0,V22(R7));
25 0008  07 0F79  3085      IF R0 > 5 THEN   BEGIN  R0 := @TYPEERROR;  ERROR;  R0 := R1;
25 0022  07 0F79  3086                       END  ELSE
25 0022  07 0F79  3087      IF R1 > 5 THEN   BEGIN  R0 := @TYPEERROR;  ERROR;  R1 := R0;
25 0040  07 0F79  3088                       END;
25 0040  07 0F79  3089    END;
25 0042  07 0F79  3090
25 0042  07 0F79  3091  PROCEDURE  CASETYPE(R10);
25 0042  07 0F79  3092    BEGIN  INTEGER RASAVE;
25 0042  07 0F80  3093      IC(R1,V22(R8));  IC(R0,V22(R8+8));  STC(R0,V22(R7));
25 004E  07 0F80  3094      IF R1 ¬= 0 THEN
25 0054  07 0F80  3095      BEGIN   RASAVE  := R10;
25 0058  07 0F80  3096
```

```
25 0058  07 0F80  3097        IF R0 ¬= R1 THEN
25 005E  07 0F80  3098        BEGIN    ARITHCHECK;
25 0062  07 0F80  3099          R5 := R0 AND R1 AND #1;  COMMENT RECOVER LONG BIT;
25 006A  07 0F80  3100          IF R0 > R1 THEN
25 0070  07 0F80  3101          BEGIN  IF R5 = 0 THEN R0 := R0 AND #FE;  STC(R0,V22(R8));
25 007E  07 0F80  3102          END ELSE
25 007E  07 0F80  3103          BEGIN  IF R5 = 0 THEN R1 := R1 AND #FE;  STC(R1,V22(R8));
25 0090  07 0F80  3104          END;
25 0090  07 0F80  3105        END  ELSE  IF R1 = 7 THEN
25 009C  07 0F80  3106        BEGIN  R5 := V1(R8);
25 00A0  07 0F80  3107          IF R5=0 THEN
25 00A6  07 0F80  3108          BEGIN  R1 := V1(R8+8);  V1(R8) := R1;
25 00AE  07 0F80  3109          END  ELSE IF R5 ¬= V1(R8+8) THEN
25 00BA  07 0F80  3110          BEGIN  R0 := @LENGTHERROR;   ERROR;
25 00CA  07 0F80  3111          END;
25 00CA  07 0F80  3112        END  ELSE  IF R1 = 9 THEN
25 00D6  07 0F80  3113        BEGIN  R5 := V1(R8+8);  R1 := V1(R8) OR R5;  V1(R8) := R1;
25 00E4  07 0F80  3114        END;
25 00E4  07 0F80  3115        R10 := RASAVE;
25 00E8  07 0F80  3116
25 00E8  07 0F80  3117        END  ELSE
25 00E8  07 0F80  3118        BEGIN  STC(R1,V21(R8+8)); R1 := V12(R8+8);  V12(R8) := R1;
25 00F8  07 0F80  3119        END;
25 00F8  07 0F80  3120      END;
25 00FA  07 0F80  3121
25 00FA  07 0F80  3122    PROCEDURE  CONVERT1(R5);
25 00FA  07 0F80  3123    COMMENT  FIRST ARGUMENT IS IN R1,  SECOND IS IN R0,  POINTER FOR
25 00FA  07 0F80  3124                FIRST IS IN R2;
25 00FA  07 0F80  3125
25 00FA  07 0F80  3126      IF R0 ¬= R1 THEN
25 0100  07 0F80  3127      BEGIN
25 01C0  07 0F80  3128        IF R0 = 1 THEN  STC(R1,CONV(R9)   )ELSE
25 010C  07 0F80  3129          CASE R1 OF
25 0110  07 0F80  3130          BEGIN
25 0110  07 0F80  3131            BEGIN  STC(R0,V22(R8));  STC(R0,CONV(R2));
25 0120  07 0F80  3132            END;
25 0120  07 0F80  3133            IF R0 = 3 THEN
25 012C  07 0F80  3134            BEGIN  R0 := 2;  STC(R0,CONV(R9));
25 0134  07 0F80  3135            END  ELSE
25 0134  07 0F80  3136            IF R0 = 4 THEN
25 0140  07 0F80  3137            BEGIN  STC(R0,V22(R8));  STC(R0,CONV(R2));
25 0148  07 0F80  3138            END  ELSE
25 0148  07 0F80  3139            BEGIN  R0 := 4;  STC(R0,V22(R8));  STC(R0,CONV(R9));
25 0158  07 0F80  3140              STC(R0,CONV(R2));
25 015C  07 0F80  3141            END;
25 015C  07 0F80  3142            BEGIN  STC(R0,CONV(R2));  STC(R0,V22(R8));
25 0168  07 0F80  3143            END;
25 0168  07 0F80  3144            BEGIN  R0 := 4;  STC(R0,CONV(R9));
25 0174  07 0F80  3145            END;
25 0174  07 0F80  3146            IF R0 = 2 THEN
25 0180  07 0F80  3147            BEGIN  R0 := 4;  STC(R0,V22(R8));  STC(R0,CONV(R9));
25 018C  07 0F80  3148              STC(R0,CONV(R2));
25 0190  07 0F80  3149            END  ELSE
25 0190  07 0F80  3150            IF R0 = 4 THEN
25 019C  07 0F80  3151            BEGIN  STC(R0,V22(R8));  STC(R0,CONV(R2));
25 01A4  07 0F80  3152            END  ELSE
25 01A4  07 0F80  3153            BEGIN  R0 := 5;  STC(R0,CONV(R9));
25 01B0  07 0F80  3154            END;
```

```
25 0180   07 0F80   3155          END;
25 01C8   07 0F80   3156        END;
25 01CA   07 0F80   3157
25 01CA   07 0F80   3158      PROCEDURE  APAREND(R5);
25 01CA   07 0F80   3159      COMMENT  CHECKS NUMBER OF APARS, TYPE OF LAST.  OUTPUTS " )" ;
25 01CA   07 0F80   3160        BEGIN
25 01CA   07 0F80   3161          IC(R0,V21(R8));                COMMENT  CHECK NUMBER OF APARS;
25 01CE   07 0F80   3162          IF R0 ¬= 1 AND R0 ¬= #FF THEN
25 01DE   07 0F80   3163          BEGIN  R0 := @APARERROR;   ERROR;
25 01EE   07 0F80   3164          END;
25 01EE   07 0F80   3165          IC(R0,V22(R8));  IF R0 = 0 THEN R0 := PROCEDURETYPE ELSE R0 := 0;
25 0204   07 0F80   3166          STC(R0,V21(R8));               COMMENT  SET TYPE;
25 02C8   07 0F80   3167          R0 := @APPAREN;          COMMENT  PREPARE TO OUTPUT;
25 02CC   07 0F80   3168        END;
25 02CE   07 0F80   3169
25 02CE   07 0F80   3170      PROCEDURE  APARLIST(R5);
25 02CE   07 0F80   3171      COMMENT  CHECKS NUMBER OF APARS, INCREASES FPAR POINTER ,
25 02CE   07 0F80   3172               OUTPUTS  ","   .  POINTER TO FPAR IS IN R2;
25 02CE   07 0F80   3173        BEGIN
25 02CE   07 0F80   3174          IC(R0,V21(R8));                COMMENT  GET NUMBER OF APARS LEFT;
25 0212   07 0F80   3175          IF R0 = 1 THEN
25 021A   07 0F80   3176          BEGIN  R0 := @APARERROR;   ERROR;
25 022A   07 0F80   3177            R0 := #FF;  STC(R0,V21(R8));
25 0232   07 0F80   3178          END ELSE
25 0232   07 0F80   3179          BEGIN   R0 := R0 - 1;
25 023A   07 0F80   3180            STC(R0,V21(R8));
25 023E   07 0F80   3181          END;
25 023E   07 0F80   3182          R2 := R2 + 12;  V34(R8) := R2;  COMMENT  STEP FPAR POINTER;
25 0246   07 0F80   3183          R0 := @APCOMMA;           COMMENT  PREPARE TO OUTPUT;
25 024A   07 0F80   3184        END;
25 024C   07 0F80   3185
25 024C   07 0F80   3186      PROCEDURE  SUBSCRIPT(R5);
25 024C   07 0F80   3187      COMMENT  SUBSCRIPT CHECKS THAT SUBSCRIPT IS INTEGER AND THAT DIMENSION
25 024C   07 0F80   3188               IS NOT TOO SMALL FOR IT  ;
25 024C   07 0F80   3189        BEGIN
25 024C   07 0F80   3190          IC(R0,V22(R8+8));                  COMMENT   GET  SIMPLE  TYPE ;
25 0250   07 0F80   3191          IF R0 ¬= 1  THEN  BEGIN  R0 := @TYPEERROR ;  ERROR;   END;
25 0268   07 0F80   3192          R2 := 0;  IC(R2,V22(R8));
25 0270   07 0F80   3193          IF R2 = #FF THEN
25 0278   07 0F80   3194          BEGIN  R2 := V1(R8);  IC(R0,DIMEN(R2));  R0 := R0 + 1;
25 0284   07 0F80   3195            STC(R0,DIMEN(R2));
25 0288   07 0F80   3196          END ELSE
25 0288   07 0F80   3197          BEGIN  R2 := R2 - 1;  STC(R2,V22(R8));
25 0294   07 0F80   3198            IF R2 < 0 THEN
25 029A   07 0F80   3199            BEGIN  R0 := @ARRAYERROR;  ERROR;
25 02AA   07 0F80   3200            END;
25 02AA   07 0F80   3201          END;
25 02AA   07 0F80   3202          MAXREG;           COMMENT  ADJUST REGISTER COUNTS;
25 02B6   07 0F80   3203          R0 := @INDX;                 COMMENT   PREPARE TO OUTPUT;
25 02BA   07 0F80   3204        END;
25 02BC   07 0F80   3205
25 02BC   07 0F80   3206
25 02BC   07 0F80   3207      PROCEDURE  ARRAYDES(R5);
25 02BC   07 0F80   3208      COMMENT  R0 CONTAINS NUMBER OF *"S;
25 02BC   07 0F80   3209        BEGIN
25 02BC   07 0F80   3210          R1 := V1(R8);  R2 := 0;  IC(R2,V22(R8));
25 02C8   07 0F80   3211          IF R2 = #FF THEN
25 02D0   07 0F80   3212          BEGIN  IC(R2,DIMEN(R1));  R2 := R2 + R0;  STC(R2,DIMEN(R1));
```

```
25 02DA  C7 0F80   3213      END ELSE
25 02DA  C7 0F80   3214      IF R0 ¬= R2 THEN
25 02E4  07 0F80   3215      BEGIN  R3 := R0;  R0 := @ARRAYERROR;  ERROR;  R0 := R3;
25 02F8  C7 0F80   3216      END;
25 02F8  07 0F80   3217      IF R0 = 0 THEN
25 02FE  07 0F80   3218      BEGIN  IC(R0,SIMPLETYPE(R1));  V2(R8) := R0;
25 03C6  07 0F80   3219      IC(R0,V3(R8));
25 030A  07 0F80   3220      IF R0=0 THEN
25 0310  07 0F80   3221      BEGIN R0 := R0 + 1;  STC(R0,V3(R8));
25 0318  07 0F80   3222      END;
25 0318  07 0F80   3223      END ELSE
25 0318  07 0F80   3224      BEGIN  R2 := TYPES(R1);  V2(R8) := R2;  V34(R8) := R0;
25 0328  07 0F80   3225        R8 := R8 SHRL 3;  IC(R0,S(R8-1));  R8 := R8 SHLL 3;
25 0334  07 0F80   3226        IF R0 ¬= APARHEAD THEN
25 033C  07 0F80   3227        BEGIN  R0 := @ARRAY2ERROR;  ERROR;
25 034C  07 0F80   3228        END;
25 034C  07 0F80   3229      END;
25 034C  07 0F80   3230      R2 := SIMTYPEINFO(R1);  V1(R8) := R2;
25 0354  07 0F80   3231    END;
25 0356  07 0F80   3232
25 0356  07 0F80   3233  PROCEDURE APSTATEND(R4);
25 0356  07 0F80   3234    BEGIN
25 0356  07 0F80   3235      R9 := R9 + 4;    R0 := @STACKADDR;    STC(R0,OP(R9));
25 0362  07 0F80   3236      R1 := DRELAD;    POINTER(R9) := R1;
25 036A  07 0F80   3237      R2 := V34(R8);  IC(R0,V21(R8));
25 0372  07 0F80   3238      IF R2 = 0 THEN R0 := @APPAREN ELSE
25 037C  07 0F80   3239      IF R0 = #FF THEN APAREND ELSE
25 038C  07 0F80   3240      BEGIN  R1 := TYPES(R2);
25 0394  07 0F80   3241        IF R1 ¬= #1300 THEN
25 039C  07 0F80   3242        BEGIN  R0 := @TYPE3ERROR;  ERROR;
25 03AC  07 0F80   3243        END;
25 03AC  07 0F80   3244        APAREND;
25 03B0  07 0F80   3245      END;
25 03B0  07 0F80   3246      R1 := #0404;  V34(R8) := R1;
25 03B8  07 0F80   3247      OUTOP;
25 03C4  07 0F80   3248      R1 := HN + 1;  HN := R1;
25 03D0  07 0F80   3249      R2 := DRELPOINT;    R1 := DRELSAVE(R2);    DRELAD := R1;
25 03DC  07 0F80   3250      R2 := R2 - 2;    DRELPOINT := R2;
25 03E4  07 0F80   3251      R2 := VARORIGIN - 4;    VARORIGIN := R2;
25 03F0  07 0F80   3252    END;
25 03F2  07 0F80   3253
25 03F2  07 0F80   3254  PROCEDURE APSTATLIST(R4);
25 03F2  07 0F80   3255    BEGIN
25 03F2  07 0F80   3256      R9 := R9 + 4;  R0 := @STACKADDR;  STC(R0,OP(R9));  R1 := DRELAD;
25 0402  07 0F80   3257      POINTER(R9) := R1;  R1 := VARORIGIN;  DRELAD := R1;
25 040E  07 0F80   3258      R2 := V34(R8);  IC(R0,V21(R8));
25 0416  07 0F80   3259      IF R2 = 0 OR R0 = #FF THEN R0 := @APCOMMA ELSE
25 0428  07 0F80   3260      BEGIN  R1 := TYPES(R2);
25 0430  07 0F80   3261        IF R1 ¬= #1300 THEN
25 0438  07 0F80   3262        BEGIN  R0 := @TYPE3ERROR;  ERROR;
25 0448  07 0F80   3263        END;
25 0448  07 0F80   3264        APARLIST;
25 044C  07 0F80   3265      END;
25 044C  07 0F80   3266      OUTOP;
25 0458  07 0F80   3267    END;
25 045A  07 0F80   3268
25 045A  07 0F80   3269   SEGMENT PROCEDURE  LEVELTWO(R10);
25 045A  07 0F80   3270  BEGIN  INTEGER RASAVE;
```

```
26 0000  07 0F84  3271    RASAVE := R10;
26 0004  07 0F84  3272       RULENUMBER := R5;
26 0008  07 0F84  3273      R5 := R5 - 68;  CASE R5 OF
26 000C  07 0F84  3274      BEGIN
26 000C  07 0F84  3275                                                          69
26 000C  07 0F84  3276         COMMENT    <T VAR>    ::=  <SI T VAR>    ;
26 000C  07 0F84  3277         NULL;
26 0014  07 0F84  3278                                                          70
26 0014  07 0F84  3279         COMMENT    <T VAR>  ::=    <T ARRAY ID>  ;
26 0014  07 0F84  3280         BEGIN
26 0018  07 0F84  3281           R1 := V1(R8);  R2 := SIMTYPEINFO(R1);  V1(R8) := R2;
26 0024  07 0F84  3282           IC(R0,DIMEN(R1));  V34(R8) := R0;
26 002C  07 0F84  3283            R8 := R8 SHRL 3;  IC(R0,S(R8-1));  R8 := R8 SHLL 3;
26 0038  07 0F84  3284            IF R0 ¬= APARHEAD THEN
26 0040  07 0F84  3285            BEGIN  R0 := @ARRAY2ERROR;  ERROR;
26 0050  07 0F84  3286            END;
26 0050  07 0F84  3287         END;
26 0050  07 0F84  3288                                                          71
26 005C  07 0F84  3289         COMMENT    <T VAR>  ::= <STR SEL HD>  <T EXP>  <LENGTH>   )  ;
26 0050  07 0F84  3290         BEGIN
26 0054  07 0F84  3291           IC(R0,V22(R8+8));
26 0058  07 0F84  3292           IF R0 ¬=1 THEN BEGIN R0 := @TYPEERROR; ERROR;END;
26 0070  07 0F84  3293           R1 := V1(R8+16);    IF R1 > V1(R8) THEN
26 007C  07 0F84  3294             BEGIN   R0 := @LENGTHERROR;   ERROR;
26 008C  07 0F84  3295              END   ELSE  V1(R8) := R1;
26 0094  07 0F84  3296           R0 := @BAR; R9 := @B9(4); STC(R0,OP(R9));
26 00A0  07 0F84  3297           R1 := V5(R8+8) - OUTBASE;  POINTER(R9) := R1;
26 00AC  07 0F84  3298           R0 := @SUBSTRING;  R0 := R0 OR #80;  OUTOP;
26 00C0  07 0F84  3299         END;
26 00CC  07 0F84  3300                                                          72
26 00C0  07 0F84  3301      COMMENT    <T VAR>   ::=   <UNIT DES>;
26 00C0  07 0F84  3302      NULL;  COMMENT  * * * * * * * * * * *;
26 00C4  07 0F84  3303                                                          73
26 00C4  07 0F84  3304      COMMENT    <STR SEL HD>    ::=     <SI T VAR>   (  ;
26 00C4  07 0F84  3305      BEGIN
26 00C8  07 0F84  3306        IC(R0,V22(R8));
26 00CC  07 0F84  3307        IF R0 ¬= 7  THEN BEGIN R0 := @TYPEERROR;  ERROR;
26 00E4  07 0F84  3308                         END;
26 00E4  07 0F84  3309        IC(R0,V21(R8));
26 00E8  07 0F84  3310        IF R0 = 2  THEN BEGIN R0 := @ARRAYERROR; ERROR;
26 0100  07 0F84  3311                         END;
26 0100  07 0F84  3312      END;
26 0100  07 0F84  3313                                                          74
26 0100  07 0F84  3314      COMMENT    <LENGTH>  ::=   |   <T NUMBER>  ;
26 0100  07 0F84  3315      BEGIN
26 0104  07 0F84  3316        IC(R0,V22(R7));
26 01C8  07 0F84  3317        IF R0 ¬= 1 THEN BEGIN R0 := @TYPEERROR; ERROR; END;
26 0120  07 0F84  3318        R5 := NUMVALUE4;
26 0124  07 0F84  3319        IF R5 <= 0 OR R5 > 256 THEN
26 0132  07 0F84  3320        BEGIN  R0 := @STRINGERROR;  ERROR;
26 0142  07 0F84  3321        END;
26 0142  07 0F84  3322        R5 := R5 - 1;    V1(R8) := R5;    NUMVALUE4 := R5;
26 014E  07 0F84  3323        R0 := @INUMBER;  OUTID;
26 015E  07 0F84  3324      END;
26 015E  07 0F84  3325                                                          75
26 015E  07 0F84  3326      COMMENT    <SI T VAR>     ::=   <T VAR ID>  ;
26 015E  07 0F84  3327      NULL;
26 0162  07 0F84  3328                                                          76
```

```
26 0162  07 0F84  3329          COMMENT    <SI T VAR>     ::=  <T FLD HD>  <T EXP>  )   ;
26 0162  07 0F84  3330          BEGIN
26 0166  07 0F84  3331            IC(R0,V22(R8+8));
26 016A  07 0F84  3332            IF R0 ¬= 9 THEN
26 0172  07 0F84  3333            BEGIN  R0 := @TYPEERROR;  ERROR;
26 0182  07 0F84  3334            END ELSE
26 0182  07 0F84  3335            BEGIN  IC(R1,V21(R8));  R2 := 1 SHLL R1 SHRL 1;
26 0196  07 0F84  3336              R3 := V1(R8+8);  IF R3 ¬= R2 THEN
26 01A0  07 0F84  3337              BEGIN  R3 := R3 AND R2;  IF = THEN
26 01A6  07 0F84  3338                 BEGIN  R0 := @REFERROR;  ERROR;
26 01B6  07 0F84  3339                 END ELSE
26 01B6  07 0F84  3340                 BEGIN  R3 := V5(R8);  STC(R1,CONV(R3));
26 01C2  07 0F84  3341                 END;
26 01C2  07 0F84  3342              END;
26 01C2  07 0F84  3343            END;
26 01C2  07 0F84  3344            R0 := @REFX;  R0 := R0 OR #80;   OUTOP;
26 01D6  07 0F84  3345            R0 := 0;  STC(R0,V21(R8));
26 01DE  07 0F84  3346          END;
26 01DE  07 0F84  3347                                                                  77
26 01DE  07 0F84  3348          COMMENT    <SI T VAR>     ::=  <T ARRAY HD>  <T EXP>  )  ;
26 01DE  07 0F84  3349          BEGIN
26 01E2  07 0F84  3350            SUBSCRIPT;   OUTOP;
26 01FA  07 0F84  3351            IC(R0,V21(R8));  ARRAYDES;
26 020A  07 0F84  3352          END;
26 020A  07 0F84  3353                                                                  78
26 020A  07 0F84  3354          COMMENT    <SI T VAR>     ::=  <T ARRAY HD>   *   )   ;
26 020A  07 0F84  3355          BEGIN
26 020E  07 0F84  3356            R9 := R9 + 4;  R0 := @ARSTAR;  STC(R0,OP(R9));
26 021A  07 0F84  3357            R0 := @INDX;  OUTOP;
26 022A  07 0F84  3358            IC(R0,V21(R8));  R0 := R0 + 1;  ARRAYDES;
26 023E  07 0F84  3359          END;
26 023E  07 0F84  3360                                                                  79
26 023E  07 0F84  3361          COMMENT    <T FLD HD>   ::=  <T FLD ID>   (    ;
26 023E  07 0F84  3362          NULL;
26 0242  07 0F84  3363                                                                  80
26 0242  07 0F84  3364          COMMENT    <T ARRAY HD> ::=  <T ARRAY ID>  (    ;
26 0242  07 0F84  3365          BEGIN
26 0246  07 0F84  3366            R1 := V1(R8);  IC(R0,DIMEN(R1));
26 024E  07 0F84  3367            IF R0 = 0 THEN R0 := #FF;  V2(R8) := R0;
26 025C  07 0F84  3368          END;
26 025C  07 0F84  3369                                                                  81
26 025C  07 0F84  3370          COMMENT    <T ARRAY HD> ::= <T ARRAY HD> <T EXP>  ,  ;
26 025C  07 0F84  3371          BEGIN
26 0260  07 0F84  3372            SUBSCRIPT;   OUTOP;
26 0278  07 0F84  3373          END;
26 0278  07 0F84  3374                                                                  82
26 0278  07 0F84  3375          COMMENT    <T ARRAY HD> ::= <T ARRAY HD>   *   ,    ;
26 0278  07 0F84  3376          BEGIN
26 027C  07 0F84  3377            R9 := R9 + 4;  R0 := @ARSTAR;  STC(R0,OP(R9));
26 0288  07 0F84  3378            IC(R0,V3(R8));   R0 := R0+1;  STC(R0,V3(R8));
26 0294  07 0F84  3379            IC(R0,V21(R8));  R0 := R0 + 1;  STC(R0,V21(R8));
26 02A0  07 0F84  3380            R0 := @INDX;   OUTOP;
26 02B0  07 0F84  3381          END;
26 02B0  07 0F84  3382                                                                  83
26 02B0  07 0F84  3383          COMMENT    <T FUNC DES> ::=  <T FUNC ID>  ;
26 02B0  07 0F84  3384          BEGIN
26 02B4  07 0F84  3385            R2 := V1(R8);  R8 := R8 SHRL 3;  IC(R0,S(R8-1));
26 02C0  07 0F84  3386            R8 := R8 SHLL 3;
```

```
26 02C4  07 0F84  3387          IF R0 ¬= APARHEAD THEN
26 02CC  07 0F84  3388          BEGIN  R1 := TYPEINFO(R2) SHLL 2;
26 02D4  07 0F84  3389            IF R1 ¬= 0 THEN
26 02DA  07 0F84  3390            BEGIN  R0 := @APARERROR;  ERROR;
26 02EA  07 0F84  3391            END;
26 02EA  07 0F84  3392          END;
26 02EA  07 0F84  3393          R1 := SIMTYPEINFO(R2);  V1(R8) := R1;
26 02F2  07 0F84  3394        END;
26 02F2  07 0F84  3395                                                                    84
26 02F2  07 0F84  3396        COMMENT    <T FUNC DES> ::= <APAR HEAD> <T EXP>    ) ;
26 02F2  07 0F84  3397        BEGIN
26 02F6  07 0F84  3398          R2 := V34(R8);  IC(R0,V21(R8));
26 02FE  07 0F84  3399          IF R2 = 0 THEN R0 := @APPAREN ELSE
26 0308  07 0F84  3400          IF R0 = #FF THEN APAREND ELSE
26 0320  07 0F84  3401          BEGIN    APAREXPCHECK;  APAREND;
26 033C  07 0F84  3402          END;
26 033C  07 0F84  3403          R1 := #0404;  V34(R8) := R1;
26 0344  07 0F84  3404          OUTOP;
26 0350  07 0F84  3405          R1 := HN + 1;  HN := R1;
26 035C  07 0F84  3406          R2 := DRELPOINT;    R1 := DRELSAVE(R2);    DRELAD := R1;
26 0368  07 0F84  3407          R2 := R2 - 2;    DRELPOINT := R2;
26 037C  07 0F84  3408          R2 := VARORIGIN - 4;    VARORIGIN := R2;
26 037C  07 0F84  3409        END;
26 037C  07 0F84  3410                                                                    85
26 037C  07 0F84  3411        COMMENT    <T FUNC DES> ::= <APAR HEAD>  <STATEMENT>    )  ;
26 037C  07 0F84  3412        APSTATEND;
26 038C  07 0F84  3413                                                                    86
26 038C  07 0F84  3414        COMMENT    <T FUNC DES>   ::=  <APAR HEAD>  )  ;
26 038C  07 0F84  3415        BEGIN
26 0390  07 0F84  3416          R0 := @NULLST;    R9 := R9 + 4;    STC(R0,OP(R9));
26 039C  07 0F84  3417          APSTATEND;
26 03A8  07 0F84  3418        END;
26 03A8  07 0F84  3419                                                                    87
26 03A8  07 0F84  3420        COMMENT    <APAR HEAD>  ::= <T FUNC ID>   (   ;
26 03A8  07 0F84  3421        BEGIN
26 03AC  07 0F84  3422          R2 := V1(R8);  IC(R0,TYPE(R2));
26 03B4  07 0F84  3423          IF R0 = #13 THEN R1 := 0 ELSE
26 03C0  07 0F84  3424          BEGIN   R2 := TYPEINFO(R2) SHLA 2;
26 03CC  07 0F84  3425            R0 := 0;  R1 := BLENGTH(R2)/12;  STC(R1,V21(R8));
26 03DC  07 0F84  3426            IF R1 = 0 THEN
26 03E2  07 0F84  3427            BEGIN  R0 := @APARERROR;  ERROR;
26 03F2  07 0F84  3428            END;
26 03F2  07 0F84  3429            R1 := NPOINT(R2);
26 03F6  07 0F84  3430            R2 := V1(R8);
26 03FA  07 0F84  3431          END;
26 03FA  07 0F84  3432          V34(R8) := R1;
26 03FE  07 0F84  3433          R1 := SIMTYPEINFO(R2);    V1(R8) := R1;
26 0406  07 0F84  3434          R1 := HN - 1;    HN := R1;    IF R1 <= 5 THEN
26 041A  07 0F84  3435          BEGIN  R0 := @HIERARCHYERROR;    ERROR;  END;
26 042A  07 0F84  3436          R1 := DRELAD;    R2 := DRELPOINT + 2;  DRELPOINT := R2;
26 043A  07 0F84  3437          DRELSAVE(R2) := R1;
26 043E  07 0F84  3438          R2 := VARORIGIN + 4;    VARORIGIN := R2;    DRELAD := R2;
26 044E  07 0F84  3439        END;
26 044E  07 0F84  3440                                                                    88
26 044E  07 0F84  3441        COMMENT    <APAR HEAD>  ::= <APAR HEAD> <T EXP>  ,  ;
26 044E  07 0F84  3442        BEGIN
26 0452  07 0F84  3443          R2 := V34(R8);  IC(R0,V21(R8));
26 045A  07 0F84  3444          IF R2 = 0 OR R0 = #FF THEN R0 := @APCOMMA ELSE
```

```
26 046C  07 0F84  3445          BEGIN    APAREXPCHECK;  APARLIST;
26 0488  07 0F84  3446          END;
26 0488  07 0F84  3447          OUTOP;
26 0494  07 0F84  3448        END;
26 0494  07 0F84  3449                                                          89
26 0494  07 0F84  3450        COMMENT    <APAR HEAD>  ::=  <APAR HEAD>  <STATEMENT>   ,   ;
26 0494  07 0F84  3451        APSTATLIST;
26 04A4  07 0F84  3452                                                          90
26 04A4  07 0F84  3453        COMMENT    <APAR HEAD>  ::=  <APAR HEAD> ,   ;
26 04A4  07 0F84  3454        BEGIN
26 04A8  07 0F84  3455          R0 := @NULLST;    R9 := R9 + 4;  STC(R0,OP(R9));
26 04B4  07 0F84  3456          APSTATLIST;
26 04C0  07 0F84  3457        END;
26 04C0  07 0F84  3458                                                          91
26 04C0  07 0F84  3459        COMMENT    <T EXP> ::=  <T EXP*> ;
26 04C0  07 0F84  3460        NULL;
26 04C4  07 0F84  3461                                                          92
26 04C4  07 0F84  3462        COMMENT    <T EXP*>  ::= <SI T EXP>;
26 04C4  07 0F84  3463        NULL;
26 04C8  07 0F84  3464                                                          93
26 04C8  07 0F84  3465        COMMENT    <T EXP*> ::= <IF CL>  <TRUE EXP>   <T EXP*>   ;
26 04C8  07 0F84  3466        BEGIN
26 04CC  07 0F84  3467          IC(R1,V22(R8+8));  R2 := V5(R8+8) - 4;  IC(R0,V22(R7));
26 04DC  07 0F84  3468          R5 := V12(R8+8);  V12(R8) := R5;
26 04E4  07 0F84  3469          IF R0 ¬= R1 THEN
26 04EA  07 0F84  3470          BEGIN  ARITHCHECK;  CONVERT1;
26 0502  07 0F84  3471          END  ELSE
26 0502  07 0F84  3472          BEGIN
26 0506  07 0F84  3473            IF R1 = 7 THEN
26 050E  07 0F84  3474            BEGIN  R5 := V1(R8+8);
26 0512  07 0F84  3475              IF R5 ¬= V1(R7) THEN
26 051A  07 0F84  3476              BEGIN  R0 := @LENGTHERROR;  ERROR;
26 052A  07 0F84  3477              END;
26 052A  07 0F84  3478            END  ELSE IF R1 = 9 THEN
26 0536  07 0F84  3479            BEGIN  R5 := V1(R8+8);  R1 := V1(R7) OR R5;  V1(R8) := R1;
26 0544  07 0F84  3480            END;
26 0544  07 0F84  3481          END;
26 0544  07 0F84  3482          R9 := R9+4;  R0 := @IFEXP;
26 054C  07 0F84  3483          STC(R0,OP(R9));  R1 := V5(R8+8) - OUTBASE;
26 0558  07 0F84  3484          POINTER(R9) := R1;  R2 := V5(R8) - OUTBASE;
26 0564  07 0F84  3485          R1 := R1 + OUTBASE;
26 0568  07 0F84  3486          POINTER(R1):=R2;       COMMENT  PATCH  UJ POINTER;
26 056C  07 0F84  3487          V5(R8):=R9;
26 0570  07 0F84  3488          R8 := R8+8;  MAXREG;   R8 := R8 - 8;  MAXREG;
26 0590  07 0F84  3489        END;
26 0590  07 0F84  3490                                                          94
26 0590  07 0F84  3491        COMMENT    <T EXP*> ::= <CASE HEAD>  <T EXP>   )   ;
26 0590  07 0F84  3492        BEGIN
26 0594  07 0F84  3493          CASETYPE;              COMMENT CHECK TYPES, SET TYPE BYTE;
26 05A0  07 0F84  3494          R0 := @CLL; OUTOP;   COMMENT OUTPUT "CL";
26 05B0  07 0F84  3495          MAXREG;                COMMENT  ADJUST REGISTER COUNTS;
26 05BC  07 0F84  3496          IC(R0,V22(R8));       COMMENT PATCH TYPE IN OUTPUT;
26 05C0  07 0F84  3497          R1 := PATCHPOINT - 2;  PATCHPOINT := R1;   R1 := PATCHLIST(R1);
26 05CC  07 0F84  3498          STC(R0,CONV(R1));
26 05D4  07 0F84  3499          IC(R0,V21(R8));       COMMENT PATCH NUMBER OF CASES;
26 05D8  07 0F84  3500          R0:=R0+1;
26 05DC  07 0F84  3501          POINTER(R1):=R0;
26 05E0  07 0F84  3502          R0 := 0;  STC(R0,V21(R8));
```

```
26 05E8  07 0F84  3503          END;
26 05E8  07 0F84  3504                                                                  95
26 05E8  07 0F84  3505          COMMENT    <IF CL>   ::=  IF  <T EXP>    THEN  ;
26 05E8  07 0F84  3506          BEGIN
26 05EC  07 0F84  3507            IC(R0,V22(R8+8));
26 05F0  07 0F84  3508            IF R0 ¬= 6 THEN
26 05F8  07 0F84  3509            BEGIN  R0 := @TYPEERROR;  ERROR;
26 0600  07 0F84  3510            END;
26 0608  07 0F84  3511            R0 := @IFF;  STC(R0,OP(R9+4));  R9 := R9 + 8;  R0 := @IFJ;
26 0618  07 0F84  3512            R0 := R0 OR #80;  STC(R0,OP(R9));  R1 := V5(R8+8)-OUTBASE;
26 0628  07 0F84  3513            POINTER(R9) := R1;  V5(R8) := R9;
26 0630  07 0F84  3514            R1 := V34(R8+8);   V34(R8) := R1;
26 0638  07 0F84  3515          END;
26 0638  07 0F84  3516                                                                  96
26 0638  07 0F84  3517          COMMENT    <TRUE EXP>   ::= <T EXP>  ELSE  ;
26 0638  07 0F84  3518          BEGIN
26 063C  07 0F84  3519            R9 := @B9(4);
26 0640  07 0F84  3520            R0 := @UJIFEXP;  STC(R0,OP(R9));
26 0648  07 0F84  3521            V5(R8):=R9;
26 064C  07 0F84  3522          END;
26 064C  07 0F84  3523                                                                  97
26 064C  07 0F84  3524          COMMENT    <CASE HEAD>  ::= <CASE CL>   (  ;
26 064C  07 0F84  3525          BEGIN
26 0650  07 0F84  3526            R0 := 0;   V2(R8) := R0;
26 0658  07 0F84  3527            COMMENT  INITIALIZE CASE COUNT AND SIMPLE TYPE;
26 0658  07 0F84  3528          END;
26 0658  07 0F84  3529                                                                  98
26 0658  07 0F84  3530          COMMENT    <CASE HEAD>  :=  <CASE HEAD>  <T EXP>  ,  ;
26 065C  07 0F84  3531          BEGIN
26 0668  07 0F84  3532            CASETYPE;        COMMENT  CHECK TYPES,  SET TYPE BYTE;
26 0678  07 0F84  3533            R0 := @UJ;     OUTOP;  COMMENT OUTPUT UJ;
26 0684  07 0F84  3534            MAXREG;                COMMENT  ADJUST REGISTER COUNTS;
26 0688  07 0F84  3535            IC(R0,V21(R8));   COMMENT STEP CASE COUNT;
26 0690  07 0F84  3536            R0:=R0+1;     STC(R0,V21(R8));
26 0690  07 0F84  3537          END;
26 0690  07 0F84  3538                                                                  99
26 069C  07 0F84  3539          COMMENT    <CASE CL> ::= CASE  <T EXP>  OF  ;
26 0690  07 0F84  3540          BEGIN
26 0694  07 0F84  3541            IC(R0,V22(R8+8));
26 0698  07 0F84  3542            IF R0 ¬= 1 THEN BEGIN R0 := @TYPEERROR; ERROR; END;
26 06B0  07 0F84  3543            R9 := @B9(4);  R0 := @CASEE; STC(R0,OP(R9));
26 06BC  07 0F84  3544            V5(R8):=R9;
26 06C0  07 0F84  3545            R0 := 0;  V1(R8) := R0;
26 06C8  07 0F84  3546            R1 := PATCHPOINT;  PATCHLIST(R1) := R9;
26 06D0  07 0F84  3547            R1 := R1 + 2;   PATCHPOINT := R1;
26 06D8  07 0F84  3548            R1 := V34(R8+8);   V34(R8) := R1;
26 06E0  07 0F84  3549          END;
26 06EC  07 0F84  3550                                                                 100
26 06E0  07 0F84  3551          COMMENT    <SI T EXP>  ::= <SI T EXP**>;
26 06E0  07 0F84  3552          NULL;
26 06E4  07 0F84  3553                                                                 101
26 06E4  07 0F84  3554          COMMENT    <SI T EXP>  ::= <SI T EXP*> <EQL OP> <SI T EXP**>;
26 06E4  07 0F84  3555          BEGIN
26 06E8  07 0F84  3556            IC(R1,V22(R8));  IC(R0,V22(R7));
26 06F0  07 0F84  3557            IF R0 ¬= R1 THEN
26 06F6  07 0F84  3558            BEGIN    ARITHCHECK;
26 0702  07 0F84  3559              IF R0 < R1 THEN
26 0708  07 0F84  3560              BEGIN  IF R0 = 3 AND R1 = 4 THEN
```

```
26 0718  07 0F84  3561              BEGIN  R1 := 5;  R2 := V5(R8);  STC(R1,CONV(R2));
26 0724  07 0F84  3562              END;
26 0724  07 0F84  3563              STC(R1,CONV(R9));
26 0728  07 0F84  3564            END   ELSE
26 0728  07 0F84  3565            BEGIN  IF R1 = 3 AND R0 = 4 THEN
26 073C  07 0F84  3566              BEGIN  R0 := 5;  STC(R0,CONV(R9));
26 0744  07 0F84  3567              END;
26 0744  07 0F84  3568              R2 := V5(R8);  STC(R0,CONV(R2));
26 074C  07 0F84  3569            END;
26 074C  07 0F84  3570          END ELSE IF R1 = 9 THEN
26 0758  07 0F84  3571          BEGIN
26 0758  07 0F84  3572            R2 := V1(R8);  R3 := V1(R7);  R2 := R2 AND R3;  IF = THEN
26 0766  07 0F84  3573            BEGIN R0 := @REFERROR;  ERROR;   END;
26 0776  07 0F84  3574          END;
26 0776  07 0F84  3575          R0:=V1(R8+8);  COMMENT  SET PATH  AND NEW REG COUNT;
26 077A  07 0F84  3576          R2 := V1(R8);
26 077E  07 0F84  3577          IF R1 = 7 AND R2 ¬= V1(R7) THEN
26 078E  07 0F84  3578          BEGIN  R9 := V5(R8);  R3 := @UNEQ;
26 0796  07 0F84  3579            IF R0 = R3 THEN
26 079C  07 0F84  3580            BEGIN  R0 := @TRUE;  R1 := 8;
26 07A4  07 0F84  3581            END ELSE
26 07A4  07 0F84  3582            BEGIN  R0 := @FALSE;  R1 := 4;
26 07B0  07 0F84  3583            END;
26 07B0  07 0F84  3584            BOOLVALUE;  STC(R0,CONV(R9));
26 07C0  07 0F84  3585          END ELSE
26 07C0  07 0F84  3586          BEGIN  IF R1 = 7 THEN STC(R2,CONV(R9+4));
26 07D0  07 0F84  3587            REGPATH;  OUTOP;  R0 := 6;  V2(R8) := R0;
26 07F0  07 0F84  3588            END;
26 07F0  07 0F84  3589          END;
26 07F0  07 0F84  3590                                                                  102
26 07F0  07 0F84  3591          COMMENT    <SI T EXP>  ::= <SI T EXP*> <REL OP> <SI T EXP**>;
26 07F0  07 0F84  3592          BEGIN
26 07F4  07 0F84  3593            IC(R0,V22(R8));  IC(R1,V22(R7));
26 07FC  07 0F84  3594            IF R0 > 3 AND R0 ¬= 7 THEN
26 080C  07 0F84  3595            BEGIN  R0 := @TYPEERROR;  ERROR;
26 081C  07 0F84  3596            END ELSE
26 081C  07 0F84  3597            IF R1 > 3 AND R1 ¬= 7 THEN
26 0830  07 0F84  3598            BEGIN  R0 := @TYPEERROR;  ERROR;
26 0840  07 0F84  3599            END ELSE
26 0840  07 0F84  3600            IF R0 = 7 AND R1 ¬= 7 THEN
26 0854  07 0F84  3601            BEGIN  R0 := @TYPEERROR;  ERROR;
26 0864  07 0F84  3602            END ELSE
26 0864  07 0F84  3603            IF R1 = 7 AND R0 ¬= 7 THEN
26 0878  07 0F84  3604            BEGIN  R0 := @TYPEERROR;  ERROR;
26 0888  07 0F84  3605            END ELSE
26 0888  07 0F84  3606            IF R1 < R0 THEN  STC(R0,CONV(R9)  )ELSE
26 0896  07 0F84  3607            IF R0 < R1 THEN  BEGIN  R2 := V5(R8);  STC( R1,CONV(R2));
26 08A8  07 0F84  3608                             END;
26 08A8  07 0F84  3609            R0:= V1(R8+8);
26 08AC  07 0F84  3610            IF R1 = 7 THEN
26 08B4  07 0F84  3611            BEGIN  R1 := V1(R8);  R2 := V1(R7);
26 08BC  07 0F84  3612              IF R1 = R2 THEN STC(R1,CONV(R9+4)) ELSE
26 08C6  07 0F84  3613              IF R1 < R2 THEN
26 08D0  07 0F84  3614              BEGIN  STC(R1,CONV(R9+4));  R3 := @LESS;
26 08D8  07 0F84  3615                IF R0 = R3 THEN R0 := @LESSEQ ELSE
26 08E2  07 0F84  3616                BEGIN  R3 := @GTEQ;  IF R0 = R3 THEN R0 := @GREATER;
26 08F4  07 0F84  3617                END;
26 08F4  07 0F84  3618              END ELSE
```

```
26 08F4  07 0F84  3619            BEGIN  STC(R2,CONV(R9+4));  R3 := @LESSEQ;
26 09C0  07 0F84  3620              IF RO = R3 THEN RO := @LESS ELSE
26 090A  07 0F84  3621              BEGIN  R3 := @GREATER;  IF RO = R3 THEN RO := @GTEQ;
26 091C  07 0F84  3622              END;
26 091C  07 0F84  3623             END;
26 091C  07 0F84  3624           END;
26 091C  07 0F84  3625           REGPATH;  OUTOP;  RO := 6;
26 C938  07 0F84  3626           STC(RO,V22(R8));        COMMENT  RESULT IS TYPE LOGICAL;
26 093C  07 0F84  3627         END;
26 C93C  07 0F84  3628                                                                    103
26 093C  07 0F84  3629         COMMENT    <SI T EXP>   ::= <SI T EXP*>  IS   <RC CL ID> ;
26 093C  07 0F84  3630         BEGIN
26 C940  07 0F84  3631           IC(RO,V22(R8));
26 C944  07 0F84  3632           IF RO ¬= 9 THEN  BEGIN  RO := @TYPEERROR;   ERROR;
26 095C  07 0F84  3633                            END;
26 C95C  07 0F84  3634           RO := @IS;     OUTOP;
26 096C  07 0F84  3635           RO := 6;   STC(RO,V22(R8)); COMMENT  SIMPLE TYPE  IS LOG;
26 0974  07 0F84  3636         END;
26 0974  07 0F84  3637                                                                    104
26 0974  07 0F84  3638         COMMENT    <SI T EXP**>  ::= <SI T EXP*>;
26 C974  07 0F84  3639           NULL;
26 0978  07 0F84  3640                                                                    105
26 C978  07 0F84  3641         COMMENT    <SI T EXP*>  ::= <T TERM>  ;
26 0978  07 0F84  3642         NULL;
26 C97C  07 0F84  3643                                                                    106
26 C97C  07 0F84  3644         COMMENT    <SI T EXP*>  := +  <T TERM> ;
26 C97C  07 0F84  3645         BEGIN
26 0980  07 0F84  3646           IC(RO,V22(R7));
26 0984  07 0F84  3647           IF RO>5 THEN  BEGIN RO := @TYPEERROR; ERROR; END;
26 C99C  07 0F84  3648           FO1:=V(R7);   V(R8):=FO1;  COMMENT SAVE STACK INFO;
26 09A4  07 0F84  3649         END;
26 09A4  07 0F84  3650                                                                    107
26 09A4  07 0F84  3651         COMMENT    <SI T EXP*>  ::= -  <T TERM>  ;
26 09A4  07 0F84  3652         BEGIN
26 09A8  07 0F84  3653           IC(RO,V22(R7));
26 09AC  07 0F84  3654           IF RO>5 THEN BEGIN RO := @TYPEERROR; ERROR; END;
26 09C4  07 0F84  3655           FO1:=V(R7);   V(R8):=FO1;
26 09CC  07 0F84  3656           RO := @UMINUS;  R9 := @B9(4);  STC(RO,OP(R9)); V5(R8):=R9;
26 09DC  07 0F84  3657         END;
26 09DC  07 0F84  3658                                                                    108
26 09DC  07 0F84  3659         COMMENT    <SI T EXP*>  ::= <SI T EXP*> + <T TERM>;
26 09DC  07 0F84  3660         BEGIN
26 09E0  07 0F84  3661           IC(R1,V22(R8));  R2 := V5(R8);
26 09E8  07 0F84  3662           ARITHCHECK;  CONVERT1;
26 0A00  07 0F84  3663           RO := @PLUS;
26 0A04  07 0F84  3664           REGPATH;  OUTOP;
26 0A1C  07 0F84  3665         END;
26 0A1C  07 0F84  3666                                                                    109
26 0A1C  07 0F84  3667         COMMENT    <SI T EXP*>  ::= <SI T EXP*> - <T TERM>;
26 0A1C  07 0F84  3668         BEGIN
26 0A20  07 0F84  3669           IC(R1,V22(R8));  R2 := V5(R8);
26 0A28  07 0F84  3670           ARITHCHECK;  CONVERT1;
26 CA40  07 0F84  3671           RO := @MINUS;
26 0A44  07 0F84  3672           REGPATH;  OUTOP;
26 0A5C  07 0F84  3673         END;
26 0A5C  07 0F84  3674                                                                    110
26 0A5C  07 0F84  3675         COMMENT    <SI T EXP*>  ::= <SI T EXP*> OR   <T TERM>;
26 0A5C  07 0F84  3676         BEGIN
```

```
26 0A60  07 0F84  3677          IC(R0,V22(R8));
26 0A64  07 0F84  3678          IC(R1,V22(R7));   IF R0 ¬= R1 THEN
26 0A6E  07 0F84  3679            BEGIN  R0 := @TYPEERROR;    ERROR;
26 0A7E  07 0F84  3680            END;
26 0A7E  07 0F84  3681          IF R0 = 6 THEN
26 0A86  07 0F84  3682          BEGIN  F01 := V(R7);  V(R8+8) := F01; MAXREG; R0 := @LOGOR;
26 0A9E  07 0F84  3683          END ELSE
26 0A9E  07 0F84  3684          IF R0 = 8 THEN
26 0AAA  07 0F84  3685          BEGIN  R0 := @BITOR;  REGPATH;
26 0ABA  07 0F84  3686          END ELSE
26 0ABA  07 0F84  3687          BEGIN  R0 := @TYPEERROR;  ERROR;  R0 := 0;
26 0AD2  07 0F84  3688          END;
26 0AD2  07 0F84  3689          OUTOP;
26 0ADE  07 0F84  3690        END;
26 0ADE  07 0F84  3691                                                              111
26 0ADE  07 0F84  3692        COMMENT    <SI T EXP*>  ::=  <RC CL ID> ;
26 0ADE  07 0F84  3693        BEGIN
26 0AE2  07 0F84  3694          IC(R1,V22(R8));  R2 := 1;  R2 := R2 SHLL R1 SHRL 1;
26 0AF2  07 0F84  3695          V1(R8) := R2;
26 0AF6  07 0F84  3696          R0 := 9;  V2(R8) := R0;   COMMENT  SIMPLE TYPE := REF;
26 0AFE  07 0F84  3697        END;
26 0AFE  07 0F84  3698                                                              112
26 0AFE  07 0F84  3699        COMMENT    <SI T EXP*>  ::=  <RC DES HD>   <T EXP>  );
26 0AFE  07 0F84  3700        BEGIN
26 0B02  07 0F84  3701          IC(R0,V21(R8));          COMMENT  GET NUMBER OF FIELDS LEFT;
26 0B06  07 0F84  3702          IF R0 ¬= 1 THEN  BEGIN  R0 := @FIELDERROR;   ERROR;
26 0B1E  07 0F84  3703                           END;
26 0B1E  07 0F84  3704          R2 := V1(R8);             COMMENT  POINTER TO FIELDS;
26 0B22  07 0F84  3705          IC(R0,SIMPLETYPE(R2));
26 0B26  07 0F84  3706          R4 := SIMTYPEINFO(R2);
26 0B2A  07 0F84  3707          CNVRTASSN;                COMMENT  TYPE CHECK AND CONVERSION;
26 0B36  07 0F84  3708          R0 := @RPAREN;    OUTOP;
26 0B46  07 0F84  3709          IC(R0,V22(R8+8));
26 0B4A  07 0F84  3710          IF R0 = 7 THEN
26 0B52  07 0F84  3711          BEGIN  R1 := V1(R8+8);  STC(R1,CONV(R9));
26 0B5A  07 0F84  3712          END;     COMMENT  OUTPUT STRING LENGTH;
26 0B5A  07 0F84  3713          R1 := 0;  IC(R1,V22(R8));  COMMENT  MOVE RCCLNO;
26 0B62  07 0F84  3714          R2 := 1;  R2 := R2 SHLL R1 SHRL 1;  V1(R8) := R2;
26 0B72  07 0F84  3715          MAXREG;  IC(R0,V3(R8));
26 0B82  07 0F84  3716          IF R0 = 0 THEN
26 0B88  07 0F84  3717          BEGIN  R0 := 1;  STC(R0,V3(R8));
26 0B90  07 0F84  3718          END;
26 0B90  07 0F84  3719          R0 := 9;  V2(R8) := R0;   COMMENT  SIMPLE TYPE := REF;
26 0B98  07 0F84  3720        END;
26 0B98  07 0F84  3721                                                              113
26 0B98  07 0F84  3722        COMMENT    <SI T EXP*>  ::=  <STRING>   ;
26 0B98  07 0F84  3723        BEGIN
26 0B9C  07 0F84  3724          R0 := @STRINGG;   OUTID;      R0 := 0;  V34(R8) := R0;
26 0BB4  07 0F84  3725          IC(R0,STRINGLENGTH);   V1(R8) := R0;
26 0BBC  07 0F84  3726        END;
26 0BBC  07 0F84  3727                                                              114
26 0BBC  07 0F84  3728        COMMENT    <SI T EXP*>  ::= NULL;
26 0BBC  07 0F84  3729        BEGIN
26 0BC0  07 0F84  3730          R9 := @B9(4);
26 0BC4  07 0F84  3731          R0 := @NUL;  STC(R0,OP(R9));
26 0BCC  07 0F84  3732          R0 := 9;   STC(R0,V22(R8));  COMMENT   SIMPLE TYPE := REF;
26 0BD4  07 0F84  3733          V5(R8) := R9;      R0 := 0;  V34(R8) := R0;
26 0BE0  07 0F84  3734          R1 := #FFFF;  V1(R8) := R1;  COMMENT  SET RC CL MASK;
```

```
26 0BE8  07 0F84  3735          END;
26 0BE8  07 0F84  3736                                                                  115
26 0BE8  07 0F84  3737          COMMENT    <T TERM>  ::= <T TERM*>;
26 0BE8  07 0F84  3738          NULL;
26 0BEC  07 0F84  3739                                                                  116
26 0BEC  07 0F84  3740          COMMENT    <T TERM*> ::= <T FACT>;
26 0BEC  07 0F84  3741          NULL;
26 0BF0  07 0F84  3742                                                                  117
26 0BF0  07 0F84  3743          COMMENT    <T TERM*> ::= <T TERM*>   *  <T FACT> ;
26 0BF0  07 0F84  3744          BEGIN
26 0BF4  07 0F84  3745              IC(R1,V22(R8));   ARITHCHECK;
26 0C04  07 0F84  3746            COMMENT  CHECK TYPES.  FIRST ARG IN R1, SECOND IN R0;
26 0C04  07 0F84  3747            IF R0 ¬= R1 THEN
26 0C0A  07 0F84  3748              BEGIN   R2 := V5(R8);  CASE R1 OF
26 0C0E  07 0F84  3749                BEGIN
26 0C0E  07 0F84  3750                  BEGIN  STC(R0,CONV(R2));
26 0C1A  07 0F84  3751                    IF R0 = 2 THEN R0 := 3 ELSE IF R0 = 4 THEN R0 := 5;
26 0C36  07 0F84  3752                    STC(R0,V22(R8));
26 0C3A  07 0F84  3753                  END;
26 0C3A  07 0F84  3754                  BEGIN   IF R0 >1 THEN STC(R0,CONV(R2) )ELSE
26 0C4A  07 0F84  3755                    BEGIN  R1 := 2;  STC(R1,CONV(R9));
26 0C56  07 0F84  3756                    END;
26 0C56  07 0F84  3757                    IF R0 >3 THEN R0 := 5 ELSE R0 := 3;  STC(R0,V22(R8));
26 0C6E  07 0F84  3758                  END;
26 0C6E  07 0F84  3759                  IF R0 < 3 THEN
26 0C7A  07 0F84  3760                    BEGIN  R0 := 3;  STC(R0,CONV(R9));
26 0C82  07 0F84  3761                    END  ELSE IF R0 = 4 THEN
26 0C8E  07 0F84  3762                    BEGIN  R0 := 5;  STC(R0,CONV(R2));
26 0C96  07 0F84  3763                      STC(R0,CONV(R9));  STC(R0,V22(R8));
26 0C9E  07 0F84  3764                    END  ELSE
26 0C9E  07 0F84  3765                    BEGIN  STC(R0,CONV(R2));  STC(R0,V22(R8));
26 0CAA  07 0F84  3766                    END;
26 0CAA  07 0F84  3767                  BEGIN  IF R0 = 3 THEN
26 0CB6  07 0F84  3768                    BEGIN  R0 := 5;  STC(R0,CONV(R9)); STC(R0,CONV(R2));
26 0CC2  07 0F84  3769                    END  ELSE IF R0 = 5 THEN STC(R0,CONV(R2) )ELSE
26 0CD2  07 0F84  3770                    BEGIN  R0 := 4;  STC(R0,CONV(R9));  R0 := 5;
26 0CE2  07 0F84  3771                    END;
26 0CE2  07 0F84  3772                    STC(R0,V22(R8));
26 0CE6  07 0F84  3773                  END;
26 0CE6  07 0F84  3774                  BEGIN  R0 := 5;  STC(R0,CONV(R9));
26 0CF2  07 0F84  3775                  END;
26 0CF2  07 0F84  3776                END;
26 0D0A  07 0F84  3777              END  ELSE IF R0 = 2 THEN
26 0D16  07 0F84  3778              BEGIN  R0 := 3;  STC(R0,V22(R8));
26 0D1E  07 0F84  3779              END  ELSE IF R0 = 4 THEN
26 0D2A  07 0F84  3780              BEGIN  R0 := 5;  STC(R0,V22(R8));
26 0D32  07 0F84  3781              END;
26 0D32  07 0F84  3782            R0 := @TIMES;
26 0D36  07 0F84  3783            REGPATH;  OUTOP;
26 0D4E  07 0F84  3784          END;
26 0D4E  07 0F84  3785                                                                  118
26 0D4E  07 0F84  3786          COMMENT    <T TERM*> ::= <T TERM*>  /   <T FACT> ;
26 0D4E  07 0F84  3787          BEGIN
26 0D52  07 0F84  3788            IC(R1,V22(R8));  R2 := V5(R8);
26 0D5A  07 0F84  3789            ARITHCHECK;  CONVERT1;
26 0D72  07 0F84  3790            IF R0 = 1 THEN IF R1 = 1 THEN
26 0D82  07 0F84  3791              BEGIN  R0 := 3;  STC(R0,V22 (R8));
26 0D8A  07 0F84  3792                STC(R0,CONV(R9));   STC(R0,CONV(R2));
```

```
26 0D92  07 0F84   3793            END;
26 0D92  07 0F84   3794          R0 := @DIVIDE;
26 0D96  07 0F84   3795          REGPATH;  OUTOP;
26 0DAE  07 0F84   3796        END;
26 0DAE  07 0F84   3797    END;   COMMENT END OF CASE STATEMENT;
26 0E7A  07 0F84   3798    R10 := RASAVE;
26 0E7E  07 0F84   3799    END;    COMMENT END OF LEVELTWO;

                  SEGMENT 26  NAME = SEG#26      LENGTH = 0ED0  BASE REG = 15

25 045A  07 0F84   3800    LEVELTWO;
25 0464  07 0F84   3801  END;  COMMENT END OF EXECUTE2;

                  SEGMENT 25  NAME = SEG#25      LENGTH = 04A8  BASE REG = 15

23 0474  07 0F84   3802
23 0474  07 0F84   3803   SEGMENT PROCEDURE   EXECUTE3(R6);
23 0474  07 0F84   3804  BEGIN
27 0000  07 0F84   3805
27 0000  07 0F84   3806  PROCEDURE IFTHENSTATEMENT(R5);
27 0000  07 0F84   3807        BEGIN
27 0004  07 0F84   3808          R1 := V5(R8+8);  R2 := V5(R8) - OUTBASE;  POINTER(R1) := R2;
27 0014  07 0F84   3809          R9 := @B9(4);  R0 := @IFST;  STC(R0,OP(R9));
27 0020  07 0F84   3810          R1 := R1 - OUTBASE;
27 0024  07 0F84   3811          POINTER(R9):=R1;
27 0028  07 0F84   3812          V5(R8):=R9;
27 002C  07 0F84   3813          R0 := 0;  V34(R8) := R0;
27 0034  07 0F84   3814        END;
27 0036  07 0F84   3815
27 0036  07 0F84   3816  PROCEDURE FORSTATEMENT(R5);
27 0036  07 0F84   3817        BEGIN
27 0036  07 0F84   3818          R1:= V1(R8);
27 003A  07 0F84   3819          IF R1 =0 THEN R0 := @ITERST ELSE R0 := @ITERST2;    OUTOP;
27 0058  07 0F84   3820          R1:=BNC;  R1:= BLOCKLIST2(R1);  BNC := R1;
27 0064  07 0F84   3821          R0 := 0;  V34(R8) := R0;
27 006C  07 0F84   3822        END;
27 006E  07 0F84   3823
27 006E  07 0F84   3824  PROCEDURE  ASSIGNMENT(R5);
27 006E  07 0F84   3825    BEGIN
27 006E  07 0F84   3826      R0 := V2(R8);  IF R0 > #FF THEN
27 007A  07 0F84   3827      BEGIN  R0 := @ARRAYERROR;  ERROR;  R0 := V2(R8) AND #FF;
27 0092  07 0F84   3828      END;
27 0092  07 0F84   3829      F01 := V(R7);  V(R8+8) := F01;  R4 := V1(R8);
27 009E  07 0F84   3830      CNVRTASSN;  REGPATH;
27 00B6  07 0F84   3831      R1 := 0;  IC(R1,V22(R7));
27 00BE  07 0F84   3832      IF R1 = 7 THEN
27 00C6  07 0F84   3833      BEGIN  R1 := V1(R7); STC(R1,CONV(R9+4));  R1 := V5(R8);
27 00D2  07 0F84   3834        R1 := @CONV(R1);  STC(R4,B1);
27 00DA  07 0F84   3835      END;
27 00DA  07 0F84   3836      OUTOP;
27 00E6  07 0F84   3837    END;
27 00E8  07 0F84   3838
27 00E8  07 0F84   3839  PROCEDURE CASESEQEND(R5);
27 00E8  07 0F84   3840    BEGIN
27 00E8  07 0F84   3841      R1 := PATCHPOINT - 2;  PATCHPOINT := R1;  R1 := PATCHLIST(R1);
27 00F8  07 0F84   3842      IC(R0,V21(R8));  R0 := R0 + 1;  POINTER(R1) := R0;
27 0104  07 0F84   3843      R0 := @CLL;  OUTOP;   R0 := 0;  V34(R8) := R0;
27 011C  07 0F84   3844    END;
```

```
27 C11E  07 0F84  3845
27 011E  07 0F84  3846  PROCEDURE SHIFT(R5);
27 011E  07 0F84  3847  COMMENT  OUTPUT  NODE IS IN R1, SIMPLE TYPE IS IN R0;
27 011E  07 0F84  3848    IF R0 ¬= 8 THEN
27 0126  07 0F84  3849    BEGIN  R0 := @TYPEERROR;  ERROR;
27 C136  07 0F84  3850    END ELSE
27 C136  07 0F84  3851    BEGIN
27 013A  07 0F84  3852      IC(R0,V22(R7));
27 013E  07 0F84  3853      IF R0 ¬= 1 THEN
27 0146  07 0F84  3854      BEGIN R0 := @TYPEERROR;  ERROR;
27 0156  07 0F84  3855      END;
27 0156  07 0F84  3856      R0 := R1;  REGPATH;  OUTOP;
27 0170  07 0F84  3857      IC(R0,V3(R8));  IF R0 < 2 THEN R0 := 2;  STC(R0,V3(R8));
27 0184  07 0F84  3858    END;
27 C186  07 0F84  3859
27 0186  07 0F84  3860  PROCEDURE RESULT(R5);
27 0186  07 0F84  3861  BEGIN
27 0186  07 0F84  3862    R1 := V34(R8);
27 018A  07 0F84  3863    IF R1 = 2 THEN
27 0192  07 0F84  3864    BEGIN  IC(R0,OP(R9));  R0 := R0 AND #7F;
27 C19A  07 0F84  3865      IF R0 ¬= 30 AND R0 ¬= 31 AND R0 ¬= 40 AND R0 ¬= 62 AND R0 ¬= 87
27 01BA  07 0F84  3866        AND R0 ¬= 89 THEN
27 01CA  07 0F84  3867      BEGIN  R0 := @RESULTERROR;  ERROR;
27 01DA  07 0F84  3868      END;
27 01DA  07 0F84  3869    END;
27 01DA  07 0F84  3870  END;
27 01DC  07 0F84  3871
27 01DC  07 0F84  3872   SEGMENT PROCEDURE  LEVELTWO(R10);
27 01DC  07 0F84  3873  BEGIN INTEGER RASAVE;    RASAVE := R10;
28 0004  07 0F88  3874       RULENUMBER := R5;
28 0008  07 0F88  3875
28 0008  07 0F88  3876     R5 := R5 - 118;  CASE R5 OF
28 000C  07 0F88  3877      BEGIN
28 000C  07 0F88  3878                                                              119
28 000C  07 0F88  3879        COMMENT    <T TERM*> ::= <T TERM*>  DIV  <T FACT> ;
28 000C  07 0F88  3880        BEGIN
28 0014  07 0F88  3881          IC(R0,V22(R8));  IC(R1,V22(R7));    TYPEINTEGER;
28 0028  07 0F88  3882          R0 := @DIV;
28 002C  07 0F88  3883          REGPATH;  OUTOP;
28 0044  07 0F88  3884          R2 := V34(R8);  IF R2 < 3 AND R3 ¬= 0 THEN R2 := 3 ELSE
28 005A  07 0F88  3885           IF R2 < 2 AND R3 = 0 THEN R2 := 2;    V34(R8) := R2;
28 0074  07 0F88  3886        END;
28 0074  07 0F88  3887                                                              120
28 0074  07 0F88  3888        COMMENT    <T TERM*> ::= <T TERM*> REM   <T FACT> ;
28 0074  07 0F88  3889        BEGIN
28 0078  07 0F88  3890          IC(R0,V22(R8));  IC(R1,V22(R7));    TYPEINTEGER;
28 008C  07 0F88  3891          R0 := @REM;
28 0090  07 0F88  3892          REGPATH;  OUTOP;
28 00A8  07 0F88  3893          R2 := V34(R8);  IF R2 < 3 AND R3 ¬= 0 THEN R2 := 3 ELSE
28 00BE  07 0F88  3894           IF R2 < 2 AND R3 = 0 THEN R2 := 2;    V34(R8) := R2;
28 00D8  07 0F88  3895        END;
28 00D8  07 0F88  3896                                                              121
28 00D8  07 0F88  3897        COMMENT    <T TERM*> ::= <T TERM*>  AND  <T FACT>  ;
28 00D8  07 0F88  3898        BEGIN
28 00DC  07 0F88  3899          IC(R0,V22(R8));  IC(R1,V22(R7));
28 00E4  07 0F88  3900          IF R0 ¬= R1 THEN
28 00EA  07 0F88  3901            BEGIN  R0 := @TYPEERROR;    ERROR;
28 00FA  07 0F88  3902            END;
```

```
28 00FA  07 0F88  3903          IF R0 = 6 THEN
28 0102  07 0F88  3904          BEGIN  F01 := V(R7);  V(R8+8) := F01; MAXREG; R0 := @LOGAND;
28 011A  07 0F88  3905          END ELSE
28 011A  07 0F88  3906          IF R0 = 8 THEN
28 0126  07 0F88  3907          BEGIN  R0 := @BITAND;  REGPATH;
28 0136  07 0F88  3908          END ELSE
28 0136  07 0F88  3909          BEGIN  R0 := @TYPEERROR;  ERROR;  R0 := 0;
28 014E  07 0F88  3910          END;
28 014E  07 0F88  3911          OUTOP;
28 015A  07 0F88  3912        END;
28 015A  07 0F88  3913                                                              122
28 015A  07 0F88  3914        COMMENT    <T FACT>   ::=  <T SECON>     ;
28 015A  07 0F88  3915        NULL;
28 015E  07 0F88  3916                                                              123
28 015E  07 0F88  3917        COMMENT    <T FACT>   ::=    ¬   <T FACT>;
28 015E  07 0F88  3918        BEGIN
28 0162  07 0F88  3919          IC(R0,V22(R7));
28 0166  07 0F88  3920          IF R0 = 6 THEN R0 := @LOGNOT  ELSE
28 C172  07 0F88  3921          IF R0 = 8 THEN R0 := @BITNOT ELSE
28 0182  07 0F88  3922          BEGIN  R0 := @TYPEERROR;  ERROR;
28 0196  07 0F88  3923          END;
28 0196  07 0F88  3924          R9 := @B9(4);  STC(R0,OP(R9));
28 019E  07 0F88  3925          F01:=V(R7);    V(R8):=F01;
28 01A6  07 0F88  3926          V5(R8):=R9;
28 01AA  07 0F88  3927        END;
28 01AA  07 0F88  3928                                                              124
28 01AA  07 0F88  3929        COMMENT    <T SECON>   ::=   <T PRIM>    ;
28 01AA  07 0F88  3930        NULL;
28 01AE  07 0F88  3931                                                              125
28 01AE  07 0F88  3932        COMMENT    <T SECON>   ::=  <T SECON> <SHL OR **> <T PRIM>  ;
28 01AE  07 0F88  3933        BEGIN
28 01B2  07 0F88  3934          IC(R0,V22(R8));
28 01B6  07 0F88  3935          IF R0<6 THEN
28 01BE  07 0F88  3936            BEGIN  IF R0 = 1 THEN
28 01C6  07 0F88  3937              BEGIN  R2 := V5(R8); R0 := 3;
28 01CE  07 0F88  3938                STC(R0,CONV(R2));  STC(R0,V22(R8));
28 01D6  07 0F88  3939              END;
28 01D6  07 0F88  3940              IC(R0,V22(R7));
28 01DA  07 0F88  3941              IF R0 ¬= 1 THEN BEGIN R0 := @TYPEERROR;ERROR;END;
28 01F2  07 0F88  3942              REGPATH;  R0 := @EXPON;  OUTOP;   COMMENT FORCE LEFT;
28 020E  07 0F88  3943            END     ELSE
28 020E  07 0F88  3944            BEGIN  R1 := @SHL;  SHIFT;
28 0222  07 0F88  3945            END;
28 0222  07 0F88  3946        END;
28 0222  07 0F88  3947                                                              126
28 0222  07 0F88  3948        COMMENT    <T SECON>   ::=  <T SECON>  SHR  <T PRIM>    ;
28 0222  07 0F88  3949        BEGIN
28 0226  07 0F88  3950          IC(R0,V22(R8));  R1 := @SHR;  SHIFT;
28 023A  07 0F88  3951        END;
28 023A  07 0F88  3952                                                              127
28 023A  07 0F88  3953        COMMENT    <T PRIM>  ::=  <T VAR>;
28 023A  07 0F88  3954        NULL;
28 023E  07 0F88  3955                                                              128
28 023E  07 0F88  3956        COMMENT    <T PRIM>  ::= <T FUNC DES> ;
28 023E  07 0F88  3957        BEGIN
28 0242  07 0F88  3958          IC(R0,V22(R7));
28 0246  07 0F88  3959          IF R0 =0 THEN SET(FLAG )ELSE
28 0250  07 0F88  3960          BEGIN
```

```
28 0254  07 0F88  3961            R0 := 8;  STC(R0,V4(R8));  R1 := HN-2;  STC(R1,V3(R8));
28 0268  07 0F88  3962            IC (R0,OP(R9));  R4 := @FUNCID;
28 0270  07 0F88  3963            IF R0 ¬= R4 THEN
28 0276  07 0F88  3964            BEGIN R1 := 0;  STC(R1,V21(R8));
28 027E  07 0F88  3965            END;
28 027E  07 0F88  3966          END;
28 027E  07 0F88  3967        END;
28 027E  07 0F88  3968                                                                    129
28 027E  07 0F88  3969        COMMENT    <T PRIM>   ::=   <ST FUNC ID>;
28 027E  07 0F88  3970        BEGIN
28 0282  07 0F88  3971          R8 := R8 SHRL 3;  IC(R0,S(R8-1));  R8 := R8 SHLL 3;
28 028E  07 0F88  3972          IF R0 ¬= APARHEAD THEN
28 0296  07 0F88  3973          BEGIN  R0 := @APARERROR;  ERROR;
28 02A6  07 0F88  3974          END;
28 02A6  07 0F88  3975          R1 := V1(R8);  R1 := SIMTYPEINFO(R1);  V1(R8) := R1;
28 02B2  07 0F88  3976        END;
28 02B2  07 0F88  3977                                                                    130
28 02B2  07 0F88  3978        COMMENT    <T PRIM>  ::=   <LEFT PAR>  <T EXP>   )  ;
28 02B2  07 0F88  3979        BEGIN
28 02B6  07 0F88  3980          R1 := V1(R8);
28 02BA  07 0F88  3981          IF R1 = 0 THEN  BEGIN   F01 := V(R8+8);   V(R8) := F01;
28 02C8  07 0F88  3982                          END   ELSE
28 02C8  07 0F88  3983             BEGIN
28 02CC  07 0F88  3984               R0 := TYPEINFO(R1);  COMMENT  GET SIMPLE TYPE OF FPAR;
28 02D0  07 0F88  3985               R4 := IDLOC1(R1);    COMMENT  GET SIMTYPEINFO OF FPAR;
28 02D4  07 0F88  3986               CNVRTASSN;  R0 := @APPAREN;  OUTOP;
28 02F0  07 0F88  3987               R1 := V1(R8);  R1 := SIMTYPEINFO(R1);  V1(R8) := R1;
28 02FC  07 0F88  3988             END;
28 02FC  07 0F88  3989        END;
28 02FC  07 0F88  3990                                                                    131
28 02FC  07 0F88  3991        COMMENT    <T PRIM>  ::=  TRUE;
28 02FC  07 0F88  3992        BEGIN
28 0300  07 0F88  3993          R0 := @TRUE;  R1 := 8;  BOOLVALUE;
28 0314  07 0F88  3994        END;
28 0314  07 0F88  3995                                                                    132
28 0314  07 0F88  3996        COMMENT    <T PRIM>  ::= FALSE ;
28 0314  07 0F88  3997        BEGIN
28 0318  07 0F88  3998          R0 := @FALSE; R1 := 4;  BOOLVALUE;
28 032C  07 0F88  3999        END;
28 032C  07 0F88  4000                                                                    133
28 032C  07 0F88  4001        COMMENT    <T PRIM>  ::=  <CON ID>;
28 032C  07 0F88  4002        NULL;
28 0330  07 0F88  4003                                                                    134
28 0330  07 0F88  4004        COMMENT    <T PRIM>  ::=  LONG <T PRIM> ;
28 0330  07 0F88  4005        BEGIN
28 0334  07 0F88  4006          F01:=V(R8+8);   V(R8):=F01;   COMMENT SAVE VALUE INFO;
28 033C  07 0F88  4007          IC(R0,V22(R8));
28 0340  07 0F88  4008          IF R0 = 1 OR R0 = 2 THEN R0 := 3 ELSE
28 0354  07 0F88  4009          IF R0 = 4 THEN R0 := 5 ELSE
28 0364  07 0F88  4010          BEGIN R0 := @TYPEERROR;  ERROR;    END;
28 0378  07 0F88  4011          STC(R0,CONV(R9));  STC(R0,V22(R8));
28 0380  07 0F88  4012        END;
28 0380  07 0F88  4013                                                                    135
28 0380  07 0F88  4014        COMMENT    <T PRIM>  ::= SHORT <T PRIM>;
28 0380  07 0F88  4015        BEGIN
28 0384  07 0F88  4016          F01 := V(R8+8);  V(R8) := F01;  COMMENT SAVE VALUE INFO;
28 038C  07 0F88  4017          IC(R0,V22(R8));
28 0390  07 0F88  4018          IF R0 = 3 THEN R0 := 2 ELSE
```

```
28 039C  07 0F88  4019          IF R0 = 5 THEN R0 := 4 ELSE
28 03AC  07 0F88  4020          BEGIN  R0 := @TYPEERROR;  ERROR;  END;
28 03C0  07 0F88  4021          STC(R0,CONV(R9));  STC(R0,V22(R8));
28 03C8  07 0F88  4022        END;
28 03C8  07 0F88  4023                                                          136
28 03C8  07 0F88  4024        COMMENT    <T PRIM>  ::=  ABS <T PRIM> ;
28 03C8  07 0F88  4025        BEGIN
28 03CC  07 0F88  4026          IC(R0,V22(R7));
28 03D0  07 0F88  4027          IF R0 = 1 THEN
28 03D8  07 0F88  4028          BEGIN  IC(R1,V3(R8));  IF R1 = 0 THEN STC(R0,V3(R8));
28 03E6  07 0F88  4029          END  ELSE IF R0 < 4 THEN
28 03F2  07 0F88  4030          BEGIN  IC(R1,V4(R8));  IF R1 = 0 THEN R1 := 1;
28 0400  07 0F88  4031            STC(R1,V4(R8));
28 0404  07 0F88  4032          END  ELSE
28 0404  07 0F88  4033          BEGIN  IC(R1,V3(R8));  IF R1 < 4 THEN R1 := 4;
28 0418  07 0F88  4034          STC(R1,V3(R8));  IC(R1,V4(R8));  IF R1 < 4 THEN R1 := 4;
28 042C  07 0F88  4035            STC(R1,V4(R8));
28 0430  07 0F88  4036          END;
28 0430  07 0F88  4037          IF R0 > 5 THEN
28 0438  07 0F88  4038          BEGIN  R0 := @TYPEERROR;  ERROR;
28 0448  07 0F88  4039          END  ELSE  IF R0 = 4 THEN
28 0454  07 0F88  4040          BEGIN  R0 := 2;  STC(R0,V22(R7));
28 045C  07 0F88  4041          END  ELSE IF R0 = 5 THEN
28 0468  07 0F88  4042          BEGIN  R0 := 3;  STC(R0,V22(R7));
28 0470  07 0F88  4043          END;
28 0470  07 0F88  4044          R9 := @B9(4);
28 0474  07 0F88  4045          R0 := @ABSS;   STC(R0,OP(R9));
28 047C  07 0F88  4046          F01:=V(R7);  V(R8):=F01;
28 0484  07 0F88  4047          V5(R8):=R9;
28 0488  07 0F88  4048        END;
28 0488  07 0F88  4049                                                          137
28 0488  07 0F88  4050        COMMENT    <T PRIM>  ::=  <T NUMBER> ;
28 0488  07 0F88  4051        BEGIN
28 048C  07 0F88  4052          R4 := @NUMVALUE;  IC(R0,V22(R8));   LITERAL;  V1(R8) := R5;
28 04A4  07 0F88  4053          R0 := @NUMBER; OUTID;
28 04B4  07 0F88  4054          R0 := 0;  V34(R8) := R0;
28 04BC  07 0F88  4055        END;
28 04BC  07 0F88  4056                                                          138
28 04BC  07 0F88  4057        COMMENT    <T PRIM>  ::=  <BIT SEQ> ;
28 04BC  07 0F88  4058        BEGIN
28 04C0  07 0F88  4059          R0 := @BITT;   OUTID;        R0 := 0;  V34(R8) := R0;
28 04D8  07 0F88  4060        END;
28 04D8  07 0F88  4061                                                          139
28 04D8  07 0F88  4062        COMMENT    <REL OP>  ::=  <  ;
28 04D8  07 0F88  4063        BEGIN
28 04DC  07 0F88  4064          R0 := @LESS;   V1(R8):=R0;
28 04E4  07 0F88  4065        END;
28 04E4  07 0F88  4066                                                          140
28 04E4  07 0F88  4067        COMMENT    <REL OP>  ::=  <  = ;
28 04E4  07 0F88  4068        BEGIN
28 04E8  07 0F88  4069          R0 := @LESSEQ; V1(R8):=R0;
28 04F0  07 0F88  4070        END;
28 04F0  07 0F88  4071                                                          141
28 04F0  07 0F88  4072        COMMENT    <REL OP>  ::=  >  ;
28 04F0  07 0F88  4073        BEGIN
28 04F4  07 0F88  4074          R0 := @GREATER;V1(R8):=R0;
28 04FC  07 0F88  4075        END;
28 04FC  07 0F88  4076                                                          142
```

```
28 04FC  07 0F88  4077          COMMENT    <REL OP>  ::=  >  = ;
28 04FC  07 0F88  4078          BEGIN
28 0500  07 0F88  4079            R0 := @GTEQ;V1(R8):=R0;
28 0508  07 0F88  4080          END;
28 0508  07 0F88  4081                                                              143
28 0508  07 0F88  4082          COMMENT    <EQL OP>  ::=  = ;
28 0508  07 0F88  4083          BEGIN
28 050C  07 0F88  4084            R0 := @EQUAL;  V1(R8):=R0;
28 0514  07 0F88  4085          END;
28 0514  07 0F88  4086                                                              144
28 0514  07 0F88  4087          COMMENT    <EQL OP>  ::=  NOT = ;
28 0514  07 0F88  4088          BEGIN
28 0518  07 0F88  4089            R0 := @UNEQ;   V1(R8):=R0;
28 0520  07 0F88  4090          END;
28 0520  07 0F88  4091                                                              145
28 0520  07 0F88  4092          COMMENT    <LEFT PAR>   ::=    (;
28 0520  07 0F88  4093          BEGIN
28 0524  07 0F88  4094            R0 := 0;    V1(R8) := R0;
28 052C  07 0F88  4095          END;
28 052C  07 0F88  4096                                                              146
28 052C  07 0F88  4097          COMMENT    <LEFT PAR>  ::=   <ST FUNC ID>  (  ;
28 052C  07 0F88  4098          BEGIN
28 0530  07 0F88  4099            R0 := 0;     STC(R0,V21(R8)); COMMENT  TYPE SET TO ZERO;
28 0538  07 0F88  4100          END;
28 0538  07 0F88  4101                                                              147
28 0538  07 0F88  4102          COMMENT    <RC DES HD>  ::=  <RC CL ID>  (;
28 0538  07 0F88  4103          BEGIN
28 053C  07 0F88  4104            R2 := V1(R8);
28 0540  07 0F88  4105            IC(R1,VR(R2));  COMMENT GET NUMBER OF FIELDS;
28 0544  07 0F88  4106            IF R1 = 0 THEN
28 054A  07 0F88  4107            BEGIN  R2 := R2 + 12;  IC(R0,TYPE(R2));
28 0552  07 0F88  4108              WHILE R0 = 5 DO
28 055A  07 0F88  4109              BEGIN  R2 := R2 + 12;  R1 := R1 + 1;  IC(R0,TYPE(R2));
28 0566  07 0F88  4110              END;
28 056A  07 0F88  4111              R2 := V1(R8);   STC(R1,VR(R2));
28 0572  07 0F88  4112            END;
28 0572  07 0F88  4113            STC(R1,V21(R8));
28 0576  07 0F88  4114            R2 := R2 + 12;
28 057A  07 0F88  4115            V1(R8) := R2;            COMMENT POINTER TO FIELDS IN V1(J);
28 057E  07 0F88  4116          END;
28 057E  07 0F88  4117                                                              148
28 057E  07 0F88  4118          COMMENT    <RC DES HD>  ::=  <RC DES HD>  <T EXP>  ,  ;
28 057E  07 0F88  4119          BEGIN
28 0582  07 0F88  4120            IC(R0,V21(R8));          COMMENT  GET NUMBER OF FIELDS LEFT;
28 0586  07 0F88  4121            IF R0 = 1 THEN  BEGIN  R0 := @FIELDERROR;   ERROR;
28 059E  07 0F88  4122                            END;
28 059E  07 0F88  4123            R0 := R0 - 1;   STC(R0,V21(R8));  COMMENT DECREASE FIELD CNT;
28 05A6  07 0F88  4124            R2 := V1(R8);            COMMENT  POINTER TO FIELDS;
28 05AA  07 0F88  4125            R4 := SIMTYPEINFO(R2);
28 05AE  07 0F88  4126            IC(R0,SIMPLETYPE(R2));
28 05B2  07 0F88  4127            R2 := R2 + 12;     V1(R8) := R2;  COMMENT  STEP FIELD POINTER;
28 05BA  07 0F88  4128            CNVRTASSN;               COMMENT  TYPE CHECK AND CONVERSION;
28 05C6  07 0F88  4129            R0 := @RCOMMA;  OUTOP;
28 05D6  07 0F88  4130            IC(R0,V22(R8+8));
28 05DA  07 0F88  4131            IF R0 = 7 THEN
28 05E2  07 0F88  4132            BEGIN  R1 := V1(R8+8);  STC(R1,CONV(R9));
28 05EA  07 0F88  4133            END;     COMMENT  OUTPUT STRING LENGTH;
28 05EA  07 0F88  4134            MAXREG;              COMMENT  ADJUST REGISTER COUNTS;
```

```
28 05F6  07 0F88  4135          END;
28 05F6  07 0F88  4136                                                                  149
28 05F6  07 0F88  4137          COMMENT    <PROGRAM>   ::=   .   <BLOCK>   .     ;
28 05F6  07 0F88  4138            NULL;
28 05FA  07 0F88  4139            COMMENT   NEVER ENTERED.  SEE END OF PROGRAM;
28 05FA  07 0F88  4140                                                                  150
28 05FA  07 0F88  4141          COMMENT    <STATEMENT>  ::= <STATEMENT*>  ;
28 05FA  07 0F88  4142          NULL;
28 05FE  07 0F88  4143                                                                  149
28 05FE  07 0F88  4144                                                                  151
28 05FE  07 0F88  4145          COMMENT    <STATEMENT*> ::=<SI ST>  ;
28 05FE  07 0F88  4146          NULL;
28 0602  07 0F88  4147                                                                  152
28 0602  07 0F88  4148          COMMENT    <STATEMENT*> ::= <FOR CL> DO;
28 0602  07 0F88  4149          BEGIN
28 06C6  07 0F88  4150            R9 := R9 + 4;          COMMENT  STEP OUTPUT POINTER;
28 06CA  07 0F88  4151            R0 := @NULLST; STC(R0,OP(R9));
28 0612  07 0F88  4152            FORSTATEMENT;
28 061E  07 0F88  4153          END;
28 061E  07 0F88  4154                                                                  153
28 061E  07 0F88  4155          COMMENT    <STATEMENT*> ::= <FOR CL>  DO  <STATEMENT*>  ;
28 061E  07 0F88  4156          FORSTATEMENT;
28 062E  07 0F88  4157                                                                  154
28 062E  07 0F88  4158          COMMENT    <STATEMENT*> ::= <WHILE CL> DO;
28 062E  07 0F88  4159          BEGIN
28 0632  07 0F88  4160            R9 := R9 + 4;          COMMENT  STEP OUTPUT POINTER;
28 0636  07 0F88  4161            R0 := @NULLST;     STC(R0,OP(R9));
28 063E  07 0F88  4162            R0 := @WHILEST;  OUTOP;
28 064E  07 0F88  4163            R0 := 0;  V34(R8) := R0;
28 0656  07 0F88  4164          END;
28 0656  07 0F88  4165                                                                  155
28 0656  07 0F88  4166          COMMENT    <STATEMENT*> ::= <WHILE CL>  DO  <STATEMENT*>   ;
28 C656  07 0F88  4167          BEGIN
28 065A  07 0F88  4168            R0 := @WHILEST;  OUTOP;
28 066A  07 0F88  4169            R0 := 0;  V34(R8) := R0;
28 0672  07 0F88  4170          END;
28 0672  07 0F88  4171                                                                  156
28 0672  07 0F88  4172          COMMENT    <STATEMENT*> ::= <IF CL> ;
28 0672  07 0F88  4173          BEGIN
28 0676  07 0F88  4174            R9 := R9 + 4;          COMMENT  STEP OUTPUT POINTER;
28 067A  07 0F88  4175            R0 := @NULLST;     STC(R0,OP(R9));
28 0682  07 0F88  4176            R0 := @IFST;       OUTOP;
28 0692  07 0F88  4177            R0 := 0;  V34(R8) := R0;
28 069A  07 0F88  4178          END;
28 C69A  07 0F88  4179                                                                  157
28 069A  07 0F88  4180          COMMENT    <STATEMENT*> ::= <IF CL>  <STATEMENT*>  ;
28 C69A  07 0F88  4181          BEGIN
28 069E  07 0F88  4182            R0 := @IFST;   OUTOP;
28 C6AE  07 0F88  4183            R0 := 0;  V34(R8) := R0;
28 06B6  07 0F88  4184          END;
28 06B6  07 0F88  4185                                                                  158
28 06B6  07 0F88  4186          COMMENT    <STATEMENT*> ::=  <IF CL>  <TRUE PART>  ;
28 06B6  07 0F88  4187          BEGIN
28 06BA  07 0F88  4188            R9 := R9 + 4;          COMMENT  STEP OUTPUT POINTER;
28 06BE  07 0F88  4189            R0 := @NULLST;     STC(R0,OP(R9));
28 06C6  07 0F88  4190            IFTHENSTATEMENT;
28 06D2  07 0F88  4191          END;
28 06D2  07 0F88  4192                                                                  159
```

```
28 06D2  07 0F88  4193          COMMENT    <STATEMENT*> ::= <IF CL>  <TRUE PART> <STATEMENT*>;
28 06D2  07 0F88  4194          IFTHENSTATEMENT;
28 06E2  07 0F88  4195                                                                        160
28 06E2  07 0F88  4196          COMMENT    <STATEMENT*> ::= <CASE SEQ>   END    ;
28 06E2  07 0F88  4197          BEGIN
28 06E6  07 0F88  4198            R9 := @B9(4);  R0 := @NULLST; STC(R0,OP(R9));
28 06F2  07 0F88  4199            CASESEQEND;
28 06FE  07 0F88  4200          END;
28 06FE  07 0F88  4201                                                                        161
28 06FE  07 0F88  4202          COMMENT    <STATEMENT*> ::= <CASE SEQ>  <STATEMENT>    END ;
28 06FE  07 0F88  4203          CASESEQEND;
28 070E  07 0F88  4204                                                                        162
28 070E  07 0F88  4205          COMMENT    <SI ST>   ::=  <BLOCK> ;
28 070E  07 0F88  4206          NULL;
28 0712  07 0F88  4207                                                                        163
28 0712  07 0F88  4208          COMMENT    <SI ST>   ::=  <T ASS ST>;
28 0712  07 0F88  4209          BEGIN
28 0716  07 0F88  4210            R0 := 0;  V34(R8) := R0;
28 071E  07 0F88  4211          END;
28 071E  07 0F88  4212                                                                        164
28 071E  07 0F88  4213          COMMENT    <SI ST>   ::=  <T FUNC DES>;
28 071E  07 0F88  4214          BEGIN  R0 := 0;  V34(R8) := R0;
28 072A  07 0F88  4215          END;
28 072A  07 0F88  4216                                                                        165
28 072A  07 0F88  4217          COMMENT    <SI ST>   ::=  GOTO  <LABEL ID>  ;
28 072A  07 0F88  4218          BEGIN
28 072E  07 0F88  4219            R9 := @B9(4);  R0 := @GOTOO;STC(R0,OP(R9));
28 073A  07 0F88  4220            V5(R8):=R9;
28 073E  07 0F88  4221            R0 := 0;  V34(R8) := R0;
28 0746  07 0F88  4222          END;
28 0746  07 0F88  4223                                                                        166
28 0746  07 0F88  4224          COMMENT    <SI ST>  ::=  <ST PROC HD>  <T EXP>  );
28 0746  07 0F88  4225          BEGIN
28 074A  07 0F88  4226            RESULT;
28 0756  07 0F88  4227            R0 := @APPAREN;  OUTOP;
28 0766  07 0F88  4228            R0 := 0;  V34(R8) := R0;
28 076E  07 0F88  4229          END;
28 076E  07 0F88  4230                                                                        167
28 076E  07 0F88  4231          COMMENT    <BLOCK>   ::=  <BLOCKBODY>     END  ;
28 076E  07 0F88  4232          BEGIN
28 0772  07 0F88  4233            BLOCKEXIT;
28 077E  07 0F88  4234            R0 := @ENDD; OUTID;
28 078E  07 0F88  4235          END;
28 078E  07 0F88  4236                                                                        168
28 078E  07 0F88  4237          COMMENT    <BLOCK>  ::= <BLOCKBODY>  <STATEMENT>    END;
28 078E  07 0F88  4238          BEGIN
28 0792  07 0F88  4239            BLOCKEXIT;
28 079E  07 0F88  4240            R0 := @COMMA;    OUTOP;
28 07AE  07 0F88  4241            CARDOUT;
28 078A  07 0F88  4242            R0 := @ENDD;    OUTID;
28 07CA  07 0F88  4243          END;
28 07CA  07 0F88  4244                                                                        169
28 07CA  07 0F88  4245          COMMENT    <BLOCKBODY> ::= <BLOCKHEAD>    ;
28 07CA  07 0F88  4246          BEGIN
28 07CE  07 0F88  4247            R3 := V2(R8);
28 07D2  07 0F88  4248            IF R3 ¬= 0  THEN
28 07D8  07 0F88  4249            BEGIN  R0 := @BBB;    OUTID;  R3 := R9 - OUTBASE; V1(R8) := R3;
28 07F2  07 0F88  4250              R9 := R9 + 4;  R0 := @NULLST;  STC(R0,OP(R9));  CARDOUT;
```

```
28 080A  07 0F88  4251            R3 := BNC SHLL 1;  R2 := BLENGTH(R3);  R3 := NPOINT(R3);
28 081A  07 0F88  4252            R4 := DRELAD;  R2 := R2 + R3 -12;
28 0824  07 0F88  4253            FOR R3 := R3 STEP 12 UNTIL R2 DO
28 0824  07 0F88  4254            BEGIN  R5 := TYPES(R3);  IF R5 = #209 THEN
28 0834  07 0F88  4255               BEGIN  R4 := R4 + 3 SHRL 2 SHLL 2;
28 0840  07 0F88  4256                  R5 := IDLOC2(R3) + R4;  IDLOC2(R3) := R4;  R4 := R5;
28 084C  07 0F88  4257               END;
28 084C  07 0F88  4258            END;
28 0856  07 0F88  4259            DRELAD := R4;
28 085A  07 0F88  4260          END;
28 085A  07 0F88  4261        END;
28 085A  07 0F88  4262                                                                    170
28 085A  07 0F88  4263        COMMENT    <BLOCKBODY>  ::=  <BLOCKBODY>   .,     ;
28 085A  07 0F88  4264        NULL;
28 085E  07 0F88  4265                                                                    171
28 085E  07 0F88  4266        COMMENT    <BLOCKBODY> ::= <BLOCKBODY> <STATEMENT>  .,      ;
28 085E  07 0F88  4267        BEGIN
28 0862  07 0F88  4268          R0 := @COMMA;  OUTOP;
28 0872  07 0F88  4269          CARDOUT;
28 087E  07 0F88  4270        END;
28 087E  07 0F88  4271                                                                    172
28 087E  07 0F88  4272        COMMENT    <BLOCKBODY>  ::= <BLOCKBODY>  <LABEL DEF>   ;
28 087E  07 0F88  4273        BEGIN
28 0882  07 0F88  4274          R0 := @COMMA;  OUTOP;
28 0892  07 0F88  4275        END;
28 0892  07 0F88  4276                                                                    173
28 0892  07 0F88  4277        COMMENT    <BLOCKHEAD>  ::=  BEGIN  ;
28 0892  07 0F88  4278        BEGIN  INTEGER R1SAVE;
28 0896  07 0F8C  4279          R3 := DRELAD;  V34(R8) := R3;
28 089E  07 0F8C  4280          BLOCKSTEP;              COMMENT STEP BN, LEAVE NEW BNC IN R1;
28 08AA  07 0F8C  4281          R1 := R1 SHLA 1;           COMMENT  INDEX TO BLOCKLIST;
28 08AE  07 0F8C  4282          R2 := BLENGTH(R1);     COMMENT R2 CONTAINS LENGTH OF NAMETABLE;
28 08B2  07 0F8C  4283          R1 := NPOINT (R1);     COMMENT R1 POINTS TO NAMETABLE;
28 08B6  07 0F8C  4284          R8 := R8 SHRL 3;  IC(R0,S(R8-1)); R8 := R8 SHLL 3;
28 08C2  07 0F8C  4285          IF R2 ¬= 0 OR R0 = ENDFILE THEN
28 08D0  07 0F8C  4286            BEGIN    R3 := TYPES(R1);
28 08D4  07 0F8C  4287              IF R3 = #0100 AND R0 ¬= ENDFILE THEN
28 08E4  07 0F8C  4288              BEGIN R4 := 0;  V2(R8) := R4;  GOTO DONE;
28 08F0  07 0F8C  4289              END  ELSE
28 08F0  07 0F8C  4290              BEGIN
28 08F4  07 0F8C  4291                IF R0 ¬= TPROCHD THEN
28 08FC  07 0F8C  4292                  BEGIN  R3 := VARORIGIN+4;  VARORIGIN := R3;
28 0908  07 0F8C  4293                    DRELAD := R3;  R3 := HN-1;
28 0914  07 0F8C  4294                    HN := R3;
28 0918  07 0F8C  4295                    IF R3 < = 5 THEN  BEGIN  R0 := @HIERARCHYERROR;
28 0924  07 0F8C  4296                                             ERROR;
28 0930  07 0F8C  4297                                           END;
28 0930  07 0F8C  4298                    R3 := #FF;
28 0934  07 0F8C  4299                  END ELSE R3 := #F;
28 093C  07 0F8C  4300                V2(R8) := R3;
28 0940  07 0F8C  4301              END;
28 0940  07 0F8C  4302            END  ELSE
28 0940  07 0F8C  4303            BEGIN  R4 := 0;  V2(R8) := R4;  GOTO  DONE;
28 0950  07 0F8C  4304            END;
28 0950  07 0F8C  4305          R4 := DRELAD;
28 0954  07 0F8C  4306          WHILE  R2 > 0  DO
28 095A  07 0F8C  4307          BEGIN   IC(R0,SIMPLETYPE(R1));
28 095E  07 0F8C  4308            IF R0 = 9  THEN
```

```
28 0966  07 0F8C  4309              BEGIN  REFBIND;  IC(R0,TYPE(R1));
28 0976  07 0F8C  4310                IF R0 = 0 THEN
28 097C  07 0F8C  4311                BEGIN R4 := R4 + 3 AND #FFFC; IDLOC2(R1) := R4; R4 := R4+4;
28 0988  07 0F8C  4312                END;
28 098C  07 0F8C  4313              END;
28 098C  07 0F8C  4314              IC(R0,TYPE(R1));
28 0990  07 0F8C  4315              IF R0 = PROCEDURETYPE THEN
28 0998  07 0F8C  4316              BEGIN  R5 := TYPEINFO(R1) SHLL 2;
28 09A0  07 0F8C  4317                IF R5 ¬= 0 THEN
28 09A6  07 0F8C  4318                BEGIN  R1SAVE := R1;  R1 :=  NPOINT(R5); R5 := BLENGTH(R5);
28 09AE  07 0F8C  4319                  WHILE R5 > 0 DO
28 09B8  07 0F8C  4320                  BEGIN  IC(R0,SIMPLETYPE(R1));  IF R0 = 9 THEN REFBIND;
28 09D0  07 0F8C  4321                    R1 := R1 + 12;  R5 := R5 - 12;
28 09D8  07 0F8C  4322                  END;
28 09DC  07 0F8C  4323                  R1 := R1SAVE;
28 09E0  07 0F8C  4324                END;
28 09E0  07 0F8C  4325              END;
28 09E0  07 0F8C  4326              R1 := R1 + 12;   R2 := R2 - 12;
28 09E8  07 0F8C  4327            END;                    COMMENT  END OF WHILE STATEMENT;
28 09EC  07 0F8C  4328            DRELAD := R4;
28 09F0  07 0F8C  4329  DONE:     R9 := R9 + 4;   R0 := @BEGINN;
28 09F8  07 0F8C  4330            R1 := BNC SHLL 11; P(R9) := R1;
28 0A04  07 0F8C  4331            STC(R0,OP(R9));  CARDOUT;  R1 := R9 - OUTBASE;  V1(R8) := R1;
28 0A1E  07 0F8C  4332            R9 := R9 + 4;   R0 := @NULLST;
28 0A26  07 0F8C  4333            STC(R0,OP(R9));   V5(R8) := R9;
28 0A2E  07 0F8C  4334          END;
28 0A2E  07 0F8C  4335                                                                    174
28 0A2E  07 0F8C  4336          COMMENT    <BLOCKHEAD>   ::=  BEGIN  PARALLEL;
28 0A2E  07 0F8C  4337          NULL;  COMMENT  * * * * * * * * * * * *;
28 0A32  07 0F8C  4338                                                                    175
28 0A32  07 0F8C  4339          COMMENT    <BLOCKHEAD>  ::= <BLOCKHEAD> <SI VAR DC>  .,      ;
28 0A32  07 0F8C  4340          NULL;
28 0A36  07 0F8C  4341                                                                    176
28 0A36  07 0F8C  4342          COMMENT    <BLOCKHEAD>  ::=<BLOCKHEAD>  <ARRAY DC>   .,      ;
28 0A36  07 0F8C  4343          BEGIN
28 0A3A  07 0F8C  4344            R0 := @COMMA;  OUTOP;
28 0A4A  07 0F8C  4345          END;
28 0A4A  07 0F8C  4346                                                                    177
28 0A4A  07 0F8C  4347          COMMENT    <BLOCKHEAD>  ::=<BLOCKHEAD>  <PROC DECL>  .,      ;
28 0A4A  07 0F8C  4348          BEGIN
28 0A4E  07 0F8C  4349            R0 := @COMMA;  OUTOP;
28 0A5E  07 0F8C  4350          END;
28 0A5E  07 0F8C  4351                                                                    178
28 0A5E  07 0F8C  4352          COMMENT    <BLOCKHEAD>  ::=<BLOCKHEAD>  <RC CL DC>   .,      ;
28 0A5E  07 0F8C  4353          BEGIN
28 0A62  07 0F8C  4354            R0 := @COMMA;  OUTOP;
28 0A72  07 0F8C  4355          END;
28 0A72  07 0F8C  4356                                                                    179
28 0A72  07 0F8C  4357          COMMENT    <BLOCKHEAD>   ::=  <BLOCKHEAD>  <FILE DC>   ;
28 0A72  07 0F8C  4358          NULL;
28 0A76  07 0F8C  4359                                                                    180
28 0A76  07 0F8C  4360          COMMENT    <LABEL DEF>  ::=  <ID>   ::   ;
28 0A76  07 0F8C  4361          BEGIN
28 0A7A  07 0F8C  4362            R1 := V1(R8);  V1(R7) := R1;
28 0A82  07 0F8C  4363            DECLARETEST;  R0 := @ID;    OUTID;
28 0A9E  07 0F8C  4364            R2 := LABELADDR;  IDLOC2(R1) := R2;  R2 := R2 + 4;
28 0AAA  07 0F8C  4365            LABELADDR := R2;
28 0AAE  07 0F8C  4366            R2 := HN;  TYPEINFO(R1) := R2;
```

```
28 0AB6  07 0F8C  4367          RO := @LABELCOLON;
28 0ABA  07 0F8C  4368          R9 := @B9(4);  STC(RO,OP(R9));
28 0AC2  07 0F8C  4369          V5(R8):=R9;
28 0AC6  07 0F8C  4370          R2 := SNC;  IDLOC1(R1) := R2;
28 0ACE  07 0F8C  4371        END;
28 0ACE  07 0F8C  4372                                                                  181
28 0ACE  07 0F8C  4373        COMMENT  <T ASS ST>  ::= <T VAR> :=  <T EXP*>    ;
28 0ACE  07 0F8C  4374        ASSIGNMENT;
28 0ADE  07 0F8C  4375                                                                  182
28 0ADE  07 0F8C  4376        COMMENT  <T ASS ST>  ::= <T VAR> :=  <T ASS ST> ;
28 0ADE  07 0F8C  4377        BEGIN
28 0AE2  07 0F8C  4378          COMMENT  MULTIPLE ASSIGNMENTS FORCED TO RIGHT;
28 0AE2  07 0F8C  4379          IC(RO,OP(R9));  RO := RO OR #90;  STC(RO,OP(R9));
28 0AEE  07 0F8C  4380            ASSIGNMENT; IC(RO,OP(R9)); RO:=RO OR #80; STC(RO,OP(R9));
28 0BC6  07 0F8C  4381        END;
28 0BC6  07 0F8C  4382                                                                  183
28 0BC6  07 0F8C  4383        COMMENT    <TRUE PART>  ::= <SI ST>  ELSE  ;
28 0BC6  07 0F8C  4384        BEGIN
28 0BCA  07 0F8C  4385          RO := @UJ; R9 := @B9(4); STC(RO,OP(R9));  V5(R8):=R9;
28 0B1A  07 0F8C  4386        END;
28 0B1A  07 0F8C  4387                                                                  184
28 0B1A  07 0F8C  4388        COMMENT    <TRUE PART>  ::=    ELSE  ;
28 0B1A  07 0F8C  4389        BEGIN
28 0B1E  07 0F8C  4390          RO := @NULLST;  STC(RO,OP(R9+4));
28 0B26  07 0F8C  4391          RO := @UJ;  R9 := R9 + 8;    STC(RO,OP(R9));
28 0B32  07 0F8C  4392          V5(R8) := R9;
28 0B36  07 0F8C  4393        END;
28 0B36  07 0F8C  4394                                                                  185
28 0B36  07 0F8C  4395        COMMENT    <CASE SEQ>  ::= <CASE CL>  BEGIN    ;
28 0B36  07 0F8C  4396        BEGIN
28 0B3A  07 0F8C  4397          CARDOUT;
28 0B46  07 0F8C  4398          RO := 0; STC(RO,V21(R8));   COMMENT  INITIALIZE COUNT;
28 0B4E  07 0F8C  4399          R1 := BN + 2;  BN := R1;  COMMENT  STEP BN;
28 0B5A  07 0F8C  4400          R2 := BNC;  BLOCKLIST2(R1) := R2;  COMMENT  CHAIN IN BLOCKLST2;
28 0B62  07 0F8C  4401        END;
28 0B62  07 0F8C  4402                                                                  186
28 0B62  07 0F8C  4403        COMMENT    <CASE SEQ>  ::= <CASE SEQ> <STATEMENT>  .,    ;
28 0B62  07 0F8C  4404        BEGIN
28 0B66  07 0F8C  4405          IC(RO,V21(R8));  RO:=RO+1;  STC(RO,V21(R8));
28 0B72  07 0F8C  4406          RO := @UJ;  OUTOP;
28 0B82  07 0F8C  4407        END;
28 0B82  07 0F8C  4408                                                                  187
28 0B82  07 0F8C  4409        COMMENT    <CASE SEQ>  ::= <CASE SEQ>  .,     ;
28 0B82  07 0F8C  4410        BEGIN
28 0B86  07 0F8C  4411          IC(RO,V21(R8)); RO:=RO+1;  STC(RO,V21(R8));
28 0B92  07 0F8C  4412          R9 := @B9(4); RO := @NULLST; STC(RO,OP(R9));
28 0B9E  07 0F8C  4413          RO := @UJ;  OUTOP;
28 0BAE  07 0F8C  4414        END;
28 0BAE  07 0F8C  4415                                                                  188
28 0BAE  07 0F8C  4416        COMMENT    <FOR CL>  ::=  <FOR HEAD>  <STEPUNTIL>   <T EXP>     ;
28 0BAE  07 0F8C  4417        BEGIN
28 0BB2  07 0F8C  4418          IC(R1,V22(R7));
28 0BB6  07 0F8C  4419          IF R1 ¬= 1 THEN
28 0BBE  07 0F8C  4420          BEGIN  RO := @TYPEERROR;  ERROR;
28 0BCE  07 0F8C  4421          END;
28 0BCE  07 0F8C  4422          RO := @STEPUNTIL;  R9 := R9 + 4;
28 0BD6  07 0F8C  4423          STC(RO,OP(R9));  R1 := V5(R8+8) - OUTBASE; POINTER(R9) := R1;
28 0BE6  07 0F8C  4424          R1 := V1(R8); BNC := R1; RO := 0; V1(R8):=RO;
```

```
28 08F6  07 0F8C  4425          RO := @FORCL OR #80;  OUTOP;
28 0C0A  07 0F8C  4426        END;
28 0C0A  07 0F8C  4427                                                          189
28 0C0A  07 0F8C  4428        COMMENT     <FOR CL> ::=  <FOR HEAD>;
28 0C0A  07 0F8C  4429        BEGIN
28 0C0E  07 0F8C  4430          R1 := V1(R8); BNC := R1; RO := 1;  V1(R8) := RO;
28 0C1E  07 0F8C  4431          RO := @ENDFORLIST;  STC(RO,OP(R9));
28 0C26  07 0F8C  4432        END;
28 0C26  07 0F8C  4433                                                          190
28 0C26  07 0F8C  4434        COMMENT    <FOR CL>  ::=  <FOR LIST> <T EXP>   ;
28 0C26  07 0F8C  4435        BEGIN
28 0C2A  07 0F8C  4436          IC(RO,V22(R7));
28 0C2E  07 0F8C  4437          IF RO ¬=1 THEN BEGIN  RO := @TYPEERROR;   ERROR; END;
28 0C46  07 0F8C  4438          RO := @ENDFORLIST;  OUTOP;
28 0C56  07 0F8C  4439          R1 := V1(R8); BNC := R1;
28 0C5E  07 0F8C  4440        END;
28 0C5E  07 0F8C  4441                                                          191
28 0C5E  07 0F8C  4442        COMMENT    <FOR HEAD>   ::=   <FOR> :=  <T EXP*>  ;
28 0C5E  07 0F8C  4443        BEGIN
28 0C62  07 0F8C  4444          IC(RO,V22(R7));
28 0C66  07 0F8C  4445          IF RO ¬= 1 THEN  BEGIN  RO := @TYPEERROR;  ERROR;
28 0C7E  07 0F8C  4446                           END;
28 0C7E  07 0F8C  4447          RO := @ACOLONEQ;  REGPATH;
28 0C8E  07 0F8C  4448          OUTOP;
28 0C9A  07 0F8C  4449        END;
28 0C9A  07 0F8C  4450                                                          192
28 0C9A  07 0F8C  4451        COMMENT    <FOR LIST>  ::= <FOR HEAD>  ,    ;
28 0C9A  07 0F8C  4452        BEGIN
28 0C9E  07 0F8C  4453          RO := @FORLIST; STC(RO,OP(R9));
28 0CA6  07 0F8C  4454        END;
28 0CA6  07 0F8C  4455                                                          193
28 0CA6  07 0F8C  4456        COMMENT    <FOR LIST>   ::=  <FOR LIST>  <T EXP>  ,  ;
28 0CA6  07 0F8C  4457        BEGIN
28 0CAA  07 0F8C  4458          IC(RO,V22(R8+8));
28 0CAE  07 0F8C  4459          IF RO ¬= 1  THEN BEGIN RO := @TYPEERROR; ERROR;  END;
28 0CC6  07 0F8C  4460          RO := @FORLIST;  OUTOP;
28 0CD6  07 0F8C  4461        END;
28 0CD6  07 0F8C  4462                                                          194
28 0CD6  07 0F8C  4463        COMMENT    <FOR>     ::=   FOR  <ID> ;
28 0CD6  07 0F8C  4464        BEGIN
28 0CDA  07 0F8C  4465          BLOCKSTEP; DECLARETEST;
28 0CF2  07 0F8C  4466          R1 := V1(R8);  R4 := DRELAD + 3 SHRL 2 SHLL 2;
28 0D06  07 0F8C  4467          IDLOC2(R1) := R4;  R4 := R4 + 4;  DRELAD := R4;
28 0D12  07 0F8C  4468          R4 := HN;  IDLOC1(R1) := R4;
28 0D1A  07 0F8C  4469          RO := @CONID;  OUTID;
28 0D2A  07 0F8C  4470          RO := 0;  V34(R8) := RO;  COMMENT  SET REGISTER COUNTS;
28 0D32  07 0F8C  4471          R1 := BNC;  V1(R8) := R1;  R1 := BLOCKLIST2(R1);  BNC := R1;
28 0D42  07 0F8C  4472        END;
28 0D42  07 0F8C  4473                                                          195
28 0D42  07 0F8C  4474        COMMENT    <STEPUNTIL>  ::= STEP <T EXP>   UNTIL;
28 0D42  07 0F8C  4475        BEGIN
28 0D46  07 0F8C  4476          IC(R1,V22(R8+8));
28 0D4A  07 0F8C  4477          IF R1 ¬= 1 THEN
28 0D52  07 0F8C  4478          BEGIN   RO := @TYPEERROR;  ERROR;
28 0D62  07 0F8C  4479          END;
28 0D62  07 0F8C  4480          F01 := V(R8+8);   V(R8) := F01;
28 0D6A  07 0F8C  4481        END;
28 0D6A  07 0F8C  4482                                                          196
```

```
28 0D6A  07 0F8C  4483          COMMENT    <STEPUNTIL>  ::= UNTIL;
28 0D6A  07 0F8C  4484          BEGIN
28 0D6E  07 0F8C  4485            R9 := R9+4;  R0 := @NUMBER;   STC(R0,OP(R9));  V5(R8) := R9;
28 0D7E  07 0F8C  4486          END;
28 0D7E  07 0F8C  4487                                                                        197
28 0D7E  07 0F8C  4488          COMMENT    <WHILE CL>   ::=  WHILE  <T EXP>   ;
28 0D7E  07 0F8C  4489          BEGIN
28 0D82  07 0F8C  4490            IC(R0,V22(R8+8));
28 0D86  07 0F8C  4491            IF R0 ¬= 6 THEN BEGIN R0 := @TYPEERROR; ERROR; END;
28 0D9E  07 0F8C  4492            R9 := @B9(4);  R0 := @WHILEE; STC(R0,OP(R9));
28 0DAA  07 0F8C  4493            R0 := @WHILEOP;  R0 := R0 OR #80;  R9 := R9 + 4;
28 0DB6  07 0F8C  4494            STC(R0,OP(R9));  R1 := V5(R8+8) - OUTBASE;
28 0DC2  07 0F8C  4495            POINTER(R9) := R1;  V5(R8) := R9;
28 0DCA  07 0F8C  4496          END;
28 0DCA  07 0F8C  4497                                                                        198
28 0DCA  07 0F8C  4498          COMMENT    <ST PROC HD> ::=  <ST PROC ID>  (    ;
28 0DCA  07 0F8C  4499          BEGIN
28 0DCE  07 0F8C  4500            R1 := V1(R8);  IC(R0,VR(R1));  V34(R8) := R0;
28 0DDA  07 0F8C  4501          END;
28 0DCA  07 0F8C  4502                                                                        199
28 0DDA  07 0F8C  4503          COMMENT    <ST PROC HD> ::=  <ST PROC HD>  <T EXP>    ,      ;
28 0DCA  07 0F8C  4504          BEGIN
28 0DDE  07 0F8C  4505            RESULT;
28 0DEA  07 0F8C  4506            R0 := @APCOMMA;  OUTOP;
28 0DFA  07 0F8C  4507          END;
28 0DFA  07 0F8C  4508                                                                        200
28 0DFA  07 0F8C  4509          COMMENT    <UNIT DES>   ::=  <UNIT HD>  <T EXP>    )  ;
28 0DFA  07 0F8C  4510          BEGIN
28 0DFE  07 0F8C  4511            R2 := V1(R8);   IC(R0,SIMPLETYPE(R2));
28 0EC6  07 0F8C  4512            IF R0 ¬= 2  THEN
28 0E0E  07 0F8C  4513            BEGIN  R0 := @FILEERROR;  ERROR;
28 0E1E  07 0F8C  4514            END;
28 0E1E  07 0F8C  4515            R2 := V2(R8+8);
28 0E22  07 0F8C  4516            IF R2 ¬= 1 THEN
28 0E2A  07 0F8C  4517            BEGIN  R0 := @TYPEERROR;  ERROR;
28 0E3A  07 0F8C  4518            END;
28 0E3A  07 0F8C  4519            R0 := @FILEINDX;  OUTOP;
28 0E4A  07 0F8C  4520          END;
28 0E4A  07 0F8C  4521                                                                        201
28 0E4A  07 0F8C  4522          COMMENT    <UNIT DES>   ::=  <FILE ID>  ;
28 0E4A  07 0F8C  4523          NULL;
28 0E4E  07 0F8C  4524                                                                        202
28 0E4E  07 0F8C  4525          COMMENT    <UNIT HD> ::=  <FILE ID>  (  ;
28 0E4E  07 0F8C  4526          NULL;
28 0E52  07 0F8C  4527
28 0E52  07 0F8C  4528      END;    COMMENT END OF CASE ST;
28 0FA6  07 0F8C  4529      R10 := RASAVE;
28 0FAA  07 0F8C  4530      END;    COMMENT END OF LEVELTWO;

                         SEGMENT 28  NAME = SEG#28      LENGTH = 1000  BASE REG = 15

27 01DC  07 0F8C  4531      LEVELTWO;
27 01E6  07 0F8C  4532    END;  COMMENT END OF EXECUTE3;

                         SEGMENT 27  NAME = SEG#27      LENGTH = 0228  BASE REG = 15

23 0474  07 0F8C  4533
23 0474  07 0F8C  4534
```

```
23 0474  07 0F8C  4535  PROCEDURE  INSYMBOL(R3);
23 0474  07 0F8C  4536  COMMENT   INSYMBOL PUTS NEXT SYMBOL IN R6, BIT SEQUENCE OR STRING IN
23 0474  07 0F8C  4537            LITERAL TABLE, NUMBER IN NUMVALUE, POINTER TO ID, STRING,
23 0474  07 0F8C  4538            OR BIT SEQUENCE IN VALUE;
23 0474  07 0F8C  4539      BEGIN
23 0474  07 0F8C  4540        R2 := INPOINT;
23 0478  07 0F8C  4541  L:    IC(R6,PROGRAM(R2));  R5 := 0;  STC(R5,SIMTYPE);
23 04E4  07 0F8C  4542        IF R6 = #FE THEN BEGIN  IC(R5,PROGRAM(R2+1));  R5 := R5 SHLA 8;
23 0494  07 0F8C  4543                                  IC(R5,PROGRAM(R2+2));   CARDNUMBER := R5;
23 C49C  07 0F8C  4544                                  R2 := R2 + 3;   GOTO L;
23 04A4  07 0F8C  4545                                END  ELSE IF R6 = BITCODE THEN
23 04B0  07 0F8C  4546                                BEGIN  R0 := 8;  STC(R0,SIMTYPE);  R5 := 3;
23 04BC  07 0F8C  4547                                  R4 := @PROGRAM(R2+1);  R2 := R2 + R5 + 1;
23 04C6  07 0F8C  4548                                  LITERAL;
23 04D2  07 0F8C  4549                                END  ELSE IF R6 = STRINGCODE THEN
23 04DE  07 0F8C  4550                                BEGIN  R0 := 7;  STC(R0,SIMTYPE);
23 04E6  07 0F8C  4551                                  R4 := @PROGRAM(R2+2);
23 04EA  07 0F8C  4552                                  IC(R5,PROGRAM(R2+1));  R2 := R2+R5+2;
23 04F4  07 0F8C  4553                                  STC(R5,STRINGLENGTH);  LITERAL;
23 0504  07 0F8C  4554                                END  ELSE IF R6 = NUMBERCODE THEN
23 0510  07 0F8C  4555                                BEGIN  IC(R5,PROGRAM(R2+1));  STC(R5,SIMTYPE);
23 0518  07 0F8C  4556                                  IC(R0,LENGTHTABLE(R5));  R4 := R0 - 1;
23 0522  07 0F8C  4557                                  F23 := F23-F23;  NUMVALUE := F23;
23 0528  07 0F8C  4558                                  NUMVALUE( 8):= F23;
23 052C  07 0F8C  4559                                  R5 := @PROGRAM(R2+2);  EX(R4,MOVENUMBER);
23 0534  07 0F8C  4560                                  R2 := R2 + R4 + 2;
23 053A  07 0F8C  4561                                END  ELSE IF R6 = IDCODE THEN
23 0546  07 0F8C  4562                                BEGIN  IC(R5,PROGRAM(R2+1));  R5 := R5 SHLA 8;
23 054E  07 0F8C  4563                                  IC(R5,PROGRAM(R2+2));
23 0552  07 0F8C  4564                                    R2 := R2+2;
23 0556  07 0F8C  4565                                END;
23 0556  07 0F8C  4566          R2 := R2 + 1;  INPOINT := R2;
23 055E  07 0F8C  4567          VALUE := R5;
23 0562  07 0F8C  4568        END;
23 0564  07 0F8C  4569
23 0564  07 0F8C  4570  SHORT INTEGER ERSYM;
23 C564  07 0F8E  4571  SHORT INTEGER RELATION;
23 0564  07 0F90  4572  INTEGER R1POINT;
23 0564  07 0F94  4573  BYTE ERRFLAG;
23 0564  07 0F95  4574
23 0564  07 0F95  4575  PROCEDURE   FETCH (R10);
23 0564  07 0F95  4576  COMMENT   FETCH   TAKES THE RELATION BETWEEN C(R1) AND C(R2) FROM
23 0564  07 0F95  4577            THE MATRIX AND PUTS IT IN R4.   REGISTERS R1 - R5  ARE
23 0564  07 0F95  4578            USED AND DESTROYED.  R1 AND R2 ARE CLEARED;
23 0564  07 0F95  4579    BEGIN
23 0564  07 0F95  4580      IC(R1,RMAP(R1));  IC(R2,CMAP(R2));
23 056C  07 0F95  4581      R1 := R1 SHLL 6 + R2;   R4 := 0;  R2 := R4;
23 0578  07 0F95  4582     IC(R4,MATRIX(R1));  R1 := R2; COMMENT BITPAIR NOW IN R4;
23 057E  07 0F95  4583    END;
23 0580  07 0F95  4584
23 C580  07 0F95  4585  PROCEDURE BACKUP(R10);
23 0580  07 0F95  4586  COMMENT BACKS UP THE INPUT STRING ONE SYMBOL;
23 0580  07 0F95  4587    BEGIN  R2 := INPOINT;
23 0584  07 0F95  4588      IF R6 = IDCODE THEN  R2 := R2 - 3 ELSE
23 0590  07 0F95  4589      IF R6 = BITCODE THEN R2 := R2 - 5 ELSE
23 05A0  07 0F95  4590      IF R6 = STRINGCODE THEN
23 05AC  07 0F95  4591      BEGIN  IC(R0,STRINGLENGTH);  R2 := R2 - R0 - 3;
23 05B6  07 0F95  4592      END ELSE
```

```
23 05B6  07 0F95  4593      IF R6 = NUMBERCODE THEN
23 05C2  07 0F95  4594      BEGIN IC(R6,SIMTYPE);  IC(R6,LENGTHTABLE(R6));  R2 := R2 - R6 - 2;
23 05D0  07 0F95  4595      END  ELSE R2 := R2 - 1;
23 05D8  07 0F95  4596      INPOINT := R2;
23 05DC  07 0F95  4597    END;
23 05DE  07 0F95  4598
23 05DE  07 0F95  4599  COMMENT * * * * EXECUTION * * * * ;
23 05DE  07 0F95  4600
23 05DE  07 0F95  4601    RASAVE := R10;            COMMENT  SAVE LINK REGISTER;
23 05E2  07 0F95  4602    R11 := COMMONBASE;  COMMENT INITIALIZE UPPER-CORE BASE REGISTER;
23 05E6  07 0F95  4603    RESET(COMMONFLAG);   R1 := _4;  TREELINK := R1;
23 05F2  07 0F95  4604    R1 := TREEBASE;  TREEORG := R1;
23 05FA  07 0F95  4605    R1 := INPOINT - 48;  LITBASE := R1;
23 0606  07 0F95  4606    R1 := INPOINT;  INPOINTSAVE := R1;
23 060E  07 0F95  4607    R1 := #1300;  TYPES := R1;  COMMENT  FOR ERROR RECOVERY;
23 0616  07 0F95  4608    R1 := 0; BN := R1; BNC := R1; FLAG := R1; TREELENGTH := R1;
23 062A  07 0F95  4609    DRELPOINT := R1; DRELAD := R1; OUTBASE := R1;
23 0636  07 0F95  4610    SNLIST(4) := R1; CONSPOINTERSTACK(0) := R1;
23 063E  07 0F95  4611    CTORG := R1; LITORG := R1;  PATCHPOINT := R1;  OUTBASE := R1;
23 064E  07 0F95  4612    R1 := 1; SN := R1; SNC := R1;  LITINITIAL := R1;
23 065E  07 0F95  4613    R1 := 4; LITPNT := R1; R1 := 12; CTPNT := R1;
23 066E  07 0F95  4614    R1 := 13; HN := R1; R1 := 16; VARORIGIN := R1;
23 067E  07 0F95  4615    R1 := 24;  LABELADDR := R1;
23 0686  07 0F95  4616    R1 := 1; CINFO(0) := R1;  R1 := 6; CINFO(4) := R1; CINFO(8) := R1;
23 069A  07 0F95  4617    R1 := 0; CADDR(0) := R1; CADDR(4) := R1;  R1 := 3; CADDR(8) := R1;
23 06AE  07 0F95  4618    MVI(#92,S);  MVI(#01,S2); MVI(" ",BUFFER);
23 06BA  07 0F95  4619    MVC(15,TRANSTABLE,"0123456789ABCDEF");
23 06C0  07 0F95  4620    MVI("1",CARRCONT);  R6 := 0;
23 06C8  07 0F95  4621    R0 := 0;
23 06CC  07 0F95  4622    INSYMBOL;    R7 := 0;  R1 := R7;
23 06D6  07 0F95  4623    R3 := LITBASE;   MVC(3,LITERALTABLE,LITINITIAL);
23 06E0  07 0F95  4624    R9 := #C01;    IDLOC2 := R9;
23 06E8  07 0F95  4625    R0 := @PROCDC;  STC(R0,OP(4));  R0 := @CARD;  STC(R0,OP(8));
23 06F8  07 0F95  4626    R3 := CARDNUMBER;  POINTER( 8):= R3;  R9 := 8;  V5(0) := R9;
23 0708  07 0F95  4627  COMMENT     ALGORITHM FOR SYNTACTIC ANALYSIS;
23 0708  07 0F95  4628    WHILE R6 ¬= ENDFILE  DO
23 0710  07 0F95  4629    BEGIN  R7 := R7 + 1;          COMMENT STEP I;
23 0714  07 0F95  4630       R8 := R7;                  COMMENT  J := I;
23 0716  07 0F95  4631       STC(R6,S(R8));            COMMENT NEW SYMBOL IN PARSING STACK;
23 071A  07 0F95  4632       R8 := R8 SHLA 3;           COMMENT STORE INFO IN VALUE STACK;
23 071E  07 0F95  4633       IC(R0,SIMTYPE);  V2(R8) := R0;
23 0726  07 0F95  4634       R5 := VALUE;    V1(R8) := R5;
23 072E  07 0F95  4635       R8 := R8 SHRA 3;
23 0732  07 0F95  4636       INSYMBOL ;        IC(R1,S(R8));
23 073A  07 0F95  4637       RESET(ERRFLAG);
23 073E  07 0F95  4638  X:   R2 := R6;         FETCH;   COMMENT  GET RELATION FROM MATRIX;
23 0744  07 0F95  4639  X2:  IF R4 = GTR THEN
23 074C  07 0F95  4640       BEGIN
23 074C  07 0F95  4641  Y:     R8 := R8-1;  IC(R4,S2(R8));
23 0754  07 0F95  4642         IF R4 = EQL  THEN  GOTO  Y;
23 075C  07 0F95  4643         R8 := R8 + 1;           COMMENT  J NOW SET;
23 0760  07 0F95  4644         SET(FLAG);     R3 := 0;
23 0768  07 0F95  4645         IC(R3,S(R8)); R3 := R3 SHLA 1; R3 := MTB(R3);
23 0774  07 0F95  4646         R3 := @PRTB(R3);            COMMENT  GET ADDR. OF FIRST PROD.;
23 0778  07 0F95  4647         R4 := @S(R8);               COMMENT GET ADDR. OF STACK;
23 077C  07 0F95  4648         R2 := R7 - R8;               COMMENT  I-J;
23 0780  07 0F95  4649  Z:     TEST(FLAG);     IF = THEN
23 0788  07 0F95  4650           BEGIN     COMMENT  PRTB  LOOKUP;
```

```
23 C788  07 0F95  4651          IC(R1,B3);  IF R1 ¬= NOMORERULES  THEN
23 C794  07 0F95  4652           BEGIN  IF R1 = R2 THEN
23 C79A  07 0F95  4653            BEGIN  EX(R2,COMPARE);   IF = THEN
23 C7A2  07 0F95  4654               BEGIN   RESET(FLAG);  R5 := 0;
23 C7AA  07 0F95  4655                  IC(R5,B3(R2+3));  STM(R1,R6,R1SAVE);
23 C7B2  07 0F95  4656                       R7 := R7 SHLA 3;  R8 := R8 SHLA 3;
23 C7BA  07 0F95  4657                  IF R5 < 69 THEN EXECUTE1 ELSE
23 C7CC  07 0F95  4658                  IF R5 < 119 THEN EXECUTE2 ELSE EXECUTE3;
23 C7F0  07 0F95  4659                        LM(R1,R6,R1SAVE);
23 C7F4  07 0F95  4660                     R7 := R7 SHRA 3;  R8 := R8 SHRA 3;
23 C7FC  07 0F95  4661               END;
23 C7FC  07 0F95  4662            END;
23 C7FC  07 0F95  4663             R3 :=   @B3(R1+4);  COMMENT SET POINTER TO NEXT RULE;
23 0800  07 0F95  4664            GOTO  Z;
23 C804  07 0F95  4665           END;
23 0804  07 0F95  4666          END;
23 0804  07 0F95  4667         TEST(FLAG);  IF = THEN
23 080C  07 0F95  4668         BEGIN  R0 := @SYNTAXERROR;  ERROR;
23 081C  07 0F95  4669           RESET(FLAG);  RESET(ERRFLAG);
23 0824  07 0F95  4670           COMMENT MOVE INPUT AHEAD TO ".","END","BEGIN", OR EOF;
23 0824  07 0F95  4671           R8 := R7;
23 0826  07 0F95  4672           IF R6 = ENDCODE OR R6 = SCOLCODE OR R6= BEGINCODE OR R6=ENDFILE
23 083E  07 0F95  4673                    THEN GOTO L2;
23 0846  07 0F95  4674           R7 := INPOINT;  R4 := 0;
23 084E  07 0F95  4675           IF R6 = FORCODE THEN
23 0856  07 0F95  4676           BEGIN  R4 := BN+2;  BN := R4;
23 0862  07 0F95  4677           END;
23 0862  07 0F95  4678  L:       IC(R4,PROGRAM(R7));
23 0866  07 0F95  4679           IF R4 = SCOLCODE THEN
23 086E  07 0F95  4680           BEGIN  R7 := R7 + 1;  INPOINT := R7;  INSYMBOL;
23 087A  07 0F95  4681           END ELSE IF R4 = ENDCODE OR R4 = BEGINCODE THEN
23 088E  07 0F95  4682           BEGIN  INPOINT := R7;  INSYMBOL;
23 0896  07 0F95  4683           END  ELSE IF R4 = IDCODE THEN
23 08A2  07 0F95  4684           BEGIN  R7 := R7 + 3;  GOTO L;
23 08AA  07 0F95  4685           END ELSE IF R4 = #FE THEN
23 08B6  07 0F95  4686           BEGIN  IC(R4,PROGRAM(R7+1));  R4 := R4 SHLL 8;
23 08BE  07 0F95  4687             IC(R4,PROGRAM(R7+2));  CARDNUMBER := R4;  R4 := 0;
23 08CA  07 0F95  4688             R7 := R7 + 3;  GOTO L;
23 08D2  07 0F95  4689           END ELSE IF R4 = BITCODE THEN
23 08DE  07 0F95  4690           BEGIN  R7 := R7 + 5;  GOTO L;
23 08E6  07 0F95  4691           END ELSE IF R4 = STRINGCODE THEN
23 08F2  07 0F95  4692           BEGIN  IC(R4,PROGRAM(R7+1));  R7 := R7 + R4 + 3;  GOTO L;
23 0900  07 0F95  4693           END ELSE IF R4 = NUMBERCODE THEN
23 090C  07 0F95  4694           BEGIN  IC(R4,PROGRAM(R7+1));  IC(R4,LENGTHTABLE(R4));
23 0914  07 0F95  4695             R7 := R7 + R4 + 2;  GOTO L;
23 091E  07 0F95  4696           END ELSE IF R4 = ENDFILE THEN
23 092A  07 0F95  4697           BEGIN  R6 := R4; R7 := R8; GOTO L3;
23 0932  07 0F95  4698           END ELSE IF R4 = FORCODE THEN
23 093E  07 0F95  4699           BEGIN  R4 := BN+2;  BN:= R4;  R4 := 0;  R7 := R7+1;  GOTO L;
23 0956  07 0F95  4700           END ELSE
23 0956  07 0F95  4701           BEGIN  R7 := R7 + 1;  GOTO L;
23 0962  07 0F95  4702           END;
23 0962  07 0F95  4703           COMMENT  MOVE STACK BACK TO "BLOCKHEAD", "CASE SEQ", OR
23 0962  07 0F95  4704                    "BLOCKBODY".  RESET OUTPUT POINTER;
23 0962  07 0F95  4705           R7 := 0;
23 0966  07 0F95  4706  L2:      IC(R7,S(R8));
23 096A  07 0F95  4707           IF R7 ¬= BHCODE AND R7 ¬= BBCODE AND R7 ¬= CASESEQCODE AND
23 0982  07 0F95  4708              R7 ¬= ENDFILE THEN
```

```
23 098A  07 0F95  4709          BEGIN
23 098A  07 0F95  4710            IF R7 = BNDLSTHD THEN
23 0992  07 0F95  4711            BEGIN  R8 := R8 SHLL 3; R2 := V2(R8); BNC := R2;
23 099E  07 0F95  4712              R8 := R8 SHRL 3;
23 09A2  07 0F95  4713            END ELSE IF R7 = PROCNT THEN
23 09AE  07 0F95  4714            BEGIN  R4 := BNC;  R4 := BLOCKLIST2(R4);  BNC := R4;
23 09BA  07 0F95  4715              R4 := HN+1;  HN := R4;
23 09C6  07 0F95  4716            END ELSE IF R7 = ENDCODE THEN SET(FLAG);
23 09D6  07 0F95  4717            R8 := R8 - 1;  GOTO L2;
23 09DE  07 0F95  4718          END;
23 09DE  07 0F95  4719          TEST(FLAG);
23 09E2  07 0F95  4720          IF = THEN
23 09E6  07 0F95  4721          BEGIN BACKUP;  R6 := ENDCODE;  RESET(FLAG);
23 09F2  07 0F95  4722          END;
23 09F2  07 0F95  4723          R7 := R8 SHLL 3;  R9 := V5(R7);
23 09FC  07 0F95  4724          R7 := R7 SHRL 3;
23 0A00  07 0F95  4725        END ELSE
23 0A00  07 0F95  4726          BEGIN
23 0A04  07 0F95  4727          R7 := R8;  R3 := R3-2;  IC(R2,B3);  STC(R2,S(R8));
23 0A12  07 0F95  4728          IC(R1,S(R8-1)); FETCH; STC(R4,S2(R8-1));
23 0A1E  07 0F95  4729          END;
23 0A1E  07 0F95  4730        IC(R1,S(R8));
23 0A22  07 0F95  4731        GOTO X;
23 0A26  07 0F95  4732      END;   COMMENT  END OF BLOCK BEGINNING AT X2;
23 0A26  07 0F95  4733        IF R4 = 0 THEN
23 0A2C  07 0F95  4734  COMMENT  ERROR RECOVERY;
23 0A2C  07 0F95  4735        BEGIN  R0 := @RELATIONERROR;  ERROR;
23 0A3C  07 0F95  4736          IF R6 ¬= 0 THEN
23 0A42  07 0F95  4737          BEGIN
23 0A42  07 0F95  4738            IC(R1,S(R0));  MVC(120,BUFFER(1),BUFFER);
23 0A4C  07 0F95  4739            IF R1 < FIRSTTERMINAL THEN
23 0A54  07 0F95  4740            BEGIN
23 0A54  07 0F95  4741              IF R1 = BBCODE THEN
23 0A5C  07 0F95  4742              BEGIN  R1 := #A;  GOTO L4;
23 0A64  07 0F95  4743              END ELSE
23 0A64  07 0F95  4744              IF R1 = SITYPECODE THEN
23 0A70  07 0F95  4745              BEGIN  R1 := #2;  GOTO L4;
23 0A78  07 0F95  4746              END ELSE
23 0A78  07 0F95  4747              IF R1 = PROCNT THEN
23 0A84  07 0F95  4748              BEGIN R1 := #68;  GOTO L4;
23 0A8C  07 0F95  4749              END ELSE
23 0A8C  07 0F95  4750              IF R1 = TFUNCDES THEN
23 0A98  07 0F95  4751              BEGIN  R1 := #84; GOTO L4;
23 0AA0  07 0F95  4752              END ELSE
23 0AA0  07 0F95  4753              IF R1 = TPROCBDY THEN
23 0AAC  07 0F95  4754              BEGIN  R1 := #88;  GOTO L4;
23 0AB4  07 0F95  4755              END ELSE
23 0AB4  07 0F95  4756              IF R1 = FORCLCODE OR R1 = WHILECLCODE THEN
23 0AC8  07 0F95  4757              BEGIN  R1 := #78;  GOTO L4;
23 0ADC  07 0F95  4758              END ELSE R1 := FIRSTTERMINAL;
23 0AC8  07 0F95  4759            END  ELSE
23 0AC8  07 0F95  4760            IF R1 = IDCODE THEN
23 0AE4  07 0F95  4761            BEGIN
23 0AE4  07 0F95  4762              R7 := R7 SHLL 3;
23 0AE8  07 0F95  4763              R8 := R8 SHLL 3;
23 0AEC  07 0F95  4764              R6SAVE := R6;
23 0AF0  07 0F95  4765              R1 := V1(R8);  RESET(UNDECLFLAG);  DECLARETEST;
23 0AFC  07 0F95  4766              V1(R8) := R1;  COMMENT RESTORE IDNO;
```

```
23 0BC0  07 0F95  4767                R8 := R8 SHRL 3;
23 0BC4  07 0F95  4768                R7 := R7 SHRL 3;
23 0BC8  07 0F95  4769                MVC(130,BUFFER(1),BUFFER);
23 0BCE  07 0F95  4770                TEST(UNDECLFLAG);
23 0BD2  07 0F95  4771                IF = THEN
23 0BD6  07 0F95  4772                BEGIN R1 := #6E;  GOTO L4;
23 0BDE  07 0F95  4773                END;
23 0BDE  07 0F95  4774                R1 := FIRSTTERMINAL + 1;
23 0BE6  07 0F95  4775              END ELSE R1 := R1 + 1;
23 0BEE  07 0F95  4776              R1 := R1 - FIRSTTERMINAL SHLL 1;  R1 := EMTB(R1);
23 0BFA  07 0F95  4777    L4:
23 0BFA  07 0F95  4778              TEST(ERRFLAG);  IF = AND R1 = R1POINT THEN R1 := R1 + 2;
23 0C0E  07 0F95  4779              SET(ERRFLAG);
23 0C12  07 0F95  4780              IC(R4,CODE(R1));
23 0C16  07 0F95  4781              WHILE R4 ¬= 0 DO
23 0C1C  07 0F95  4782              BEGIN
23 0C1C  07 0F95  4783                R1POINT := R1;
23 0C20  07 0F95  4784                CASE  R4 OF
23 0C2C  07 0F95  4785                BEGIN
23 0C30  07 0F95  4786                  BEGIN  COMMENT REDUCE;
23 0C38  07 0F95  4787                    MVC(15,BUFFER(1),"REDUCTION FORCED");
23 0C3E  07 0F95  4788                    IC(R0,OUTFLAG);  IF R0 >= 3 THEN  BEGIN
23 0C4A  07 0F95  4789                    R0 := @BUFFER;  PRINT; R0 := 0;
23 0C5E  07 0F95  4790                                                    END;
23 0C5E  07 0F95  4791                    R1 := 0;
23 0C62  07 0F95  4792                    R4 := GTR;  GOTO X2;
23 0C9A  07 0F95  4793                  END;
23 0C9A  07 0F95  4794                  BEGIN  COMMENT INSERT;
23 0C9E  07 0F95  4795                  R2 := 0;   IC(R2,NODE(R1));  ERSYM := R2;
23 0CAA  07 0F95  4796                  R1 := 0;   IC(R1,S(R8));   FETCH;
23 0CB6  07 0F95  4797                  IF R4 ¬= 0 THEN
23 0CBC  07 0F95  4798                  BEGIN
23 0CBC  07 0F95  4799                  R1 := ERSYM;  RELATION := R4;
23 0CC4  07 0F95  4800                  R2 := R6;  FETCH;
23 0CCA  07 0F95  4801                  IF R4 ¬= 0 THEN
23 0CD0  07 0F95  4802                    BEGIN  R2 := RELATION;
23 0CD4  07 0F95  4803                      IF R2 = GTR THEN
23 0CDC  07 0F95  4804                      BEGIN  BACKUP; R6 := ERSYM;  R4 := GTR;
23 0CE8  07 0F95  4805                        IF R6 = IDCODE THEN R2 := 1 ELSE R2 := 0;
23 0CFC  07 0F95  4806                        STC(R2,SIMTYPE);  R2 := 0;  VALUE := R2;
23 0D08  07 0F95  4807                      END ELSE
23 0D08  07 0F95  4808                  BEGIN R7 := R7+1;  R1 := ERSYM; STC(R1,S(R7));
23 0D18  07 0F95  4809                    IF R1 = IDCODE THEN R1 := 1 ELSE R1 := 0;
23 0D2C  07 0F95  4810                    R7 := R7 SHLL 3;  V12(R7) := R1; R7 := R7 SHRL 3;
23 0D38  07 0F95  4811                    R1 := RELATION;  STC(R1,S2(R7-1));
23 0D40  07 0F95  4812                    R8 := R7;
23 0D42  07 0F95  4813                  END;
23 0D42  07 0F95  4814                    MVC(7,BUFFER(14),"INSERTED");
23 0D48  07 0F95  4815                    R2 := ERSYM * 12S + METABASE;  MVC(11,BUFFER(1),B2);
23 0D5A  07 0F95  4816                    IC(R0,OUTFLAG);   IF R0 >= 3 THEN BEGIN
23 0D66  07 0F95  4817                    R2 := R1;  R0 := @BUFFER; PRINT;  R0 := 0; R1 := R2;
23 0D7E  07 0F95  4818                                                      END;
23 0D7E  07 0F95  4819                    R2 := 0;
23 0D82  07 0F95  4820                    GOTO X2;
23 0D86  07 0F95  4821                  END;
23 0D86  07 0F95  4822                  END;
23 0D86  07 0F95  4823                END;
23 0D86  07 0F95  4824                BEGIN  COMMENT  REPLACE;
```

```
23 0C8A  07 0F95  4825                    R2 := 0;    IC(R2,NODE(R1)); ERSYM := R2;
23 0C96  07 0F95  4826                    R1 := 0;    IC(R1,S(R8-1));  FETCH;
23 0CA2  07 0F95  4827                    IF R4 ¬= 0 THEN
23 0CA8  07 0F95  4828                    BEGIN   R1 := ERSYM;   RELATION := R4;
23 0CB0  07 0F95  4829                      R2 := R6;   FETCH;
23 0CB6  07 0F95  4830                      IF R4 ¬= 0 THEN
23 0CBC  07 0F95  4831                      BEGIN  R1 := ERSYM;  STC(R1,S(R8));
23 0CC4  07 0F95  4832                        IF R1 = IDCODE THEN R1 := 1 ELSE R1 := 0;
23 0CD8  07 0F95  4833                        R2 := RELATION;
23 0CDC  07 0F95  4834                        IF R2 = GTR THEN
23 0CE4  07 0F95  4835                        BEGIN  BACKUP;  R6 := ERSYM;  R4 := GTR;
23 0CF0  07 0F95  4836                          R8 := R8 - 1;  STC(R1,SIMTYPE);  R1 := 0;
23 0CFC  07 0F95  4837                          VALUE := R1;  R7 := R8;
23 0DC2  07 0F95  4838                        END ELSE
23 0D02  07 0F95  4839                        BEGIN  R8 := R8 SHLL 3;  V12(R8) := R1;
23 0D0E  07 0F95  4840                          R8 := R8 SHRL 3;  STC(R2,S2(R8-1));
23 0D16  07 0F95  4841                        END;
23 0D16  07 0F95  4842                        MVC(20,BUFFER(1),"REPLACEMENT BY SYMBOL");
23 0D1C  07 0F95  4843                        R2 := ERSYM * 12S + METABASE;
23 0D28  07 0F95  4844                        MVC(11,BUFFER(23),B2);
23 0D2E  07 0F95  4845                        IC(R0,OUTFLAG);  IF R0 >= 3 THEN BEGIN
23 0D3A  07 0F95  4846                        R2 := R1; R0 := @BUFFER; PRINT; R1 := R2; R0 := 0;
23 0D52  07 0F95  4847                                                              END;
23 0D52  07 0F95  4848                        R2 := 0;
23 0D56  07 0F95  4849                        GOTO X2;
23 0D5A  07 0F95  4850                      END;
23 0D5A  07 0F95  4851                    END;
23 0D5A  07 0F95  4852                  END;
23 0D5A  07 0F95  4853                  BEGIN  COMMENT DELETE;
23 0D5E  07 0F95  4854                    MVC(13,BUFFER(1),"SYMBOL DELETED");
23 0D64  07 0F95  4855                    IC(R0,OUTFLAG);  IF R0 >= 3 THEN BEGIN
23 0D70  07 0F95  4856                    R0 := @BUFFER;  PRINT; R0 := 0;
23 0D84  07 0F95  4857                                                    END;
23 0D84  07 0F95  4858                    R8 := R8 - 1; R7 := R8; R1 := 0; IC(R1,S(R8));
23 0D92  07 0F95  4859                    GOTO X;
23 0D96  07 0F95  4860                  END;
23 0D96  07 0F95  4861                END;      COMMENT  END OF CASE STATEMENT;
23 0DAA  07 0F95  4862                R1 := R1POINT + 2;   IC(R4,CODE(R1));
23 0DB6  07 0F95  4863              END;        COMMENT    END OF WHILE STATEMENT;
23 0DBA  07 0F95  4864              COMMENT  IF NO OBVIOUS REMEDY, STACK SYMBOL AND CONTINUE;
23 0DBA  07 0F95  4865              R4 := EQL;  GOTO X2;
23 0DC2  07 0F95  4866            END ELSE GOTO EXIT;
23 0DC2  07 0F95  4867          END;   COMMENT  END OF ERROR RECOVERY;
23 0DC2  07 0F95  4868        STC(R4,S2(R8));
23 0DC6  07 0F95  4869    L3:
23 0DC6  07 0F95  4870      END;
23 0DCA  07 0F95  4871      IC(R0,S(1));
23 0DCE  07 0F95  4872      IF R0 ¬= BLOCKCODE OR R7 ¬= 1 THEN
23 0DDE  07 0F95  4873      BEGIN  R0 := @SYNTAXERROR;  ERROR;
23 0DEE  07 0F95  4874      END;
23 0DEE  07 0F95  4875      R9 := R9 + 4; R0 := @PCL; STC(R0,OP(R9)); R0 := 8; POINTER(R9) := R0;
23 0E02  07 0F95  4876      R1 := TREELENGTH+R9+4;  TREELENGTH := R1;
23 0E10  07 0F95  4877      R1 := POINTER(12) AND #FFF;  IDLOC1 := R1;
23 0E1C  07 0F95  4878      R1 := POINTER(12) AND #F000;  POINTER(12) := R1;
23 0E28  07 0F95  4879      R1 := 8; R2 := TREEORG;  CHECKSPACE;  IF ¬NOGO THEN
23 0E3C  07 0F95  4880      BEGIN  R3 := TREELINK; B2(0) := R3; TREELINK := R2;
23 0E48  07 0F95  4881        MVC(7,B2(4),"(MAIN)  ");
23 0E4E  07 0F95  4882      END;
```

```
23 0E4E  07 0F95  4883    R2 := @B2(12);  TREEORG := R2;
23 0E56  07 0F95  4884    R0 := 0;  R1 := R9;  R4 := 4;  R3 := 0; IC(R3,OUTFLAG);
23 0E68  07 0F95  4885    IF R3 > 5 THEN WRITETREE;  R2 := TREEORG; R4 := @P(R4); MOVETREE;
23 0E86  07 0F95  4886    R1 := CTPNT + 4;    R4 := @CONSTANTTABLE(_4);  MOVETREE;
23 0E96  07 0F95  4887    R1 := LITPNT;  R3 := LITBASE;  R4 := @LITERALTABLE;
23 0EA2  07 0F95  4888    MOVETREE;  R2 := R2 + 3 SHRA 2 SHLA 2;
23 0EB2  07 0F95  4889    TREEORG := R2;
23 0EB6  07 0F95  4890    R2 := #300;  TYPES := R2;  COMMENT  RESTORE FOR PASS 3;
23 0EBE  07 0F95  4891
23 0EBE  07 0F95  4892
23 0EBE  07 0F95  4893  BEGIN
23 0EBE  07 0F95  4894   SEGMENT PROCEDURE   OUTPUT(R8);
23 0EBE  07 0F95  4895    BEGIN
29 0000  07 0F95  4896
29 0000  07 0F95  4897  ARRAY 132  BYTE OUTBUF SYN BUFFER;
29 0000  07 0F95  4898  INTEGER TEMP SYN HEX;
29 0000  07 0F95  4899
29 0000  07 0F95  4900  MVC(130,OUTBUF(1),OUTBUF); MVC(19,OUTBUF(1),"TOTAL TREE LENGTH IS");
29 000C  07 0F95  4901  UNPK(4,2,OUTBUF(25),TREELENGTH);  MVI(" ",OUTBUF(29));
29 0016  07 0F95  4902  TR(3,OUTBUF(25),TRANSTABLE(_240));  R0 := @OUTBUF;  PRINT;
29 002C  07 0F95  4903  R1 := TREEORG - TREEBASE;  TEMP := R1;
29 0038  07 0F95  4904  MVC(21,OUTBUF(1),"TOTAL OUTPUT LENGTH IS");
29 003E  07 0F95  4905  UNPK(4,2,OUTBUF(25),TEMP(2));  MVI(" ",OUTBUF(29));
29 0048  07 0F95  4906  TR(3,OUTBUF(25),TRANSTABLE(_240));  PRINT;
29 005A  07 0F95  4907
29 005A  07 0F95  4908  COMMENT  NAMETABLE  OUTPUT;       MVI("1",CARRCONT);
29 005E  07 0F95  4909    R0 := @OUTBUF;   MVC(39,OUTBUF(1),OUTBUF);
29 0068  07 0F95  4910    MVC(8,OUTBUF(1),"NAMETABLE");  PRINT;
29 007A  07 0F95  4911    MVC(77,OUTBUF(1),HEADER2); PRINT;
29 008C  07 0F95  4912    MVC(80,OUTBUF(1),OUTBUF);
29 0092  07 0F95  4913    MVC(63,OUTBUF(1),HEADER5);  PRINT;
29 00A4  07 0F95  4914    MVC(63,OUTBUF(1),OUTBUF);  R0 := 0;
29 00AE  07 0F95  4915    FOR R2 := 0 STEP 12 UNTIL  NAMETABLESIZE  DO
29 00B2  07 0F95  4916    BEGIN  TEMP := R2;  UNPK(4,2,OUTBUF(4),TEMP(2));
29 00C0  07 0F95  4917      MVI(" ",OUTBUF(8));
29 00C4  07 0F95  4918       TR(3,OUTBUF(4),TRANSTABLE(_240));
29 00CA  07 0F95  4919      IC(R0,TYPE(R2));
29 00CE  07 0F95  4920  COMMENT  IDLOC1 TO BUFFER;
29 00CE  07 0F95  4921    R1 := IDLOC1(R2);  TEMP := R1;  UNPK(4,2,OUTBUF(14),TEMP(2));
29 00DC  07 0F95  4922    MVI(" ",OUTBUF(18));  TR(3,OUTBUF(14),TRANSTABLE(_240));
29 00E6  07 0F95  4923  COMMENT  IDLOC2 TO BUFFER;
29 00E6  07 0F95  4924      IF R0 = 3 THEN
29 00EE  07 0F95  4925      BEGIN  R1 := 0;  IC(R1,HIERARCHY(R2));  TEMP := R1;
29 00FA  07 0F95  4926        UNPK(2,1,OUTBUF(21),TEMP(3));  MVI(" ",OUTBUF(23));
29 0104  07 0F95  4927        TR(1,OUTBUF(21),TRANSTABLE(_240));
29 010A  07 0F95  4928        MVC(5,OUTBUF(23),MASK2);  IC(R1,PROGSEG(R2));  CVD(R1,DEC);
29 0118  07 0F95  4929        ED(5,OUTBUF(23),DEC(5));
29 011E  07 0F95  4930      END ELSE
29 011E  07 0F95  4931      BEGIN  R1 := IDLOC2(R2);  TEMP := R1;
29 012A  07 0F95  4932        UNPK(4,2,OUTBUF(25),TEMP(2));  MVI(" ",OUTBUF(29));
29 0134  07 0F95  4933        TR(3,OUTBUF(25),TRANSTABLE(_240));
29 013A  07 0F95  4934        MVC(1,OUTBUF(21),OUTBUF(90));
29 0140  07 0F95  4935      END;
29 0140  07 0F95  4936  COMMENT  TYPEINFO TO BUFFER;
29 0140  07 0F95  4937      IF R0 = 0 OR R0 = 9 THEN
29 014E  07 0F95  4938      BEGIN    IC(R0,VR(R2));   CVD(R0,DEC);   R0 := 0;
29 015A  07 0F95  4939        MVC(5,OUTBUF(40),MASK);   ED(5,OUTBUF(40),DEC(5));
29 0166  07 0F95  4940        MVC(5,OUTBUF(50),OUTBUF(90));
```

```
29 016C  07 0F95   4941      END ELSE IF R0 = 4 THEN
29 0178  07 0F95   4942      BEGIN  IC(R0,VR(R2));  TEMP := R0;
29 0180  07 0F95   4943        UNPK(2,1,OUTBUF(43),TEMP(3));
29 0186  07 0F95   4944        TR(1,OUTBUF(43),TRANSTABLE(_240));  MVI(" ",OUTBUF(45));
29 0190  07 0F95   4945        IC(R0,RCCLNUMBER(R2));  CVD(R0,DEC);  MVC(5,OUTBUF(50),MASK);
29 019E  07 0F95   4946        ED(5,OUTBUF(50),DEC(5));  R0 := 4;
29 01A8  07 0F95   4947      END  ELSE
29 01A8  07 0F95   4948      BEGIN  IF R0 = 3 THEN  MVC(5,OUTBUF(50),MASK2  )ELSE
29 01BA  07 0F95   4949        MVC(5,OUTBUF(50),MASK);   R1 := TYPEINFO(R2);
29 01C8  07 0F95   4950        CVD(R1,DEC);  ED(5,OUTBUF(50),DEC(5));
29 01D2  07 0F95   4951        MVC(5,OUTBUF(40),OUTBUF(90));
29 01D8  07 0F95   4952      END;
29 01D8  07 0F95   4953  COMMENT  TYPE TO BUFFER;
29 01D8  07 0F95   4954    TEMP := R0;  UNPK(2,1,OUTBUF(62),TEMP(3));  MVI(" ",OUTBUF(64));
29 01E6  07 0F95   4955    TR(1,OUTBUF(62),TRANSTABLE(_240));
29 01EC  07 0F95   4956  COMMENT  SIMTYPEINFO TO BUFFER;
29 01EC  07 0F95   4957    IC(R0,SIMPLETYPE(R2));
29 01F0  07 0F95   4958      IF R0 = 9 THEN
29 01F8  07 0F95   4959      BEGIN  IC(R0,SIMTYPEINFO(R2));  CVD(R0,DEC);
29 0200  07 0F95   4960        MVC(5,OUTBUF(29),MASK);  ED(5,OUTBUF(29),DEC(5));
29 020C  07 0F95   4961        IC(R0,SIMTYPEINFO(R2+1));   CVD(R0,DEC);
29 0214  07 0F95   4962      MVC(5,OUTBUF(35),MASK);  ED(5,OUTBUF(35),DEC(5));  R0 := 9;
29 0224  07 0F95   4963      END  ELSE
29 0224  07 0F95   4964      BEGIN  R1 := SIMTYPEINFO(R2);
29 022C  07 0F95   4965        IF R0 = 7 THEN  MVC(5,OUTBUF(35),MASK2  )ELSE
29 023A  07 0F95   4966        MVC(5,OUTBUF(35),MASK);   CVD(R1,DEC);
29 0248  07 0F95   4967        ED(5,OUTBUF(35),DEC(5));  MVC(5,OUTBUF(29),OUTBUF(90));
29 0254  07 0F95   4968      END;
29 0254  07 0F95   4969      CVD(R0,DEC);  MVC(5,OUTBUF(67),MASK);  ED(5,OUTBUF(67),DEC(5));
29 0264  07 0F95   4970      R3 := IDDIRBASE; R1 := IDNO(R2) SHLL 2;  R10 := IDLENGTH(R1);
29 0274  07 0F95   4971      IF R10 > 55 THEN R10 := 55;  R1 := IDPOINT(R1);  R3 := IDLISTBASE;
29 0288  07 0F95   4972      R1 := @IDLIST(R1);  EX(R10,MOVE);
29 0290  07 0F95   4973      R0 := @OUTBUF;  PRINT;  R0 := 0;
29 02A4  07 0F95   4974      MVI(" ",OUTBUF(75));  MVC(54,OUTBUF(76),OUTBUF(75));
29 02AE  07 0F95   4975    END;
29 02BA  07 0F95   4976  COMMENT  BLOCKLIST OUTPUT;       MVI("1",CARRCONT);
29 02BE  07 0F95   4977    R0 := @OUTBUF;  MVC(81,OUTBUF(1),OUTBUF);
29 02C8  07 0F95   4978    MVC(8,OUTBUF(1),"BLOCKLIST");  PRINT;
29 02DA  07 0F95   4979    MVC(29,OUTBUF(1),"  BLOCKNO    LENGTH    POINTER");  PRINT;
29 02EC  07 0F95   4980    MVC(29,OUTBUF(1),OUTBUF);
29 02F2  07 0F95   4981    FOR R2 := 0 STEP 4 UNTIL BLOCKLISTSIZE  DO
29 02F6  07 0F95   4982    BEGIN  CVD(R2,DEC);
29 02FE  07 0F95   4983      MVC(5,OUTBUF(3),MASK2);  ED(5,OUTBUF(3),DEC(5));
29 030A  07 0F95   4984      R1 := BLENGTH(R2);  TEMP := R1;  UNPK(4,2,OUTBUF(15),TEMP(2));
29 0318  07 0F95   4985      TR(3,OUTBUF(15),TRANSTABLE(_240));  MVI(" ",OUTBUF(19));
29 0322  07 0F95   4986      R1 := NPOINT(R2);  TEMP := R1;  UNPK(4,2,OUTBUF(26),TEMP(2));
29 0330  07 0F95   4987      TR(3,OUTBUF(26),TRANSTABLE(_240));  MVI(" ",OUTBUF(30));
29 033A  07 0F95   4988      PRINT;
29 0346  07 0F95   4989    END;
29 0352  07 0F95   4990  END;   COMMENT  END OF OUTPUT PROCEDURE;

                  SEGMENT 29  NAME = SEG#29      LENGTH = 03D0  BASE REG = 15

23 0EBE  07 0F95   4991    R0 := 0;  IC(R0,OUTFLAG);  IF R0 < 3 THEN GOTO EXIT;
23 0ECE  07 0F95   4992    OUTPUT;
23 0ED8  07 0F95   4993    END;
23 0ED8  07 0F95   4994  EXIT:
23 0ED8  07 0F95   4995    R0 := 0;  IC(R0,NOGO);
```

```
23 0EE0   07 0F95   4996     R10 := INPOINTSAVE;  INPOINT := R10;
23 0EE8   07 0F95   4997     P10 := RASAVE;
23 0EEC   07 0F95   4998   END;  COMMENT  END OF PASS 2;

                          SEGMENT 23  NAME = SEG#23      LENGTH = 0FB0  BASE REG = 15

06 0874   07 0F95   4999
06 0874   07 0F95   5000   BYTE SEGMENTEND;  COMMENT  *** LAST DECLARATION IN SEGMENT *** ;
06 0874   07 0F96   5001
06 0874   07 0F96   5002   COMMENT   * * * *  EXECUTION BEGINS HERE  * * * * ;
06 0874   07 0F96   5003   OLDSAVE := R5;  STM(R10,R14,BASESAVE);
06 087C   07 0F96   5004   MVC(63,XFERVECTOR,B4);
06 0882   07 0F96   5005   R3 := AGETTIME; BALR(R2,R3); R11 := COMMLIM(0); COMMTIME := R0;
06 0890   07 0F96   5006   R0 := 60; LINENO := R0;  R0 := 0; PAGENO := R0;  MVI("1",CARRCONT);
06 08A4   07 0F96   5007   MVC(36,HEADING,"STANFORD ALGOL W  (VERSION 21SEPT69) ");
06 08AA   07 0F96   5008   MVC(94,HEADING(37),HEADING(36));  MVC(3,HEADING(113),"PAGE");
06 08B6   07 0F96   5009   R1 := ARUNID;  MVC(31,HEADING(80),B1);
06 08C0   07 0F96   5010   PASS1;  RETCODE := R0;  IF R0 = 0 THEN
06 08D4   07 0F96   5011   BEGIN  COMMENT  NO FATAL PASS 1 ERRORS;
06 08D4   07 0F96   5012     LM(R10,R14,BASESAVE);
06 08D8   07 0F96   5013     R0 := #2000; R1 := #4800;  R3 := AGETMAIN; BALR(R2,R3);
06 08E6   07 0F96   5014     R12 := R0; BASESAVE(8) := R1;  R0 := @P - R12;
06 08F2   07 0F96   5015     R0 := NEG R0 + R1 - 8 AND #FFFF00;  COMMENT (R0) = SIZE OF P - 8;
06 08FE   07 0F96   5016     PLIMIT := R0;
06 0902   07 0F96   5017     R1 := @P(0);  R2 := R1 + R0 - 256;  R0 := #FF000000;
06 0912   07 0F96   5018     P(0) := R0; P(4) := R0;
06 091A   07 0F96   5019     FOR R1 := R1 STEP 256 UNTIL R2 DO MVC(255,B1(8),B1);
06 092E   07 0F96   5020     PASS2;  RETCODE := R0;
06 093C   07 0F96   5021     R3 := AFREEMAIN;   COMMENT RELEASE WORKING STORAGE;
06 0940   07 0F96   5022     R0 := R12; R1 := BASESAVE(8);  BALR(R2,R3);
06 0948   07 0F96   5023   END;
06 0948   07 0F96   5024   R11 := COMMLIM(0);  R0 := RETCODE; IF R0 ¬= 0 THEN
06 0956   07 0F96   5025   BEGIN  MVI("0",CARRCONT);
06 095A   07 0F96   5026     MVC(42,BUFFER(6)," SECONDS IN COMPILATION, NO CODE GENERATED ");
06 0960   07 0F96   5027     MVC(82,BUFFER(49),BUFFER(48));
06 0966   07 0F96   5028     R3 := AGETTIME; BALR(R2,R3);  R1 := R0 -- COMMTIME;
06 0972   07 0F96   5029     R1 := R1*5 / 1920;  CVD(R1,PKDEC);  UNPK(4,2,BUFFER(1),PKDEC(5));
06 0984   07 0F96   5030     MVC(2,BUFFER(0),BUFFER(1)); MVI(".",BUFFER(3)); OI("0",BUFFER(5));
06 0992   07 0F96   5031     R0 := @BUFFER;  PRINT;
06 099A   07 0F96   5032   END ELSE
06 099A   07 0F96   5033   BEGIN  R0 := LINENO; COMMLINE := R0;  R0 := PAGENO;  COMMPAGE := R0;
06 09AE   07 0F96   5034   END;
06 09AE   07 0F96   5035   R5 := OLDSAVE;  R6 := RETCODE;
06 09B6   07 0F96   5036   R0 := R13; R1 := 4096; R3 := AFREEMAIN; BALR(R2,R3);
06 09C2   07 0F96   5037   R13 := R5;   R15 := R6;
06 09C6   07 0F96   5038   END;
06 09C6   00 0000   5039   R14 := B13(12);  LM(R0,R12,B13(20));
06 09CE   00 0000   5040   END.

                          SEGMENT 06  NAME = AWXCMPA1    LENGTH = 0D58  BASE REG = 15
```

```
IEF285I   T123.PLLIB                                  PASSED
IEF285I   VOL SER NOS= SYS11 .
IEF285I   SYSOUT                                      SYSOUT
IEF285I   VOL SER NOS=       .
IEF285I   AWXCMPA1                                    KEPT
IEF285I   VOL SER NOS= NIR001.
IEF280I K 0C2,NIR001,TOTAPE
IEF285I   T123.PLLIB                                  KEPT
IEF285I   VOL SER NOS= SYS11 .


H A S P  JOB STATISTICS --  5,051 CARDS READ --    5,174 LINES PRINTED --      0 CARDS PUNCHED --   0.65 MINUTES EXECUTION TIME
          36 I/O CALLS              113 SVC CALLS          0.43 MINUTES CPU TM    TIME=  23:59:09 DATE=  11/21/69
```

```
* 0.02.01 JOB 2349 IEF233A M 0C2,WIR001,TOTAPE
* 0.02.55 JOB 2349 IEF280I K 0C2,WIR001,TOTAPE
```

```
//
//TOTAPE JOB (T123***,914,2,10,,,,T),' ED SATTERTHWAITE ',MSGLEVEL=1     JOB2349
//JOBLIB  DD   DSNAME=T123.PLLIB,UNIT=2314,VOLUME=SER=SYS11,           X
//             DISP=(OLD,PASS)
//PL360  EXEC  PGM=PL360,PARM='NOLOAD,DECK'
//SYSPRINT DD  SYSOUT=A,DCB=BLKSIZE=133
//SYSPUNCH DD  DSNAME=AWXCMPB1,UNIT=TAPE9,VOLUME=SER=WIR001,           X
//             LABEL=(6,SL),DISP=(NEW,KEEP),                           X
//             DCB=(BLKSIZE=2000,LRECL=80,RECFM=FB)
//SYSIN    DD  *
IEF236I ALLOC. FOR TOTAPE   PL360
IEF237I JOBLIB   ON 391
IEF237I SYSPUNCH ON 0C2
IEF237I SYSIN    ON 00C
```

```
01 0000   00 0000          0001     COMMENT
01 0000   00 0000          0002                          STANFORD ALGOL W COMPILER
01 0000   00 0000          0003                          PHASE B - CODE GENERATION
01 0000   00 0000          0004                            OCTOBER 1969 VERSION          ;
01 0000   00 0000          0005
01 0000   00 0000          0006   GLOBAL PROCEDURE AWXCMPB1(R14);
01 0000   00 0000          0007   BEGIN  STM(R14,R12,B13(12));  R5 := R13;
06 0006   00 0000          0008    BEGIN
06 0006   00 0000          0009      SEGMENT BASE R13;
06 000A   07 0000          0010       LOGICAL SYSINF, OLDSAVE, NEWSAVE; ARRAY 15 LOGICAL SAVEAREA;
06 000A   07 0048          0011       ARRAY 16 INTEGER XFERVECTOR;  COMMENT SUPV ENTRY ADDRESSES;
06 000A   07 0088          0012          ARRAY 2 INTEGER COMMLIM SYN XFERVECTOR(0);
06 000A   07 0088          0013            INTEGER COMSTART SYN COMMLIM(0);
06 000A   07 0088          0014          INTEGER APUTLINE SYN XFERVECTOR(8);   COMMENT WRITE ENTRY;
06 000A   07 0088          0015            BYTE CARRCONT SYN APUTLINE(0);      COMMENT PRINT CONTROL CHAR;
06 000A   07 0088          0016          INTEGER APUTCARD SYN XFERVECTOR(16);  COMMENT PUNCH ENTRY;
06 000A   07 0088          0017          INTEGER AGETMAIN SYN XFERVECTOR(20);  COMMENT GETMAIN ENTRY;
06 000A   07 0088          0018          INTEGER AFREEMAIN SYN XFERVECTOR(24); COMMENT FREEMAIN ENTRY;
06 000A   07 0088          0019          INTEGER AGETTIME SYN XFERVECTOR(28);  COMMENT GETTIME ENTRY;
06 000A   07 0088          0020          INTEGER ARUNID SYN XFERVECTOR(60);    COMMENT ADDR OF SYSTEM ID;
06 000A   07 0088          0021       FUNCTION BALR(1,#0500);
06 000A   07 0088          0022       LONG REAL PKDEC;
06 000A   07 0090          0023         INTEGER WORK SYN PKDEC(0);
06 000A   07 0090          0024
06 000A   07 0090          0025       LOGICAL ALGOLDATAORG SYN B13(72);  COMMENT ALGOLRUN DATA ORIGIN;
06 000A   07 0090          0026
06 000A   07 0090          0027      DUMMY BASE R11;   COMMENT COMPILER COMMON FORMAT;
06 000A   08 0000          0028       INTEGER COMMTIME;
06 000A   08 0004          0029       SHORT INTEGER LINENO, PAGENO;
06 000A   08 0008          0030       ARRAY 3 LOGICAL COMMFLAGS;
06 000A   08 0014          0031          BYTE NOGO SYN COMMFLAGS(0);
06 000A   08 0014          0032          BYTE TRACE SYN COMMFLAGS(1);
06 000A   08 0014          0033          BYTE CHECKFLAG SYN COMMFLAGS(2);
06 000A   08 0014          0034          BYTE CARDFLAG SYN COMMFLAGS(3);
06 000A   08 0014          0035          ARRAY 2 LOGICAL TRACEBITS SYN COMMFLAGS(4);
06 000A   08 0014          0036       SHORT INTEGER BLOCKLISTSIZE, NAMETABLESIZE;
06 000A   08 0018          0037       INTEGER REFRECBASE, IDDIRBASE, IDLISTBASE, INPOINT;
06 000A   08 0028          0038       INTEGER TREELINK, TREEBASE, TREETOP;
06 000A   08 0034          0039       ARRAY 512 LOGICAL BLOCKLIST;
06 000A   08 0834          0040       COMMENT  *** SIZE OF OTHER TABLES IS DYNAMIC ***;
06 000A   08 0834          0041       ARRAY 3 LOGICAL NAMETABLE;
06 000A   08 0840          0042          BYTE VR SYN NAMETABLE(6);
06 000A   08 0840          0043          SHORT INTEGER IDLOC1 SYN NAMETABLE;
06 000A   08 0840          0044          SHORT INTEGER IDLOC2 SYN NAMETABLE(2);
06 000A   08 0840          0045          SHORT INTEGER SIMTYPEINFO SYN NAMETABLE(4);
06 000A   08 0840          0046          SHORT INTEGER TYPEINFO SYN NAMETABLE(6);
06 000A   08 0840          0047          BYTE TYPE SYN NAMETABLE(8);
06 000A   08 0840          0048          BYTE SIMPLETYPE SYN NAMETABLE(9);
06 000A   08 0840          0049       LOGICAL TREE SYN B12;
06 000A   08 0840          0050         SHORT INTEGER TREEP SYN TREE(2);
06 000A   08 0840          0051      CLOSE BASE;
06 000A   07 0090          0052
06 000A   07 0090          0053       LOGICAL SAVE14;   COMMENT STACK BASE ADDRESS;
06 000A   07 0094          0054       INTEGER STACKTOP; COMMENT LENGTH OF STACK SEGMENT;
06 000A   07 0098          0055
06 000A   07 0098          0056       LOGICAL STACK SYN MEM(R14);
06 000A   07 0098          0057         SHORT INTEGER STACKP SYN STACK(2);
06 000A   07 0098          0058         LOGICAL STACKP4 SYN STACK(4);
```

Micro Business Forms, Inc.

```
06 000A  07 0098  0059        LOGICAL STACKP8 SYN STACK(8);
06 000A  07 0098  0060      LOGICAL PROGRAMM  SYN MEM(R14);
06 000A  07 0098  0061        SHORT INTEGER PROGRAM SYN PROGRAMM;
06 000A  07 0098  0062      ARRAY   52  INTEGER LSTACKM8;
06 000A  07 0168  0063        INTEGER LSTACKM4 SYN LSTACKM8(4);
06 000A  07 0168  0064        INTEGER LSTACK SYN LSTACKM8(8);
06 000A  07 0168  0065      ARRAY  10  INTEGER NUMSTACK;
06 000A  07 0190  0066        ARRAY 10 SHORT INTEGER NUMSEQLEN SYN NUMSTACK(0);
06 000A  07 0190  0067        ARRAY 10 SHORT INTEGER NUMORIGIN SYN NUMSTACK(2);
06 000A  07 0190  0068      ARRAY 50  INTEGER LOGSTACK;
06 000A  07 0258  0069        SHORT INTEGER LOGSTACK1 SYN LOGSTACK;
06 000A  07 0258  0070        SHORT INTEGER LOGSTACK2 SYN LOGSTACK(2);
06 000A  07 0258  0071      ARRAY  5  INTEGER FSTACKM4;
06 000A  07 026C  0072        INTEGER FSTACK SYN FSTACKM4(4);
06 000A  07 026C  0073      ARRAY  13  INTEGER RSTACKM4;
06 000A  07 02A0  0074        INTEGER RSTACK SYN RSTACKM4(4);
06 000A  07 02A0  0075      ARRAY 32 LOGICAL RLDTABLE;
06 000A  07 0320  0076         ARRAY 32 SHORT INTEGER SEGNO SYN RLDTABLE(0);
06 000A  07 0320  0077         ARRAY 32 SHORT INTEGER CHAIN SYN RLDTABLE(2);
06 000A  07 0320  0078      ARRAY  20   SHORT INTEGER UMREC,MREC;
06 000A  07 0370  0079
06 000A  07 0370  0080      LOGICAL CONSPTTAB SYN 0;        LOGICAL CPTBASE;
06 000A  07 0374  0081      INTEGER CONSTAB,CONSTABL,CONSPTTABL;
06 000A  07 0380  0082      INTEGER LITORG;
06 000A  07 0384  0083      INTEGER NEXTADDR, SAVE, SAVEL, SAVER0, SAVER2, SAVER3, SAVER4,
06 000A  07 03A0  0084         SAVER5,SAVER6,LPOINTER,TEMP,CNTP;
06 000A  07 03B4  0085      INTEGER MNBASE, NEXTBASE;
06 000A  07 03BC  0086      INTEGER REFPCOUNT,REFADD,REFCOUNT,AREFADD,AREFCOUNT,LAADD,MVIADD;
06 000A  07 03D8  0087      INTEGER BASELOC, BASESTLOC;
06 000A  07 03E0  0088      LOGICAL FIRSTCASE;
06 000A  07 03E4  0089         SHORT INTEGER FIRSTCASE1 SYN FIRSTCASE;
06 000A  07 03E4  0090      INTEGER NUMSTACKP;
06 000A  07 03E8  0091      INTEGER RUNERRSTART,RUNERRPT,WRITRUNPT;
06 000A  07 03F4  0092      SHORT INTEGER SB2 SYN B2;
06 000A  07 03F4  0093      BYTE PRINT, TRACEIT; COMMENT LOCAL TRACEFLAGS;
06 000A  07 03F6  0094      BYTE ANDFLAG, RESFLAG, IFEXP, LOGFLAG, ARGFLAG, TERMNODE;
06 000A  07 03FC  0095
06 000A  07 03FC  0096      INTEGER TERMINALNODE=#53000000,  UNARYOP=#43000000;
06 000A  07 0404  0097      INTEGER FP=0,RETA=4,DL=8,REFVAR=#C,REFARY=#10,DSP=#14;
06 000A  07 041C  0098      LOGICAL MP = @ALGOLDATAORG(0), STRINGERR = @ALGOLDATAORG(#140),
06 000A  07 0424  0099         REFBINDERR = @ALGOLDATAORG(#14C), REFTEMP = @ALGOLDATAORG(#13C),
06 000A  07 042C  0100         CASEERR = @ALGOLDATAORG(#156), ARRAYERR = @ALGOLDATAORG(#16A),
06 000A  07 0434  0101         ALLOCERR = @ALGOLDATAORG(#188), NAMEERR = @ALGOLDATAORG(#17E),
06 000A  07 043C  0102         UBLBERR = @ALGOLDATAORG(#198), DESCERROR = @ALGOLDATAORG(#1A2),
06 000A  07 0444  0103         DUBLMASK = @ALGOLDATAORG(#014), SINGLMASK = @ALGOLDATAORG(#010),
06 000A  07 044C  0104         ALLONES = @ALGOLDATAORG(#018), DELTALIMIT = @ALGOLDATAORG(#00E),
06 000A  07 0454  0105         NULLREF = @ALGOLDATAORG(#01C), ADDRSTEP = @ALGOLDATAORG(#024),
06 000A  07 045C  0106         RECTABLE = @ALGOLDATAORG(#03C);
06 000A  07 0460  0107      INTEGER SREAD=#80000024,READFLAG=@ALGOLDATAORG(9);
06 000A  07 0468  0108      INTEGER SWRITE=#8000000C, WRITEFLAG=@ALGOLDATAORG(#A);
06 000A  07 0470  0109      LOGICAL LOINTREF=@ALGOLDATAORG(#1D4), HIINTREF=@ALGOLDATAORG(#1F8);
06 000A  07 0478  0110      LOGICAL FCON1 = @ALGOLDATAORG(#200), FCON2 = @ALGOLDATAORG(#208);
06 000A  07 0480  0111      LOGICAL SIGN=#80000000, TYPEMASK=#00FF0000, MASK=#0000FFFF,
06 000A  07 048C  0112         MASK1=#FFFF0000, MASK2=#FF00FFFF, MASK3=#FFF0FFFF,
06 000A  07 0498  0113         MASK4=#000F0000, MASK5=#00F00000, MASK6=#FF0FFFFF,
06 000A  07 04A4  0114         MASK7=#FF00FFFF, MASK8=#00000FFF, MASK9=#000000FF,
06 000A  07 04B0  0115         OPMASK=#7F000000, SMASK=#FFFFFFFC,
06 000A  07 04B8  0116         BASEMASK=#0000F000, OPCODEMASK=#FF000000;
```

```
06 000A  07 04C0  0117      LOGICAL A=#0A000000, D=#0D000000, M=#0C000000, S=#0B000000;
06 000A  07 04D0  0118      ARRAY 5 LOGICAL OPTYPE =      COMMENT  OPCODE DATA FORMAT BITS;
06 000A  07 04E4  0119         (#10000000, #30000000, #20000000, #30000000, #20000000);
06 000A  07 04E4  0120      LOGICAL LOADMARK=#B8000000, STOREMARK=#B0000000;
06 000A  07 04EC  0121      LOGICAL  ICC =#43000000,STCC=#42000000,MVII=#92000000;
06 000A  07 04F8  0122      LOGICAL OII=#96000000, NII=#94000000, CLII=#95000000;
06 000A  07 0504  0123      LOGICAL LAA=#41000000;
06 000A  07 0508  0124      LOGICAL TRUEE=#00010000;
06 000A  07 050C  0125      LOGICAL NULLST=#68000000, CONID=#5D;
06 000A  07 0514  0126      INTEGER CASE1=#10000, CASE2=#20000, CASE3=#30000, CASE4=#40000;
06 000A  07 0524  0127      INTEGER CASE5=#50000, CASE6=#60000;
06 000A  07 052C  0128      INTEGER SSIZE=_12;
06 000A  07 0530  0129      INTEGER NUMBYTESUBR=46;  COMMENT LENGTH OF IMPLICIT SUBR ENTRY;
06 000A  07 0534  0130      SHORT INTEGER BEGINN=83, NUMBER=86,
06 000A  07 0538  0131         BIT=98, STRYNG=99, TRUE=100, FALSE=101, NULLL=103, ID=87,
06 000A  07 0544  0132         FUNCID=90, ARRAYDC=105, ARRAYID=89, SEG=97,  RCCLID=91,
06 000A  07 054E  0133         RCCLDC=96, BBB=37, PROCDC=95, STACKADDR=77;
06 000A  07 0556  0134      BYTE  FORMALCALL SYN 1,   FPARCONV SYN 2,
06 000A  07 0556  0135           RECALLOCATE SYN 3,  REFCHECK SYN 4,     NULLREFARY SYN 5,
06 000A  07 0556  0136           ANALFN SYN 6,       GTIME SYN 7,
06 000A  07 0556  0137           READIN SYN 8,       WRITEEDIT SYN 9,
06 000A  07 0556  0138           REALINTEGER SYN 10, ARITHFN SYN 11,     CARITHFN SYN 12;
06 000A  07 0556  0139      SHORT INTEGER STFCASELOW=@READIN, STFCASEHIGH=@CARITHFN;
06 000A  07 055A  0140      SHORT INTEGER STFUNCID=107,STPROCID=108,ARSTAR=106,APRPAREN=29;
06 000A  07 0562  0141      SHORT INTEGER FLAG =#FFFFS, ENDCHAIN=1S;
06 000A  07 0566  0142      FUNCTION OI(4,#9600), XC(5,#D700);
06 000A  07 0566  0143      FUNCTION DECR(6,#0600);  COMMENT USE BCTR TO DECREMENT;
06 000A  07 0566  0144
06 000A  07 0566  0145      SHORT INTEGER SEQNO=0S;  COMMENT OBJECT CARD SEQ NO;
06 000A  07 0568  0146      INTEGER CARDNUM=0;
06 000A  07 056C  0147      INTEGER LOGPOINTER=0;
06 000A  07 0570  0148      SHORT INTEGER PUMREC=0,PMREC=0;
06 000A  07 0574  0149      INTEGER INSCOUNTER=0;
06 000A  07 0578  0150      INTEGER CODELENGTH=_256;     COMMENT TOTAL CODE LENGTH;
06 000A  07 057C  0151      INTEGER CLN=#D;  COMMENT CURRENT HEIRARCHY NUMBER;
06 000A  07 0580  0152      SHORT INTEGER LOGSEG=0S;  COMMENT LOGICAL SEGMENT NUMBER;
06 000A  07 0582  0153      SHORT INTEGER MAXLOGSEG=127S;
06 000A  07 0584  0154      SHORT INTEGER SEGTABIDX=0S, SEGTABMAX=124S;
06 000A  07 0588  0155      ARRAY 4 SHORT INTEGER F = (0S, 0S, 0S, 0S);
06 000A  07 0590  0156      ARRAY 14 SHORT INTEGER R = ( 2(#FFFFS), 12(0S) );
06 000A  07 05AC  0157      BYTE R3FLAG=#00X;
06 000A  07 05AD  0158      BYTE PAIRFLAG=#00X, DIVIDE=#00X, REMAINDER=#00X;
06 000A  07 05B0  0159      BYTE RECORDCLFLAG=#00X;
06 000A  07 05B1  0160         BYTE RECORDFLAG SYN RECORDCLFLAG;
06 000A  07 05B1  0161      BYTE CLFLAG=#00X, UJIFEXP=#00X, CEQ2=#00X, SUBARFLAG=#00X;
06 000A  07 05B5  0162      BYTE FIELDOP=#00X;  COMMENT SET IMMEDIATELY AFTER FIELD ACCESS;
06 000A  07 05B6  0163      ARRAY  132  BYTE OWBUF = 132(" ");
06 000A  07 063A  0164      ARRAY 16 BYTE  OTRTABLE=("0123456789ABCDEF");
06 000A  07 064A  0165      ARRAY 132 BYTE HEADING =
06 000A  07 06CE  0166         ("STANFORD ALGOL W  (VERSION 22OCT69) ",77(" "),"PAGE",15(" "));
06 000A  07 06CE  0167      BYTE COMPACTED=#00X;  COMMENT SET AFTER FIRST STORAGE COMPACTION;
06 000A  07 06CF  0168
06 000A  07 06CF  0169    PROCEDURE WRITE(R14);  COMMENT OUTPUT LINE AT (R0);
06 000A  07 06CF  0170       BEGIN  ARRAY 4 LOGICAL SAVE03;  STM(R0,R3,SAVE03);
06 0012  07 06E0  0171          R1 := R1-R1; IC(R1,CARRCONT); R2 := LINENO + 1; R3 := APUTLINE;
06 0024  07 06E0  0172          IF R1 = "0" THEN
06 002C  07 06E0  0173          BEGIN  IF R2 >= 60 THEN R1 := "1" ELSE R2 := R2 + 1;
06 004C  07 06E0  0174          END;
```

```
06 0040  07 06E0  0175          IF R1 = "1" THEN
06 0048  07 06E0  0176          BEGIN  R0 := PAGENO + 1; PAGENO := R0; CVD(R0,PKDEC);
06 0058  07 06E0  0177             MVC(3,HEADING(117),#40202120); ED(3,HEADING(117),PKDEC(6));
06 0064  07 06E0  0178             R0 := @HEADING; BALR(R2,R3);  R0 := SAVE03(0);
06 006E  07 06E0  0179             R2 := 3;  R1 := "0";
06 0076  07 06E0  0180          END;
06 0076  07 06E0  0181          LINENO := R2;
06 007A  07 06E0  0182          IF R2 = 60 THEN MVI("1",CARRCONT) ELSE MVI(" ",CARRCONT);
06 008E  07 06E0  0183          BALR(R2,R3);  COMMENT LINK TO WRITE;   LM(R0,R3,SAVE03);
06 0094  07 06E0  0184       END;
06 0096  07 06E0  0185
06 0096  07 06E0  0186    PROCEDURE EROR(R1); COMMENT THIS PROCEDURE PRINTS OUT PASS3 ERROR
06 0096  07 06E0  0187       MESSAGES, SETS R0 := 1 AND BRANCHES OUT OF PASS3. R3 IS PARAMETER;
06 0096  07 06E0  0188       BEGIN  MVC(59,OWBUF(4),OWBUF);  MVC(4,OWBUF(4),"*****");
06 00A2  07 06E0  0189          MVI("1",CARRCONT);
06 00A6  07 06E0  0190          CASE R3 OF
06 00A6  07 06E0  0191          BEGIN
06 00A6  07 06E0  0192             MVC(23,OWBUF(11),"PROGRAM SEGMENT OVERFLOW");
06 00B4  07 06E0  0193             MVC(22,OWBUF(11),"COMPILER STACK OVERFLOW");
06 00BE  07 06E0  0194             MVC(31,OWBUF(11),"CONSTANT POINTER TABLE TOO LARGE");
06 00C8  07 06E0  0195             MVC(23,OWBUF(11),"BLOCKS NESTED TOO DEEPLY");
06 00D2  07 06E0  0196             MVC(20,OWBUF(11),"DATA SEGMENT OVERFLOW");
06 00DC  07 06E0  0197             MVC(20,OWBUF(11),"CARD TABLES TOO LARGE");
06 00E6  07 06E0  0198             MVC(18,OWBUF(11),"TOO MANY PROCEDURES");
06 00F0  07 06E0  0199             MVC(20,OWBUF(11),"PROGRAM AREA OVERFLOW");
06 00FA  07 06E0  0200          END;
06 011E  07 06E0  0201          R0 := @OWBUF;       WRITE;  R14 := SAVE14;
06 012A  07 06E0  0202          R1 := CARDNUM;     MVC(54,OWBUF(10),OWBUF(9));
06 0134  07 06E0  0203          MVC(8,OWBUF(11),"NEAR CARD");  CVD(R1,PKDEC);
06 013E  07 06E0  0204          UNPK(3,7,OWBUF(21),PKDEC);  OI(#F0,OWBUF(24));
06 0148  07 06E0  0205          R0 := @OWBUF;     WRITE;  R14 := SAVE14;
06 0154  07 06E0  0206          R0 := 16;     GOTO ERXIT;
06 015C  07 06E0  0207       END;
06 015E  07 06E0  0208
06 015E  07 06E0  0209    PROCEDURE INCRADDR(R1); COMMENT INCREASES NEXTADDR BY NUMBER OF BYTES
06 015E  07 06E0  0210       SPECIFIED IN RA;
06 015E  07 06E0  0211       BEGIN   R10 := R10 + NEXTADDR; NEXTADDR := R10;
06 0166  07 06E0  0212       END;
06 0168  07 06E0  0213
06 0168  07 06E0  0214    PROCEDURE EMIT (R1); COMMENT TAKES ASSEMBLED INSTRUCTION FROM R0
06 0168  07 06E0  0215       (AND R3 IF 6 BYTE INSTRUCTION), WRITES, AND UPDATES INSTRUCTION
06 0168  07 06E0  0216       COUNTER. ALL INSTRUCTIONS MUST BE LEFT JUSTIFIED;
06 0168  07 06E0  0217       BEGIN
06 0168  07 06E0  0218       PROCEDURE WRITEINSTRUCTION(R1);
06 0168  07 06E0  0219          BEGIN ARRAY  4  INTEGER TEMP;
06 016C  07 06F0  0220             BYTE SEGNO SYN OWBUF (12);
06 016C  07 06F0  0221             BYTE LC SYN OWBUF(21);
06 016C  07 06F0  0222             BYTE MNEM SYN OWBUF(32);
06 016C  07 06F0  0223             BYTE INS SYN OWBUF (39);
06 016C  07 06F0  0224             STM(R0,R3,TEMP);  MVC(59,OWBUF(4),OWBUF);
06 0176  07 06F0  0225             R1 := R0 SHRL 24 SHLL 2;  R2 := MNBASE + R1;  MVC(3,MNEM,B2);
06 018C  07 06F0  0226             R1 := R1 SHRL 8; UNPK(4,2,INS,TEMP);
06 0196  07 06F0  0227             TR(3,INS,OTRTABLE(_240));
06 019C  07 06F0  0228             IF R1 = 0 THEN MVI(" ",INS(4) )ELSE
06 01A6  07 06F0  0229             BEGIN UNPK(4,2,INS(4),TEMP(2));
06 01B0  07 06F0  0230                TR(3,INS(4),OTRTABLE(_240));
06 01B6  07 06F0  0231                IF R1 < 3 THEN MVI(" ",INS(8) )ELSE
06 01C2  07 06F0  0232                BEGIN UNPK(4,2,INS(8),TEMP(12));  MVI(" ",INS(12));
```

```
06 01D0  07 06F0  0233                  TR(3,INS(8),OTRTABLE(_240));
06 01D6  07 06F0  0234               END;
06 01D6  07 06F0  0235            END;
06 01D6  07 06F0  0236            R1 := CARDNUM;     R2 := INSCOUNTER;
06 01DE  07 06F0  0237            CVD(R1,PKDEC);  UNPK(3,7,SEGNO,PKDEC);  OI(#F0,SEGNO(3));
06 01EC  07 06F0  0238            WORK := R2; UNPK(4,2,LC,WORK(2));
06 01F6  07 06F0  0239            TR(3,LC,OTRTABLE(_240));  MVI(" ",LC(4));
06 0200  07 06F0  0240            R0 := @OWBUF;     WRITE;  R14 := SAVE14;     LM(R0,R3,TEMP);
06 021C  07 06F0  0241         END;
06 0212  07 06F0  0242
06 0212  07 06F0  0243         ARRAY 2 INTEGER SAVEREG; INTEGER SAVER1;
06 0212  07 06FC  0244            ARRAY 3 SHORT INTEGER INSTRUCTION SYN SAVEREG(0);
06 0212  07 06FC  0245         ARRAY 3 SHORT INTEGER MOVECODE=(#D200,@B3,@INSTRUCTION);
06 0212  07 0702  0246         SAVER1 := R1; IF PRINT THEN WRITEINSTRUCTION;
06 0222  07 0702  0247         R1 := R3; STM(R0,R1,SAVEREG);
06 0228  07 0702  0248         IC(R1,INSTRUCTION(0));  R1 := R1 AND #C0 SHRL 5;
06 0234  07 0702  0249         IF R1 < 4 THEN R1 := R1 + 2;  COMMENT R1 = INSTRUCTION LENGTH;
06 0240  07 0702  0250         IF FIELDOP THEN
06 0248  07 0702  0251         BEGIN  RESET(FIELDOP);  IF R1 ¬= 2 THEN
06 0254  07 0702  0252            BEGIN  R3 := INSCOUNTER - 4;  R0 := PROGRAM(R3) AND #FF0F;
06 0264  07 0702  0253               IF R0 = #4100 THEN
06 026C  07 0702  0254               BEGIN  COMMENT LA;
06 026C  07 0702  0255                  R0 := PROGRAM(R3) AND #F0 SHLL 8;
06 0278  07 0702  0256                  IF R0 = INSTRUCTION(2) THEN
06 0280  07 0702  0257                  BEGIN  COMMENT REPLACE WITH DISPLACEMENT;
06 0280  07 0702  0258                     INSCOUNTER := R3;
06 0284  07 0702  0259                     R0 := PROGRAM(R3+2); INSTRUCTION(2) := R0;
06 028C  07 0702  0260                  END;
06 028C  07 0702  0261               END;
06 028C  07 0702  0262            END;
06 028C  07 0702  0263         END;
06 028C  07 0702  0264         R3 := INSCOUNTER; R0 := R3 + R1; INSCOUNTER := R0;
06 0298  07 0702  0265         IF R0 > 8192 THEN
06 02A0  07 0702  0266         BEGIN  R3 := 1; EROR;
06 02A8  07 0702  0267         END;
06 02A8  07 0702  0268         IF R0 >= R8 THEN
06 02AE  07 0702  0269         BEGIN  R3 := 2; EROR;
06 02B6  07 0702  0270         END;
06 02B6  07 0702  0271         R3 := @PROGRAM(R3); DECR(R1); EX(R1,MOVECODE);
06 02C0  07 0702  0272         LM(R0,R1,SAVEREG); R3 := R1; R1 := SAVER1;
06 02CA  07 0702  0273      END;
06 02CC  07 0702  0274
06 02CC  07 0702  0275      PROCEDURE CHAINOUT(R1);
06 02CC  07 0702  0276         BEGIN  COMMENT
06 02CC  07 0702  0277            INPUT - R0 = SEG NO,  OUTPUT - R0 = SEG CHAIN;
06 02CC  07 0702  0278            ARRAY 2 LOGICAL SAVE12; STM(R1,R2,SAVE12);
06 02D0  07 070C  0279            FOR R2 := 4 STEP 4 UNTIL SEGTABIDX DO
06 02D4  07 070C  0280            IF R0 = SEGNO(R2) THEN
06 02E0  07 070C  0281            BEGIN  R0 := CHAIN(R2); GOTO X;
06 02E8  07 070C  0282            END;
06 02F4  07 070C  0283            R2 := SEGTABIDX + 4; SEGTABIDX := R2;
06 0300  07 070C  0284            IF R2 > SEGTABMAX OR R0 > MAXLOGSEG THEN
06 0310  07 070C  0285            BEGIN  R3 := 7; EROR;
06 0318  07 070C  0286            END;
06 0318  07 070C  0287            SEGNO(R2) := R0; R0 := ENDCHAIN;
06 0320  07 070C  0288      X:    R1 := INSCOUNTER; CHAIN(R2) := R1; LM(R1,R2,SAVE12);
06 032C  07 070C  0289         END;
06 032E  07 070C  0290
```

```
06 C32E  07 070C  0291    PROCEDURE CHAININ(R1);
06 032E  07 070C  0292       BEGIN COMMENT EMITS CODE TO RESTORE CURRENT SEG BASE;
06 032E  07 070C  0293          LOGICAL SAVER1; SAVER1 := R1;
06 0332  07 0710  0294          R1 := INSCOUNTER; R0 := CHAIN(0) OR LOADMARK;
06 033E  07 0710  0295          CHAIN(0) := R1; EMIT; R1 := SAVER1;
06 034A  07 0710  0296       END;
06 034C  07 0710  0297
06 034C  07 0710  0298    PROCEDURE EMITCALL(R1);
06 034C  07 0710  0299       BEGIN COMMENT  EMITS CALL AND RELOAD TO SEG NO IN R0;
06 034C  07 0710  0300          ARRAY 2 LOGICAL SAVE01; STM(R0,R1,SAVE01);
06 0350  07 0718  0301          CHAINOUT; R0 := R0 OR #58F00000; EMIT;
06 035C  07 0718  0302          R0 := #051F0000; EMIT; R0 := ABS SAVE01(0);
06 036A  07 0718  0303          IF R0 <= STFCASEHIGH AND R0 >= STFCASELOW THEN
06 037A  07 0718  0304          BEGIN  R0 := R3; EMIT;
06 0380  07 0718  0305          END;
06 0380  07 0718  0306          CHAININ; LM(R0,R1,SAVE01);
06 0388  07 0718  0307       END;
06 038A  07 0718  0308
06 038A  07 0718  0309    PROCEDURE ASSEMBLE(R1);
06 038A  07 0718  0310       COMMENT  ASSEMBLES AND EMITS INSTRUCTION ACCORDING TO TYPE. INPUTS:
06 038A  07 0718  0311         R6 = TYPE, R0(8:11/8:15) = FIRST OPERAND REGISTER/REGISTERS,
06 038A  07 0718  0312         R3(16:31) = SECOND OPERAND (R3>0) OR R3(8:11/8:15) = SECOND REG,
06 038A  07 0718  0313         R0(4:7) = OPERATION.   R0, R1 CHANGED;
06 038A  07 0718  0314       BEGIN  LOGICAL SAVER1, SAVER6, TEMP;  SAVER1 := R1; SAVER6 := R6;
06 0392  07 0724  0315          IF R6 > 5 THEN R6 := 1;
06 039E  07 0724  0316          IF R6 >= 4 THEN
06 03A6  07 0724  0317          BEGIN  COMMENT  COMPLEX, SAVE SECOND INSTRUCTION FIELDS;
06 03A6  07 0724  0318             R1 := R0 SHLL 4 AND #00F00000;  TEMP := R1;
06 03B4  07 0724  0319             IF R3 < 0 THEN R1 := R3 AND #000F0000  ELSE
06 03C0  07 0724  0320             IF R6 = 4 THEN R1 := R3 + 4  ELSE  R1 := R3 + 8;
06 03DC  07 0724  0321             R1 := R1 OR TEMP;  TEMP := R1;
06 03E4  07 0724  0322          END;
06 03E4  07 0724  0323          IF R3 >= 0 THEN R0 := R0 OR #40000000;  COMMENT SELECT RR OR RX;
06 03EE  07 0724  0324          R1 := R6 SHLA 2;  R0 := R0 OR OPTYPE(R1-4);  COMMENT TYPE CODE;
06 03F8  07 0724  0325          R0 := R0 AND #7FF00000;  COMMENT OPERATION, FIRST REGISTER;
06 03FC  07 0724  0326          IF R3 >= 0 THEN R0 := R0 OR R3  ELSE
06 0404  07 0724  0327          BEGIN  R1 := R3 AND #00F00000 SHRL 4;  R0 := R0 OR R1;
06 0414  07 0724  0328          END;
06 0414  07 0724  0329          EMIT;     IF R6 >= 4 THEN
06 0420  07 0724  0330          BEGIN  R0 := R0 AND #FF000000 OR TEMP;  EMIT;
06 042C  07 0724  0331          END;
06 042C  07 0724  0332          R6 := SAVER6;  R1 := SAVER1;
06 0434  07 0724  0333       END;
06 0436  07 0724  0334
06 0436  07 0724  0335    PROCEDURE WRITENUMBER(R1);  COMMENT THIS PROCEDURE WRITES OUT A 4 BYTE
06 0436  07 0724  0336       NUMBER FOUND IN R0 AND UPDATES INSCOUNTER BY 4;
06 0436  07 0724  0337       BEGIN  ARRAY 2 LOGICAL TEMP;  INTEGER NUMB SYN TEMP(0);
06 0436  07 072C  0338          STM(R0,R1,TEMP);  CLI(#00,TRACEIT);  IF ¬= THEN
06 0442  07 072C  0339          BEGIN  R1 := NUMSTACKP; R0 := NUMORIGIN(R1) + NUMSEQLEN(R1) + 4;
06 0452  07 072C  0340             IF R0 = INSCOUNTER THEN
06 045A  07 072C  0341             BEGIN   R0 := NUMSEQLEN(R1) + 4;  NUMSEQLEN(R1) := R0;
06 0466  07 072C  0342             END ELSE
06 0466  07 072C  0343             BEGIN   R1 := R1 + 4; NUMSTACKP := R1;
06 0472  07 072C  0344                R0 := INSCOUNTER;  NUMSTACK(R1) := R0;
06 047A  07 072C  0345             END;
06 047A  07 072C  0346          END;
06 047A  07 072C  0347          R1 := INSCOUNTER; R1 := @PROGRAM(R1);  MVC(3,B1,NUMB);
06 0488  07 072C  0348          IF PRINT THEN
```

```
06 0490  07 072C  0349          BEGIN  MVC(59,OWBUF(4),OWBUF); UNPK(8,4,OWBUF(39),NUMB);
06 049C  07 072C  0350             MVI(" ",OWBUF(47)); TR(7,OWBUF(39),OTRTABLE(_240));
06 04A6  07 072C  0351             UNPK(4,2,OWBUF(21),INSCOUNTER(2)); MVI(" ",OWBUF(25));
06 04B0  07 072C  0352             TR(3,OWBUF(21),OTRTABLE(_240));
06 04B6  07 072C  0353             R0 := @OWBUF;  WRITE;  R14 := SAVE14;
06 04C2  07 072C  0354          END;
06 04C2  07 072C  0355          R0 := INSCOUNTER + 4; INSCOUNTER := R0;
06 04CE  07 072C  0356          IF R0 > 8192 THEN BEGIN R3 := 1; EROR; END;
06 04DE  07 072C  0357          IF R0 >= R8 THEN BEGIN R3 := 2; EROR; END;
06 04EC  07 072C  0358          LM(R0,R1,TEMP);
06 04F0  07 072C  0359       END;
06 04F2  07 072C  0360
06 04F2  07 072C  0361    PROCEDURE LMINS(R1); COMMENT THIS PROCEDURE EMITS THE L OR LM
06 04F2  07 072C  0362       INSTRUCTION WHICH UPDATES THE DISPLAY AFTER RETURNING
06 04F2  07 072C  0363       TO A PROCEDURE;
06 04F2  07 072C  0364       BEGIN INTEGER SAVER1;  SAVER1 := R1;
06 04F6  07 0730  0365          R1 := CLN;
06 04FA  07 0730  0366          IF R1 <= 11 THEN
06 0502  07 0730  0367          BEGIN  IF = THEN R0 := #58B02000 ELSE
06 050A  07 0730  0368                            R0 := R1 SHLL 20 OR #980B2000;
06 0518  07 0730  0369             R0 := R0 OR DSP;  EMIT;
06 0520  07 0730  0370          END;
06 0520  07 0730  0371          R1 := SAVER1;
06 0524  07 0730  0372       END;
06 0526  07 0730  0373
06 0526  07 0730  0374    PROCEDURE REFSTOR(R1);  COMMENT STORING A REFERENCE - CHECK
06 0526  07 0730  0375       FOR INTERRUPT.  R10=4 IF ST , R10=6 IF MVC  ;
06 0526  07 0730  0376       BEGIN INTEGER SAVER1;   SAVER1 := R1;
06 052A  07 0734  0377          R1 := INSCOUNTER - R10; R10 := PROGRAM(R1+2) AND MASK;
06 0538  07 0734  0378          R1 := R10 AND MASK8; IF R1 = 0 OR R10 >= LOINTREF THEN
06 054C  07 0734  0379          BEGIN  R0 := #41000000 OR R10; EMIT;
06 0556  07 0734  0380             R0 := @REFCHECK; R0 := NEG R0; EMITCALL;
06 0560  07 0734  0381          END; R1 := SAVER1;
06 0564  07 0734  0382       END;
06 0566  07 0734  0383
06 0566  07 0734  0384    PROCEDURE PROCCALLCODE(R1);  COMMENT EMITS THE CODE FOR
06 0566  07 0734  0385       A PROCEDURE CALL;
06 0566  07 0734  0386       BEGIN INTEGER SAVER1;    SAVER1 := R1;
06 056A  07 0738  0387          R1 := 0;  IC(R1,TYPE(R7));
06 0572  07 0738  0388          IF R1 = 19 THEN
06 057A  07 0738  0389          BEGIN COMMENT FORMAL PROCEDURE;
06 057A  07 0738  0390             IF R3=0 THEN R0:=#18000000 ELSE R0:=#41000000 OR R3;  EMIT;
06 0592  07 0738  0391             R0 := IDLOC1(R7) SHLL 12 + IDLOC2(R7) OR #98450000;  EMIT;
06 05A6  07 0738  0392             R0 := #18340000;  EMIT;
06 05AE  07 0738  0393             R0 := @FORMALCALL; R0 := NEG R0; CHAINOUT;
06 05B8  07 0738  0394             R0 := R0 OR #58F00000;
06 05BC  07 0738  0395          END ELSE
06 05BC  07 0738  0396          BEGIN  R0 := IDLOC2(R7) AND #FF00 + #100 SHLL 8 OR #18500000;
06 05D4  07 0738  0397             EMIT; R0 := IDLOC2(R7) AND #FF;
06 05E0  07 0738  0398             IF R0 = LOGSEG THEN R0 := #18FE0000 ELSE
06 05EC  07 0738  0399             BEGIN  CHAINOUT; R0 := R0 OR #58F00000;
06 05F8  07 0738  0400             END;
06 05F8  07 0738  0401          END;
06 05F8  07 0738  0402          EMIT;  R0 := #051F0000; EMIT; LMINS; CHAININ;  R1 := SAVER1;
06 0610  07 0738  0403       END;
06 0612  07 0738  0404
06 0612  07 0738  0405    PROCEDURE WRITEALL(R1); COMMENT THIS PROCEDURE WRITES OUT THE CODE
06 0612  07 0738  0406       GENERATED FOR AN ENTIRE PROGRAM SEGMENT;
```

```
06 0612  07 0738  0407      BEGIN  LOGICAL SAVER1, NUMINFO;  INTEGER INSC;
06 0612  07 0744  0408         SHORT INTEGER NUMBYT SYN NUMINFO(0), INUM SYN NUMINFO(2);
06 0612  07 0744  0409         SAVER1 := R1;  MVI(#00,TRACEIT);  SET(PRINT);
06 061E  07 0744  0410         R0 := NUMSTACK(4);  NUMINFO := R0;
06 0626  07 0744  0411         R0 := 4;  R1 := NUMSTACKP; NUMSTACK(R1+4) := R0; NUMSTACKP := R0;
06 0632  07 0744  0412         R0 := RUNERRSTART + 12; WRITRUNPT := R0;
06 0642  07 0744  0413         R0 := INSCOUNTER;  INSC := R0;  R10 := 0;  INSCOUNTER := R10;
06 0652  07 0744  0414         WHILE R10 < INSC DO
06 065A  07 0744  0415         BEGIN     IF R10 = INUM THEN
06 0662  07 0744  0416            BEGIN COMMENT OUTPUT A SEQUENCE OF NUMBERS;
06 0662  07 0744  0417               FOR R3 := 0 STEP 4 UNTIL NUMBYT DO
06 0666  07 0744  0418               BEGIN  R1 := INSCOUNTER;  R0 := PROGRAM(R1) SHLL 16;
06 0676  07 0744  0419                  R1 := PROGRAM(R1+2) AND #FFFF;
06 067E  07 0744  0420                  R0 := R0 OR R1;  WRITENUMBER;
06 0684  07 0744  0421               END;
06 0690  07 0744  0422               R1 := NUMSTACKP + 4; NUMSTACKP := R1;
06 069C  07 0744  0423               R0 := NUMSTACK(R1);  NUMINFO := R0;
06 06A4  07 0744  0424            END ELSE
06 06A4  07 0744  0425            BEGIN
06 06A8  07 0744  0426               R3 := WRITRUNPT;  R1 := B3 SHRL 16; IF R1 = R10 THEN
06 06BA  07 0744  0427               BEGIN  R1 := B3 AND MASK;  CARDNUM := R1;
06 06C6  07 0744  0428                  R3 := @B3(4); WRITRUNPT := R3;
06 06CE  07 0744  0429               END;
06 06CE  07 0744  0430               R3 := PROGRAM(R10) SHLL 16;
06 06D6  07 0744  0431               R0 := PROGRAM(R10+2) AND MASK OR R3;
06 06EC  07 0744  0432               R3 := PROGRAM(R10+4) SHLL 16;      EMIT;
06 06EC  07 0744  0433            END;
06 06EC  07 0744  0434               R10 := INSCOUNTER;
06 06F0  07 0744  0435            END;
06 06F4  07 0744  0436            RESET(PRINT);  R1 := SAVER1;
06 06FC  07 0744  0437         END;
06 06FE  07 0744  0438
06 06FE  07 0744  0439      PROCEDURE GETTYPE(R1); COMMENT TAKES SIMPLE TYPE FROM NAME TABLE
06 06FE  07 0744  0440         AND PUTS IT IN LOW ORDER BYTE OF R3;
06 06FE  07 0744  0441         BEGIN            R3 := R7 SHRL 24;
06 0704  07 0744  0442            IF R3 = NUMBER THEN
06 070C  07 0744  0443            BEGIN R3 := R7 AND MASK + CPTBASE;  R3 := CONSPTTAB(R3) SHRL 16;
06 071E  07 0744  0444            END ELSE
06 071E  07 0744  0445            IF R3 = TRUE OR R3 = FALSE THEN R3 := 6 ELSE
06 0736  07 0744  0446            IF R3 = STRYNG THEN R3 := 7 ELSE
06 0746  07 0744  0447            IF R3 = BIT THEN R3 := 8 ELSE
06 0756  07 0744  0448            IF R3 = NULLL OR R3 = RCCLID  THEN R3 := 9 ELSE
06 076E  07 0744  0449            BEGIN  R3 := R7 AND #FFFF;  IC(R3,SIMPLETYPE(R3));
06 077C  07 0744  0450               R3 := R3 AND #FF;
06 0780  07 0744  0451            END;
06 0780  07 0744  0452         END;
06 0782  07 0744  0453
06 0782  07 0744  0454      PROCEDURE DUMPFLREG(R1); COMMENT WRITES INSTRUCTION TO DUMP OLDEST
06 0782  07 0744  0455         FLOATING REGISTER VALUE INTO LOCAL STACK, MAKES ADJUSTMENTS IN
06 0782  07 0744  0456         FSTACK AND LSTACK, PUTS REGISTER NUMBER IN R3, BITS 8 TO 11;
06 0782  07 0744  0457         BEGIN LOGICAL SAVER1, SAVER4, SAVERA;
06 0782  07 0750  0458            SAVER1 := R1; SAVER4 := R4; SAVERA := R10;
06 078E  07 0750  0459            MVC(11,FSTACK,FSTACK(4));
06 0794  07 0750  0460            R10 := R0 AND MASK;
06 079A  07 0750  0461            R4 := NEXTADDR;
06 079E  07 0750  0462            R3 := LSTACK(R10) SHLL 1 SHRL 1;
06 07AA  07 0750  0463            R0 := R0 AND MASK1 SHLL 1;
06 07B2  07 0750  0464            IF R0 < 0 THEN
```

```
06 07B8  07 0750  0465          BEGIN COMMENT SHORT;
06 07B8  07 0750  0466             R4 := R4 + 3 AND SMASK;  LSTACK(R10) := R4;
06 07C4  07 0750  0467             R0 := R3 OR R4 OR #70000000;
06 07CC  07 0750  0468             R4 := R4+4;
06 07D0  07 0750  0469          END ELSE
06 07D0  07 0750  0470          BEGIN COMMENT LONG;
06 07D4  07 0750  0471             R4 := R4 + 7 AND #FFFFFFF8;  LSTACK(R10) := R4;
06 07E0  07 0750  0472             R0 := R3 OR R4 OR #60000000;
06 07E8  07 0750  0473             R4 := R4+8;
06 07EC  07 0750  0474          END;
06 07EC  07 0750  0475          NEXTADDR := R4;
06 07F0  07 0750  0476          EMIT;
06 07F4  07 0750  0477          R1 := SAVER1; R4 := SAVER4-4;
06 0800  07 0750  0478          R10 := SAVERA;
06 0804  07 0750  0479       END;
06 0806  07 0750  0480
06 0806  07 0750  0481    PROCEDURE DUMPPRFLREG(R1); COMMENT WRITES INSTRUCTIONS TO DUMP OLDEST
06 0806  07 0750  0482       VALUE , WHICH IS COMPLEX, IN LOCAL STACK.  UPDATES LSTACK AND
06 0806  07 0750  0483       FSTACK, PUTS REGISTER NUMBERS IN BITS 8 TO 11 AND 12 TO 15 OF R3;
06 0806  07 0750  0484       BEGIN LOGICAL SAVER1, SAVER4;
06 0806  07 0758  0485          LOGICAL SAVERA; SAVERA := R10;
06 080A  07 075C  0486          SAVER1 := R1; SAVER4 := R4;
06 0812  07 075C  0487          MVC(11,FSTACK,FSTACK(4));
06 0818  07 075C  0488          R10 := R0 AND MASK; R4 := NEXTADDR;
06 0822  07 075C  0489          R3 := LSTACK(R10);
06 0826  07 075C  0490          R0 := R0 AND MASK1 SHLL 1; IF R0 < 0 THEN
06 0834  07 075C  0491          BEGIN COMMENT SHORT;
06 0834  07 075C  0492             R4 := R4 + 3 AND SMASK;  LSTACK(R10) := R4;
06 0840  07 075C  0493             R10 := R3 AND MASK5;
06 0846  07 075C  0494             R0 := R10 OR R4 OR #70000000; EMIT; R4 := R4+4;
06 0856  07 075C  0495             R10 := R3 AND MASK4 SHLL 4;
06 0860  07 075C  0496             R0 := R10 OR R4 OR #70000000; R4 := R4+4;
06 086C  07 075C  0497          END ELSE
06 086C  07 075C  0498          BEGIN  COMMENT LONG;
06 0870  07 075C  0499             R4 := R4 + 7 AND #FFFFFFF8;  LSTACK(R10) := R4;
06 087C  07 075C  0500             R10 := R3 AND MASK5;
06 0882  07 075C  0501             R0 := R10 OR R4 OR #60000000; EMIT; R4 := R4+8;
06 0892  07 075C  0502             R10 := R3 AND MASK4 SHLL 4;
06 089C  07 075C  0503             R0 := R10 OR R4 OR #60000000; R4 := R4+8;
06 08A8  07 075C  0504          END;
06 08A8  07 075C  0505          NEXTADDR := R4; EMIT;
06 08BC  07 075C  0506          R1 := SAVER1; R4 := SAVER4-4;
06 08BC  07 075C  0507          R10 := SAVERA;
06 08C0  07 075C  0508       END;
06 08C2  07 075C  0509
06 08C2  07 075C  0510    PROCEDURE FLREG(R1); COMMENT MAKES AVAILABLE A FLOATING REGISTER,
06 08C2  07 075C  0511       LEAVING ITS NUMBER IN BITS 8 TO 11 OF R0.  FSTACK AND LSTACK ARE
06 08C2  07 075C  0512       UPDATED. R0 IS AN INPUT WITH ITS SIGN BIT TO INDICATE REAL(1) OR
06 08C2  07 075C  0513       COMPLEX(0) AND THE ADJACENT BIT TO INDICATE SHORT(1) OR LONG(0);
06 08C2  07 075C  0514       BEGIN LOGICAL SAVER0, SAVER1;
06 08C2  07 0764  0515          SAVER0 := R0; SAVER1 := R1;
06 08CA  07 0764  0516          FOR R3 := 0 STEP 2 UNTIL 6 DO
06 08CE  07 0764  0517          BEGIN R0 := F(R3); IF R0 >= 0 THEN
06 08DC  07 0764  0518             BEGIN COMMENT REGISTER FREE;
06 08DC  07 0764  0519                R0 := FLAG; F(R3) := R0;
06 08E4  07 0764  0520                R3 := R3 SHLA 20; GOTO FLR1;
06 08EC  07 0764  0521             END;
06 08EC  07 0764  0522          END;
```

```
06 08F8  07 0764   0523          COMMENT NO REGISTER FREE;
06 08F8  07 0764   0524          R0 := FSTACK(0);
06 08FC  07 0764   0525          IF R0 < 0 THEN DUMPFLREG ELSE
06 09C6  07 0764   0526          BEGIN
06 09CA  07 0764   0527             DUMPPRFLREG;
06 09CE  07 0764   0528             R1 := R3 AND MASK4 SHRL 16; R0 := 0; F(R1) := R0;
06 09D2  07 0764   0529             R3 := R3 AND MASK5;
06 09D4  07 0764   0530          END;
06 09D4  07 0764   0531    FLR1: R0 := R3;
06 09D6  07 0764   0532          R3 := R3 OR SIGN;
06 09DA  07 0764   0533          LSTACK(R5) := R3;
06 09DE  07 0764   0534          R3 := R5 OR SAVER0;
06 09E4  07 0764   0535          FSTACK(R4) := R3;
06 09E8  07 0764   0536          R4 := R4+4; R5 := R5+4;
06 09EC  07 0764   0537          R1 := SAVER1;
06 09F4  07 0764   0538       END;
06 09F6  07 0764   0539
06 09F6  07 0764   0540    PROCEDURE PRFLREG(R1); COMMENT MAKES AVAILABLE A PAIR OF FLOATING
06 09F6  07 0764   0541       REGISTERS: LEAVING THEIR NUMBERS IN BITS 8 TO 11 AND 12 TO 15 OF R0.
06 09F6  07 0764   0542       OTHERWISE SIMILAR TO FLREG;
06 09F6  07 0764   0543       BEGIN
06 09F6  07 0764   0544          LOGICAL SAVER0, SAVER1; SAVER0 := R0; SAVER1 := R1;
06 09FE  07 076C   0545          R10 := 0;
06 0A02  07 076C   0546    PRF5: FOR R3 := 0 STEP 2 UNTIL 6 DO
06 0A06  07 076C   0547          BEGIN R0 := F(R3); IF R0 >= 0 THEN
06 0A14  07 076C   0548             BEGIN R0 := FLAG; F(R3) := R0; GOTO PRF1; END;
06 0A7C  07 076C   0549          END;
06 0A7C  07 076C   0550    PRF3: R0 := FSTACK(0);
06 0A80  07 076C   0551          IF R0 < 0 THEN
06 0A86  07 076C   0552          BEGIN DUMPFLREG; IF R10 < 0 THEN
06 0A90  07 076C   0553             BEGIN R3 := R3 SHRA 4; GOTO PRF2; END ELSE
06 0A98  07 076C   0554             BEGIN R10 := R3 OR SIGN; GOTO PRF3; END;
06 0AA6  07 076C   0555          END ELSE
06 0AA6  07 076C   0556          BEGIN DUMPPRFLREG; IF R10 < 0 THEN
06 0AB4  07 076C   0557             BEGIN R3 := R3 SHRA 4; R1 := R3 AND MASK SHRA 12; R0 := 0;
06 0AC6  07 076C   0558                F(R1) := R0; R3 := R3 AND MASK1; GOTO PRF2; END ELSE
06 0AD2  07 076C   0559             BEGIN R0 := R3 OR SIGN; GOTO PRF4; END;
06 0AE0  07 076C   0560          END;
06 0AE0  07 076C   0561    PRF1: IF R10 >= 0 THEN
06 0AE6  07 076C   0562          BEGIN R10 := R3 SHLA 20 OR SIGN; GOTO PRF5; END ELSE
06 0AF4  07 076C   0563          R3 := R3 SHLA 16;
06 0AFC  07 076C   0564    PRF2: R0 := R10 OR R3;
06 0B00  07 076C   0565    PRF4: LSTACK(R5) := R0; R3 := R5 OR SAVER0; FSTACK(R4) := R3;
06 0B0E  07 076C   0566          R4 := R4+4; R5 := R5+4;
06 0B16  07 076C   0567          R1 := SAVER1;
06 0B1A  07 076C   0568       END;
06 0B1C  07 076C   0569
06 0B1C  07 076C   0570    SEGMENT PROCEDURE  LEVELA(R1);
06 0B1C  07 076C   0571     BEGIN  LOGICAL SAVER1;
09 0000  07 0770   0572
09 0000  07 0770   0573     PROCEDURE DUMPALLFLREG(R1); COMMENT WRITES INSTRUCTIONS TO DUMP ALL
09 0000  07 0770   0574        OCCUPIED FLOATING REGISTERS INTO THE LOCAL STACK, CLEARS F AND
09 0000  07 0770   0575        FSTACK, UPDATES LSTACK;
09 0000  07 0770   0576        BEGIN
09 0004  07 0770   0577           LOGICAL SAVER0,SAVER1,SAVER3;  STM(R0,R1,SAVER0); SAVER3:=R3;
09 000C  07 077C   0578           WHILE R4 > 0 DO
09 0012  07 077C   0579           BEGIN
09 0012  07 077C   0580              R0 := FSTACK(0);
```

```
09 0016  07 077C  0581          IF R0 < 0 THEN DUMPFLREG ELSE DUMPPRFLREG;
09 0038  07 077C  0582        END;
09 003C  07 077C  0583        FOR R3 := 0 STEP 2 UNTIL 6 DO F(R3) := R4;
09 0054  07 077C  0584        LM(R0,R1,SAVER0);  R3 := SAVER3;
09 005C  07 077C  0585      END;
09 005E  07 077C  0586
09 005E  07 077C  0587    PROCEDURE RELEASE(R1); COMMENT ADJUSTS THE REGISTER ARRAYS TO EFFECT
09 005E  07 077C  0588       THE FREEING OF A REGISTER OR PAIR OF REGISTERS.  R3 IS AN INPUT  IN
09 005E  07 077C  0589       THE SAME FORMAT AS FOR ASSEMBLE.  TYPE IN LOW BITS OF R6;
09 005E  07 077C  0590       BEGIN LOGICAL SAVER3, SAVER0;
09 005E  07 0784  0591          LOGICAL SAVER6,SAVERA;  SAVER6 := R6;  SAVERA := R10;
09 0066  07 078C  0592          SAVER3 := R3; SAVER0 := R0; R0 := 0;
09 0072  07 078C  0593          IF R6 > 5 THEN R6 := 1;
09 007E  07 078C  0594          R10 := R3 SHRL 12;
09 0084  07 078C  0595          IF R3 >= 0 AND R10 >= CLN THEN NULL ELSE IF R3 >=0 THEN
09 009C  07 078C  0596          BEGIN R3 := R10 + R10; R(R3) := R0; R2 := R2-4; END ELSE
09 00A8  07 078C  0597          IF R6 = 1 THEN
09 00B4  07 078C  0598          BEGIN
09 00B4  07 078C  0599             R10 := R3 SHLL 12; IF R10 ¬= 0 THEN
09 00C0  07 078C  0600             BEGIN R10 := R10 SHRL 27; R(R10) := R0;
09 00C8  07 078C  0601                R3 := R3 AND MASK5 SHRL 19; R(R3) := R0;
09 00D4  07 078C  0602             END ELSE
09 00D4  07 078C  0603             BEGIN R3 := R3 SHLL 1 SHRL 20; R(R3) := R0; END;
09 00E4  07 078C  0604             R2 := R2-4;
09 00E8  07 078C  0605          END ELSE
09 00E8  07 078C  0606          IF R6 < 4 THEN
09 00F4  07 078C  0607          BEGIN R3 := R3 SHLL 1 SHRL 21; F(R3) := R0; R4 := R4-4; END ELSE
09 01C4  07 078C  0608          BEGIN R3 := R3 AND MASK4 SHRL 16; F(R3) := R0;
09 0114  07 078C  0609             R3 := SAVER3 AND MASK5 SHRL 20; F(R3) := R0; R4 := R4-4;
09 0128  07 078C  0610          END; R3 := SAVER3; R0 := SAVER0;
09 0130  07 078C  0611          R10 := SAVERA;  R6 := SAVER6;
09 0138  07 078C  0612       END;
09 013A  07 078C  0613
09 013A  07 078C  0614    PROCEDURE ZRELEASE(R1); COMMENT SAME AS RELEASE EXCEPT INPUT IN R0.
09 013A  07 078C  0615       R0 AND R3 MUST BE SAVED;
09 013A  07 078C  0616       BEGIN LOGICAL SAVER3;
09 013A  07 0790  0617          LOGICAL SAVER1; SAVER1 := R1;
09 013E  07 0794  0618          SAVER3 := R3; R3 := R0; RELEASE;
09 0148  07 0794  0619          R0 := R3; R3 := SAVER3;
09 014E  07 0794  0620          R1 := SAVER1;
09 0152  07 0794  0621       END;
09 0154  07 0794  0622
09 0154  07 0794  0623    PROCEDURE DUMPGENREG(R1); COMMENT WRITES INSTRUCTION TO DUMP OLDEST
09 0154  07 0794  0624       GENERAL REGISTER VALUE INTO LOCAL STORE, ADJUSTS LSTACK AND RSTACK,
09 0154  07 0794  0625       AND PUTS AVAILABLE REGISTER NUMBER IN BITS 8 TO 11 OF R3.  IF THE
09 0154  07 0794  0626       LSTACK ENTRY IS A PAIR, ONLY THE ODD REGISTER IS SAVED AND THE EVEN
09 0154  07 0794  0627       REGISTER IS FREED;
09 0154  07 0794  0628       BEGIN LOGICAL SAVER1,TEMP;   SAVER1 := R1;
09 0158  07 079C  0629          R0 := RSTACK(0); TEMP := R0; R2 := R2-8;
09 0164  07 079C  0630          FOR R3 := 0 STEP 4 UNTIL R2 DO
09 0168  07 079C  0631          BEGIN R10 := R3+4; R0 := RSTACK(R10); RSTACK(R3) := R0; END;
09 0184  07 079C  0632          R3 := TEMP;  R0 := LSTACK(R3);  R10 := NEXTADDR +3 AND SMASK;
09 0198  07 079C  0633          NEXTADDR := R10;
09 019C  07 079C  0634          LSTACK(R3) := R10;
09 01A0  07 079C  0635          IF R0 < 0 THEN
09 01A6  07 079C  0636          BEGIN
09 01A6  07 079C  0637          R3 := R0; R0 := R0 SHLL 12;
09 01AC  07 079C  0638          IF R0 ¬= 0 THEN
```

```
09 01B2  07 079C  0639          BEGIN R3 := R3 AND MASK5 SHRL 19;
09 01BA  07 079C  0640             TEMP := R0; R0 := 0; R(R3) := R0;
09 01C6  07 079C  0641             R3 := TEMP SHRL 8;
09 01CE  07 079C  0642          END;
09 01CE  07 079C  0643          R3 := R3 SHLL 1 SHRL 1;
09 01D6  07 079C  0644          END ELSE
09 01D6  07 079C  0645          BEGIN
09 01DA  07 079C  0646             R3 := R0 AND BASEMASK SHLL 8;
09 01E4  07 079C  0647             R10 := R0 AND MASK8;
09 01EA  07 079C  0648             IF R10 ¬= 0 THEN BEGIN R0 := R0 OR R3 OR #41000000; EMIT; END;
09 0202  07 079C  0649          END;
09 0202  07 079C  0650          R0 := R3 OR NEXTADDR OR #50000000; EMIT;
09 0218  07 079C  0651          R10 := 4; INCRADDR;
09 0228  07 079C  0652          R2 := R2+4;
09 022C  07 079C  0653          R1 := SAVER1;
09 0230  07 079C  0654       END;
09 0232  07 079C  0655
09 0232  07 079C  0656    PROCEDURE FORCECOMPLEX(R1); COMMENT FORCES REAL PART INTO F0 AND IMAG
09 0232  07 079C  0657       PART INTO F2.  INPUT IS R0 = #38000000 OR R0 = #28000000
09 0232  07 079C  0658       DEPENDING ON SIMPLETYPE SHORT OR LONG COMPLEX;
09 0232  07 079C  0659       BEGIN INTEGER SAVER1,SAVER3,SAVERA,PREFX;
09 0232  07 07AC  0660          SAVER1 := R1;  SAVER3 := R3;  SAVERA := R10;  PREFX := R0;
09 0242  07 07AC  0661          R3 := LSTACKM4(R5) AND MASK5 SHRL 4;
09 024E  07 07AC  0662          R10 := LSTACKM4(R5) AND MASK4;        R1 := F(0);
09 025A  07 07AC  0663         IF R1 = 0 AND R10 ¬= 0 THEN
09 0266  07 07AC  0664          BEGIN COMMENT F0 NOT OCCUPIED;
09 0266  07 07AC  0665             R0 := PREFX OR R3;  IF R0 ¬= PREFX THEN EMIT;
09 0280  07 07AC  0666          END ELSE
09 0280  07 07AC  0667          BEGIN COMMENT F0 OCCUPIED;
09 0284  07 07AC  0668             IF R3 ¬= 0 THEN
09 028A  07 07AC  0669             BEGIN COMMENT REAL PART NOT IN F0;
09 028A  07 07AC  0670                R1 := F(2);
09 028E  07 07AC  0671                IF R1 = 0 AND R3 ¬= #00020000 THEN
09 029C  07 07AC  0672                BEGIN COMMENT F2 NOT OCCUPIED.  IMAG PART IN F0 -
09 029C  07 07AC  0673                   MOVE INTO F2.  MOVE REAL PART INTO F0;
09 029C  07 07AC  0674                   R0 := PREFX OR #00200000;      EMIT;
09 02B0  07 07AC  0675                   R0 := PREFX OR R3;       EMIT;
09 02C2  07 07AC  0676                END ELSE
09 02C2  07 07AC  0677                BEGIN COMMENT F2 OCCUPIED.  SWAP F0 AND F2;
09 02C6  07 07AC  0678                   R0 := #00600000 OR PREFX;      EMIT;
09 02DA  07 07AC  0679                   R0 := #00020000 OR PREFX;       EMIT;
09 02EE  07 07AC  0680                   R0 := #00260000 OR PREFX;      EMIT;
09 0302  07 07AC  0681                END;
09 03C2  07 07AC  0682                R10 := #20000;
09 03C6  07 07AC  0683             END;
09 03C6  07 07AC  0684          END;
09 03C6  07 07AC  0685          R1 := F(2);        IF R1 = 0 AND R10 ¬= #20000 THEN
09 0318  07 07AC  0686          BEGIN COMMENT MOVE IMAG PART INTO F2;
09 0318  07 07AC  0687             R0 := PREFX OR #00200000 OR R10;    EMIT;
09 032E  07 07AC  0688          END;
09 032E  07 07AC  0689          R1 := SAVER1;  R3 := SAVER3;  R10 := SAVERA;
09 033A  07 07AC  0690       END;
09 033C  07 07AC  0691
09 033C  07 07AC  0692    PROCEDURE GENREG(R1);
09 033C  07 07AC  0693       COMMENT  MAKES AVAILABLE A GENERAL REGISTER.
09 033C  07 07AC  0694         NUMBER TO R0(8:11), LSTACK, RSTACK UPDATED. CHANGES R1, R3 ALSO;
09 033C  07 07AC  0695       BEGIN  LOGICAL SAVER1, SAVERA;  SAVER1 := R1; SAVERA := R10;
09 0344  07 07B4  0696          R10 := CLN - 1 SHLA 1;  R0 := FLAG;
```

```
09 0354   07 07B4   0697          FOR R3 := 4 STEP 2 UNTIL R10 DO
09 0358   07 07B4   0698          IF R0 ¬= R(R3) THEN
09 0364   07 07B4   0699          BEGIN  COMMENT REGISTER FREE UNLESS R3 AND PROTECTED BY R3FLAG;
09 0364   07 07B4   0700             IF ¬R3FLAG OR R3 ¬= 6 THEN
09 0374   07 07B4   0701             BEGIN  R(R3) := R0;  R3 := R3 SHLA 19;  GOTO X;
09 038C   07 07B4   0702             END;
09 038C   07 07B4   0703          END;
09 038A   07 07B4   0704          DUMPGENREG;  COMMENT NO REGISTER FREE, DUMP AND ALLOCATE OLDEST;
09 038E   07 07B4   0705    X:    R0 := R3;  RESET(R3FLAG);  RSTACK(R2) := R5;  R5 := R5 + 4;
09 039C   07 07B4   0706          R3 := R3 OR SIGN; LSTACKM4(R5) := R3;  R2 := R2 + 4;
09 03A8   07 07B4   0707          R10 := SAVERA;  R1 := SAVER1;
09 03BC   07 07B4   0708       END;
09 03B2   07 07B4   0709
09 03B2   07 07B4   0710    PROCEDURE FIXUP(R1);
09 03B2   07 07B4   0711       COMMENT R3 HAS BRANCH ADDRESS : LOGPOINTER POINTS TO FIRST FIXUP;
09 03B2   07 07B4   0712       BEGIN LOGICAL SAVER1,SAVER2;
09 03B2   07 07BC   0713          SAVER1 := R1;  SAVER2 := R2;
09 03BA   07 07BC   0714          R2 := LOGPOINTER;  R2 := R2 - 4;  LOGPOINTER := R2;
09 03C6   07 07BC   0715          R0 := LOGSTACK(R2);  R3 := R3 OR #E000;
09 03CE   07 07BC   0716          WHILE R0 ¬= 0 DO
09 03D4   07 07BC   0717          BEGIN R2 := R0;  R0 := PROGRAM(R2+2);  R0 := R0 AND #1FFF;
09 03DE   07 07BC   0718             PROGRAM(R2+2) := R3;
09 03E2   07 07BC   0719          END;
09 03E6   07 07BC   0720          R3 := LOGPOINTER;  R0 := LOGSTACK(R3+4);  LOGSTACK(R3) := R0;
09 03F2   07 07BC   0721          R1 := SAVER1;  R2 := SAVER2;
09 03FA   07 07BC   0722       END;
09 03FC   07 07BC   0723    PROCEDURE LOGBRANCH(R1);
09 03FC   07 07BC   0724       BEGIN LOGICAL SAVER1;  SAVER1 := R1;
09 0400   07 07C0   0725          R0 := R0 SHLL 20 OR #4700E000;  EMIT;
09 0414   07 07C0   0726          R1 := SAVER1;
09 0418   07 07C0   0727       END;
09 041A   07 07C0   0728    PROCEDURE LINKBRANCHES(R1);
09 041A   07 07C0   0729       BEGIN LOGICAL SAVER1;  SAVER1 := R1;
09 041E   07 07C4   0730          R0 := INSCOUNTER - 4;  R10 := LOGPOINTER;
09 042A   07 07C4   0731          R1 := LOGSTACK2(R10);     LOGSTACK2(R10) := R0;
09 0432   07 07C4   0732          R10 := R0;  PROGRAM(R10+2) := R1;
09 0438   07 07C4   0733          R1 := SAVER1;
09 043C   07 07C4   0734       END;
09 043E   07 07C4   0735    PROCEDURE EVALUATELOG(R1);
09 043E   07 07C4   0736       BEGIN LOGICAL SAVER1;  SAVER1 := R1;
09 0442   07 07C8   0737          R3 := LOGPOINTER;  R0 := 0;  IC(R0,LOGSTACK(R3+1));
09 044E   07 07C8   0738          R3 := R3 - 4;  LOGPOINTER := R3;  LOGBRANCH;  LINKBRANCHES;
09 045E   07 07C8   0739          GENREG;  R3 := LOGPOINTER;  IC(R3,LOGSTACK(R3+4));
09 046A   07 07C8   0740          R3 := R3 AND #FF;  CASE R3 OF
09 046E   07 07C8   0741          BEGIN
09 046E   07 07C8   0742             BEGIN R0 := R0 OR #41000000;  EMIT;  R3 := INSCOUNTER - 4;
09 048E   07 07C8   0743                FIXUP;  R3 := INSCOUNTER + 8;  R0 := #47F0E000 OR R3;
09 04A0   07 07C8   0744                EMIT;  R3 := LOGPOINTER + 4;  LOGPOINTER := R3;
09 04B8   07 07C8   0745                R3 := INSCOUNTER;  FIXUP;
09 04C0   07 07C8   0746                R0 := LSTACKM4(R5) XOR SIGN OR #41000001;  EMIT;
09 04D8   07 07C8   0747             END;
09 04D8   07 07C8   0748             BEGIN R0 := R0 OR #41000001;  EMIT;  R3 := INSCOUNTER - 4;
09 04F4   07 07C8   0749                FIXUP;  R3 := INSCOUNTER + 8;  R0 := #47F0E000 OR R3;
09 05C6   07 07C8   0750                EMIT;  R3 := LOGPOINTER + 4;  LOGPOINTER := R3;
09 051E   07 07C8   0751                R3 := INSCOUNTER;  FIXUP;
09 0526   07 07C8   0752                R0 := LSTACKM4(R5) XOR SIGN OR #41000000;  EMIT;
09 053E   07 07C8   0753             END;
09 053E   07 07C8   0754          END;
```

```
09 054A  07 07C8  0755          R3 := LOGPOINTER - 4;   LOGPOINTER := R3;
09 0556  07 07C8  0756          R1 := SAVER1;
09 055A  07 07C8  0757       END;
09 055C  07 07C8  0758
09 055C  07 07C8  0759    PROCEDURE STORERESULT(R1);  COMMENT THIS PROCEDURE STORES THE
09 055C  07 07C8  0760       REGISTER(S) GIVEN BY THE TOP ENTRY OF LSTACK INTO THE LOCAL STACK
09 055C  07 07C8  0761       STARTING AT NEXTADDR, AND EMITS  LA 3,NEXTADDR  .  NEXTADDR IS RESET
09 055C  07 07C8  0762       TO A WORD BOUNDARY WHEN DONE.  SIMPLETYPE IN LOW BITS OF R3 -
09 055C  07 07C8  0763       BITS AND REFERENCE = INTEGER        ;
09 055C  07 07C8  0764       BEGIN INTEGER SS1,SS6,SS3;  SS1 := R1;  SS6 := R6;  SS3 := R3;
09 0568  07 07D4  0765          IF R3 ¬= 6 THEN
09 0570  07 07D4  0766          BEGIN
09 0570  07 07D4  0767             R10 := NEXTADDR + 3 SHRL 2 SHLL 2;  NEXTADDR := R10;
09 0584  07 07D4  0768          END;
09 0584  07 07D4  0769          R10 := LSTACKM4(R5) AND MASK5;
09 058C  07 07D4  0770          CASE R3 OF
09 058C  07 07D4  0771          BEGIN
09 058C  07 07D4  0772          BEGIN
09 0594  07 07D4  0773             R1:=NEXTADDR; R0:=R10 OR #50000000 OR R1; R10:=R1+4;
09 05A6  07 07D4  0774          END;
09 05A6  07 07D4  0775          BEGIN
09 05AA  07 07D4  0776             R1 := NEXTADDR;  R0 := R10 OR #70000000 OR R1;  R10 := R1 + 4;
09 05B8  07 07D4  0777          END;
09 05BC  07 07D4  0778          BEGIN
09 05C0  07 07D4  0779             R1 := NEXTADDR + 7 SHRL 3 SHLL 3;  NEXTADDR := R1;
09 05D4  07 07D4  0780             R0 := R10 OR #60000000 OR R1;  R10 := R1 + 8;
09 05E2  07 07D4  0781          END;
09 05E2  07 07D4  0782          BEGIN
09 05E6  07 07D4  0783             R0:=R10 OR #70000000 OR NEXTADDR; EMIT; R1:=NEXTADDR;
09 0600  07 07D4  0784             R0 := LSTACKM4(R5) AND MASK4 SHLL 4 OR #70000000 OR R1 + 4;
09 0616  07 07D4  0785             R10 := R1 + 8;
09 061C  07 07D4  0786          END;
09 061C  07 07D4  0787          BEGIN
09 0620  07 07D4  0788             R1 := NEXTADDR + 7 SHRL 3 SHLL 3;  NEXTADDR := R1;
09 0634  07 07D4  0789             R0 := R10 OR #60000000 OR R1;    EMIT;  R1 := NEXTADDR;
09 064C  07 07D4  0790             R0 := LSTACKM4(R5) AND MASK4 SHLL 4 OR #60000000 OR R1 + 8;
09 0662  07 07D4  0791             R10 := R1 + 8;
09 0668  07 07D4  0792          END;
09 0668  07 07D4  0793          BEGIN COMMENT LOGICAL;
09 066C  07 07D4  0794             TEST(TERMNODE);  IF ¬= THEN
09 0674  07 07D4  0795             BEGIN COMMENT NONTERMINAL;
09 0674  07 07D4  0796                R10 := R7 SHLL 8 SHRL 24;
09 067E  07 07D4  0797                IF R10 = 0 THEN EVALUATELOG;
09 0688  07 07D4  0798                R3 := LSTACKM4(R5);  R6 := 6;  RELEASE;  R5 := R5 - 4;
09 0698  07 07D4  0799                IF R3 >= 0 THEN
09 069E  07 07D4  0800                BEGIN R0 := #43300000 OR R3;  EMIT;  END ELSE
09 06B0  07 07D4  0801                BEGIN R0 := R3 SHLL 1 SHRL 5 OR #18300000;
09 06C2  07 07D4  0802                   IF R0 ¬= #18330000 THEN EMIT;
09 06D6  07 07D4  0803                END;
09 06D6  07 07D4  0804             END;  COMMENT VALUE IN R3;
09 06D6  07 07D4  0805             R1 := NEXTADDR;  R0 := #42300000 OR R1;  R10 := R1 + 4;
09 06E6  07 07D4  0806          END;
09 06E6  07 07D4  0807          END;
09 0702  07 07D4  0808          NEXTADDR := R10;    R10 := #41300000 OR R1;
09 070C  07 07D4  0809          EMIT;        R0 := R10;      EMIT;
09 0726  07 07D4  0810          R1 := SS1;  R6 := SS6;  R3 := SS3;
09 0732  07 07D4  0811       END;
09 0734  07 07D4  0812
```

```
09 0734   07 07D4   0813    PROCEDURE SETRECORD(R1);
09 0734   07 07D4   0814       BEGIN LOGICAL SR3,SRA;    SR3 := R3;    SRA := R10;
09 073C   07 07DC   0815          R3 := PUMREC;    R10 := R5 - 4;  UMREC(R3) := R10;  R3 := R3 + 2;
09 074A   07 07DC   0816          PUMREC := R3;
09 0752   07 07DC   0817          COMMENT TEST FOR ERROR;
09 0752   07 07DC   0818          R3 := SR3;  R10 := SRA;
09 075A   07 07DC   0819       END;
09 075C   07 07DC   0820    PROCEDURE RESETRECORD(R1);
09 075C   07 07DC   0821       BEGIN  LOGICAL SR1,SR3,SR0,SRA;
09 075C   07 07EC   0822          SR1 := R1;  SR3 := R3;  SR0 := R0;  SRA := R10;
09 076C   07 07EC   0823          FOR R10 := PUMREC - 2  STEP _2 UNTIL 0 DO
09 0774   07 07EC   0824          IF R5 = UMREC(R10) THEN
09 0780   07 07EC   0825          BEGIN R0 := PUMREC -2 ;  PUMREC := R0;
09 078C   07 07EC   0826             FOR R3 := R10 STEP 2  UNTIL PUMREC DO
09 078E   07 07EC   0827             BEGIN R0 := UMREC(R3+2);  UMREC(R3) := R0;
09 079A   07 07EC   0828             END;
09 07A6   07 07EC   0829             GOTO RS1;
09 07AA   07 07EC   0830          END;
09 07B6   07 07EC   0831          FOR R10 := PMREC - 2  STEP _2 UNTIL 0 DO
09 07BE   07 07EC   0832          IF R5  = MREC(R10) THEN
09 07CA   07 07EC   0833          BEGIN R0 := LSTACK(R5);
09 07CE   07 07EC   0834             IF  R0 >= 0 THEN
09 07D4   07 07EC   0835             BEGIN R0 := R0 OR #58100000;  EMIT;
09 07E4   07 07EC   0836                R0 := NII OR #3F1000;  EMIT;
09 07F8   07 07EC   0837             END ELSE
09 07F8   07 07EC   0838             BEGIN  R0 := R0 XOR SIGN SHRL 8 OR NII  OR #3F0000;  EMIT;
09 0818   07 07EC   0839             END;
09 0818   07 07EC   0840             R0 := PMREC - 2;  PMREC := R0;
09 0824   07 07EC   0841             FOR R3 := R10 STEP 2 UNTIL PMREC DO
09 0826   07 07EC   0842             BEGIN  R0 := MREC(R3+2);  MREC(R3) := R0;
09 0832   07 07EC   0843             END;
09 083E   07 07EC   0844             GOTO RS1;
09 0842   07 07EC   0845          END;
09 084E   07 07EC   0846    RS1:  R1 := SR1;  R3 :=SR3;  R0 := SR0;  R10 := SRA;
09 085E   07 07EC   0847       END;
09 0860   07 07EC   0848     PROCEDURE CHECKMARK(R1);
09 0860   07 07EC   0849       BEGIN  LOGICAL SR1;  SR1 := R1;
09 0864   07 07F0   0850          R5 := R5 - 4;
09 0868   07 07F0   0851          FOR R3 := PMREC - 2 STEP _2 UNTIL 0 DO
09 0870   07 07F0   0852          IF R5 = MREC(R3) THEN
09 087C   07 07F0   0853          BEGIN  R0 := LSTACK(R5+4);
09 0880   07 07F0   0854             IF R0 < 0 THEN R0 := R0 XOR SIGN SHRL 8;
09 088E   07 07F0   0855             R0 := R0 OR OII OR #400000;  EMIT;  GOTO CM1;
09 08A6   07 07F0   0856          END;
09 08B2   07 07F0   0857    CM1:  R1 := SR1;  R5 := R5 + 4;
09 08BA   07 07F0   0858       END;
09 08BC   07 07F0   0859     PROCEDURE MARKRECORDS(R1);
09 08BC   07 07F0   0860       BEGIN  LOGICAL SR1,SR2;   SR1 := R1;  SR2 := R2;
09 08C4   07 07F8   0861          R10 := PUMREC - 2;  R2 := PMREC;
09 08D0   07 07F8   0862          FOR R3 := 0  STEP 2  UNTIL R10 DO
09 08D4   07 07F8   0863          BEGIN
09 08D8   07 07F8   0864             R1 := UMREC(R3);  R0 := LSTACK(R1);
09 08E0   07 07F8   0865             IF R0 >= 0 THEN
09 08E6   07 07F8   0866             BEGIN  R0 := R0 OR #58100000;  EMIT;
09 08F6   07 07F8   0867                R0 := OII OR #401000;    EMIT;
09 090A   07 07F8   0868             END ELSE
09 090A   07 07F8   0869             BEGIN R0 := R0 XOR SIGN SHRL 8 OR OII OR #400000;  EMIT;
09 092A   07 07F8   0870             END;
```

```
09 092A  07 07F8  0871            R1 := UMREC(R3);
09 092E  07 07F8  0872            MREC(R2) := R1;  R2 := R2 + 2;
09 0936  07 07F8  0873         END;
09 0940  07 07F8  0874         PMREC := R2;   R0 := R0 - R0;  PUMREC := R0;
09 094A  07 07F8  0875         R1 := SR1;  R2 := SR2;
09 0952  07 07F8  0876      END;
09 0954  07 07F8  0877
09 0954  07 07F8  0878   SEGMENT PROCEDURE  LEVEL0(R1);
09 0954  07 07F8  0879    BEGIN LOGICAL SAVER1;
10 0000  07 07FC  0880
10 0000  07 07FC  0881    PROCEDURE DUMPALLGENREG(R1);
10 0000  07 07FC  0882       BEGIN LOGICAL SAVER1, SAVER0, SAVER3;
10 0004  07 0808  0883          SAVER1 := R1; SAVER0 := R0; SAVER3 := R3;
10 0010  07 0808  0884          WHILE R2 > 0 DO
10 0016  07 0808  0885          BEGIN DUMPGENREG; R3 := R3 SHRL 19; R0 := 0; R(R3) := R0;
10 002E  07 0808  0886          END;
10 0032  07 0808  0887          R1 := SAVER1; R0 := SAVER0; R3 := SAVER3;
10 003E  07 0808  0888       END;
10 0040  07 0808  0889
10 0040  07 0808  0890    PROCEDURE PRGENREG(R1); COMMENT MAKES AVAILABEL AN EVEN-ODD PAIR OF
10 0040  07 0808  0891       GENERAL REGISTERS, LEAVING THE EVEN OR ODD NUMBER, ACCORDING AS THE
10 0040  07 0808  0892       DIVIDE FLAG IS ON OR OFF, IN BITS 8 TO 11 OF R0.  RSTACK AND LSTACK
10 0040  07 0808  0893       ARE UPDATED.  THE LSTACK ENTRY CONTAINS THE EVEN NUMBER IN BITS
10 0040  07 0808  0894       8 TO 11 AND THE ODD NUMBER IN BITS 12 TO 15.  BOTH REGISTERS ARE
10 004C  07 0808  0895       FLAGGED IN R;
10 0040  07 0808  0896       BEGIN LOGICAL SAVER1, TOP;
10 0040  07 0810  0897          SAVER1 := R1;  R1 := CLN - 1 SHLA 1 - 2;  TOP := R1;
10 0058  07 0810  0898       A: R0 := 4;    TEST(R3FLAG);    IF = THEN
10 0064  07 0810  0899             BEGIN RESET(R3FLAG);    R0 := 8;    END;
10 006C  07 0810  0900          FOR R3 := R0 STEP 4 UNTIL TOP DO
10 006E  07 0810  0901          BEGIN R0 := R(R3); IF R0 = 0 THEN
10 007C  07 0810  0902             BEGIN R10 := R3+2; R0 := R(R10); IF R0 = 0 THEN
10 008C  07 0810  0903                BEGIN R0 := FLAG; R(R3) := R0; R(R10) := R0; GOTO B; END;
10 009C  07 0810  0904             END;
10 00AC  07 0810  0905          END;
10 00A8  07 0810  0906          COMMENT NO FREE PAIR SO DUMP OLDEST VALUE AND TRY AGAIN;
10 00A8  07 0810  0907          DUMPGENREG; R3 := R3 SHRL 19; R0 := 0; R(R3) := R0; GOTO A;
10 00C4  07 0810  0908       B: TEST(DIVIDE); IF = THEN R0 := R3 SHLL 19 ELSE R0 := R10 SHLL 19;
10 00DC  07 0810  0909          R3 := R3 SHLL 4 OR R10 SHLL 15 OR SIGN;
10 00EA  07 0810  0910          LSTACK(R5) := R3; RSTACK(R2) := R5;
10 00F2  07 0810  0911          R2 := R2+4; R5 := R5+4;
10 00FA  07 0810  0912          R1 := SAVER1;
10 00FE  07 0810  0913       END;
10 0100  07 0810  0914
10 0100  07 0810  0915    PROCEDURE FORCEPAIR(R1); COMMENT A VALUE IN A GENERAL REGISTER IS
10 0100  07 0810  0916       FORCED INTO THE ODD REGISTER OF AN EVEN-ODD PAIR.  R0 IS AN INPUT
10 0100  07 0810  0917       SHOWING THE REGISTER(S) HOLDING THE VALUE.  IF THIS IS A PAIR NO
10 0100  07 0810  0918       CODE IS GENERATED.  R0 IS AN OUTPUT WITH EVEN REGISTER IN BITS 8 TO
10 0100  07 0810  0919       11.  THE TOP LSTACK ENTRY IS UPDATED TO SHOW THE PAIR UNLESS
10 0100  07 0810  0920       THE DIVIDE FLAG IS ON, IN WHICH CASE IT SHOWS THE EVEN OR ODD
10 0100  07 0810  0921       REGISTER ACCORDING AS THE REMAINDER FLAG IS ON OR OFF.  JUST THE
10 0100  07 0810  0922       REGISTER(S) IN THIS ENTRY ARE FLAGGED AS OCCUPIED BY THE RESULT;
10 0100  07 0810  0923       BEGIN LOGICAL SAVER0, SAVER1, SAVER3;
10 0100  07 081C  0924       PROCEDURE NEWPAIR(R1); COMMENT FINDS NEW EVEN-ODD PAIR AND LOADS
10 0100  07 081C  0925          ODD REGISTER;
10 0100  07 081C  0926          BEGIN LOGICAL SAVER1;  SAVER1 := R1;
10 0108  07 0820  0927             R2 := R2-4; R5 := R5-4; PRGENREG;
10 0114  07 0820  0928             R3 := SAVER0 SHRL 19;
```

```
10 011C  07 0820  0929            R10 := 0; R(R3) := R10; R3 := SAVER0 SHRL 4;
10 012C  07 0820  0930            TEST(DIVIDE); IF ¬= THEN
10 0134  07 0820  0931            BEGIN R0 := R0 OR R3 OR #18000000; EMIT; COMMENT LOAD NEW ODD
10 0146  07 0820  0932               REGISTER;
10 0146  07 0820  0933               R0 := LSTACKM4(R5) AND MASK5;
10 014E  07 0820  0934            END ELSE
10 014E  07 0820  0935            BEGIN LOGICAL SAVER0; SAVER0 := R0;
10 0156  07 0824  0936               R0 := R0 OR R3 OR #18000000; EMIT; COMMENT LOAD NEW EVEN
10 0168  07 0824  0937               REGISTER AND SHIFT;
10 0168  07 0824  0938               R0 := SAVER0 OR #8E000020; EMIT;
10 017C  07 0824  0939               R0 := SAVER0;
10 0180  07 0824  0940               R1 := R1 - R1;  IF REMAINDER THEN
10 018A  07 0824  0941               BEGIN COMMENT MARK ODD FREE;
10 018A  07 0824  0942                  R3 := LSTACKM4(R5) AND MASK4 SHRL 15;  R(R3) := R1;
10 019A  07 0824  0943                  R3 := LSTACKM4(R5) AND MASK5 OR SIGN;
10 01A6  07 0824  0944               END ELSE
10 01A6  07 0824  0945               BEGIN COMMENT MARK EVEN FREE;
10 01AA  07 0824  0946                  R3 := LSTACKM4(R5) AND MASK5 SHRL 19;  R(R3) := R1;
10 01BA  07 0824  0947                  R3 := LSTACKM4(R5) AND MASK4 SHLL 4 OR SIGN;
10 01CA  07 0824  0948               END;
10 01CA  07 0824  0949               LSTACKM4(R5) := R3;
10 01CE  07 0824  0950            END;
10 01CE  07 0824  0951         R1 := SAVER1;
10 01D2  07 0824  0952         END;
10 01D4  07 0824  0953         SAVER3 := R3;
10 01D8  07 0824  0954         SAVER1 := R1; R0 := R0 SHLL 1 SHRL 1; SAVER0 := R0;
10 01E8  07 0824  0955         R3 := R0 SHLL 12; IF R3 = 0 THEN
10 01F4  07 0824  0956         BEGIN R0 := R0 SHRL 20; R3 := R0 SHRL 1 SHLL 1;
10 02C2  07 0824  0957            IF R0 = R3 THEN
10 02C8  07 0824  0958            BEGIN R3 := R3+1 SHLL 1; R10 := R(R3); IF R10 = 0 THEN
10 021A  07 0824  0959               BEGIN
10 021A  07 0824  0960                  TEST(REMAINDER); IF ¬= THEN
10 0222  07 0824  0961                  BEGIN R10 := FLAG; R(R3) := R10; END;
10 022A  07 0824  0962                  R0 := SAVER0 OR #8E000020; EMIT; COMMENT SHIFT TO
10 023E  07 0824  0963                  ODD REGISTER;
10 023E  07 0824  0964                  R0 := SAVER0;
10 0242  07 0824  0965                  TEST(DIVIDE); IF ¬= THEN
10 024A  07 0824  0966                  BEGIN R3 := R3 SHLL 15; R10 := R0 OR R3 OR SIGN;
10 0256  07 0824  0967                     LSTACKM4(R5) := R10;
10 025A  07 0824  0968                  END ELSE
10 025A  07 0824  0969                  BEGIN TEST(REMAINDER); IF ¬= THEN
10 0266  07 0824  0970                     BEGIN R10 := R0 SHRL 19; R0 := 0; R(R10) := R0;
10 0274  07 0824  0971                        R3 := R3 SHLL 19; R10 := R3 OR SIGN;
10 027E  07 0824  0972                        LSTACKM4(R5) := R10;
10 0282  07 0824  0973                        R0 := SAVER0;
10 0286  07 0824  0974                     END;
10 0286  07 0824  0975                  END;
10 0286  07 0824  0976               END ELSE NEWPAIR;
10 028E  07 0824  0977            END ELSE
10 028E  07 0824  0978            BEGIN R3 := R3 SHLL 1; R10 := R(R3); IF R10 = 0 THEN
10 02A0  07 0824  0979               BEGIN TEST(DIVIDE); IF = THEN
10 02A8  07 0824  0980                  BEGIN R3 := R3 SHLL 19;
10 02AC  07 0824  0981                     R0 := SAVER0 SHRL 4 OR R3 OR #18000000; EMIT;
10 02C6  07 0824  0982                     COMMENT LOAD FROM ODD TO EVEN REGISTER AND SHIFT
10 02C6  07 0824  0983                     TO EXTEND SIGN;
10 02C6  07 0824  0984                     R0 := R3 OR #8A00001F; EMIT;
10 02D8  07 0824  0985                     R0 := R3;
10 02DA  07 0824  0986                      TEST(REMAINDER); IF = THEN
```

```
10 02E2  07 0824   0987                    BEGIN  R3 := R3 OR SIGN;  LSTACKM4(R5) := R3;
10 02EA  07 0824   0988                       R3 := R3 SHLL 1 SHRL 20;  R10 := FLAG;
10 02F6  07 0824   0989                       R(R3) := R10;  R10 := R10 - R10;  R(R3+2) := R10;
10 0300  07 0824   0990                    END;
10 0300  07 0824   0991                 END ELSE
10 0300  07 0824   0992                 BEGIN R10 := FLAG; R(R3) := R10;
10 030C  07 0824   0993                    R3 := R3 SHLL 19; R0 := R0 SHLL 16 OR R3 OR SIGN;
10 031A  07 0824   0994                    LSTACKM4(R5) := R0; R0 := R3;
10 0320  07 0824   0995                 END;
10 0320  07 0824   0996              END ELSE NEWPAIR;
10 0328  07 0824   0997           END;
10 0328  07 0824   0998        END ELSE
10 0328  07 0824   0999        BEGIN TEST(DIVIDE); IF = THEN
10 0334  07 0824   1000           BEGIN TEST(REMAINDER); IF = THEN
10 033C  07 0824   1001              BEGIN R3 := R0 AND MASK4 SHRL 15;
10 0346  07 0824   1002                 R10 := 0; R(R3) := R10;
10 034E  07 0824   1003                 R10 := R0 AND MASK5 OR SIGN;
10 0358  07 0824   1004              END ELSE
10 0358  07 0824   1005              BEGIN R3 := R0 AND MASK5 SHRL 19;
10 0366  07 0824   1006                 R10 := 0; R(R3) := R10;
10 036E  07 0824   1007                 R10 := R0 AND MASK4 SHLL 4 OR SIGN;
10 037C  07 0824   1008              END;
10 037C  07 0824   1009              LSTACKM4(R5) := R10;
10 0380  07 0824   1010           END;
10 0380  07 0824   1011           R0 := R0 AND MASK5;
10 0384  07 0824   1012        END;
10 0384  07 0824   1013        R1 := SAVER1;    R3 := SAVER3;
10 038C  07 0824   1014        R0 := R0 OR SIGN;
10 0390  07 0824   1015     END;
10 0392  07 0824   1016  PROCEDURE ADJSTACKS(R1); COMMENT DEPENDING ON TYPE (IN R6), DECREMENTS
10 0392  07 0824   1017     BY 4 THE TOP OF RSTACK OR FSTACK TO POINT TO TOP OF LSTACK AFTER
10 0392  07 0824   1018     BINARY OPERATION;
10 0392  07 0824   1019     BEGIN LOGICAL SAVERA; SAVERA := R10;
10 0396  07 0828   1020        IF R6 = 1 OR R6 >= 8 THEN
10 03A6  07 0828   1021        BEGIN R10 := RSTACKM4(R2)-4; RSTACKM4(R2) := R10; END ELSE
10 03B2  07 0828   1022        BEGIN R10 := FSTACKM4(R4)-4; FSTACKM4(R4) := R10; END;
10 03C2  07 0828   1023        R10 := SAVERA;
10 03C6  07 0828   1024     END;
10 03C8  07 0828   1025
10 03C8  07 0828   1026  PROCEDURE FPARCODE(R1);  COMMENT THIS PROCEDURE EMITS THE CODE FOR
10 03C8  07 0828   1027     FORMAL NAME PARAMETERS IN EXPRESSIONS AND ASSIGNMENTS.  ALL
10 03C8  07 0828   1028     SUBROUTINES CALLED BY THE CODE EMITTED HERE RETURN A COMPUTED
10 03C8  07 0828   1029     ADDRESS IN R3(AT RUN TIME).  AT COMPILE TIME, IF R0 = #FF THIS
10 03C8  07 0828   1030     PROCEDURE EMITS CODE FOR FPAR ON RIGHT SIDE OF ASSIGNMENT STATEMENT.
10 03C8  07 0828   1031     IF R0 = 0, THE EMITTED CODE IS FOR FPAR ON LEFT OF ASSIGNMENT
10 03C8  07 0828   1032     STATEMENT.  R3 MUST BE POINTER TO DPD IN DATA SEG (WITH LEV NUMBER);
10 03C8  07 0828   1033     BEGIN INTEGER SAVER1,DPOINT;  BYTE SWW;
10 03C8  07 0831   1034        SAVER1 := R1;  DPOINT := R3;  STC(R0,SWW);
10 03D4  07 0831   1035        MARKRECORDS;    DUMPALLGENREG;    DUMPALLFLREG;
10 03F0  07 0831   1036        R0 := #58300000 OR DPOINT;  EMIT;
10 0404  07 0831   1037        R0 := #91000000 OR DPOINT; IF SWW THEN R0 := R0 OR #00020000 ELSE
10 0414  07 0831   1038        BEGIN   R0 := R0 OR #00030000; EMIT;
10 042C  07 0831   1039           R0 := #45100000 OR NAMEERR;
10 0434  07 0831   1040        END;
10 0434  07 0831   1041        EMIT;  R1 := LOGPOINTER + 4; LOGPOINTER := R1;  R0 := INSCOUNTER;
10 044C  07 0831   1042        LOGSTACK(R1) := R0;  R0 := #47E0E000; EMIT;
10 0464  07 0831   1043        R0 := #58500000 OR DPOINT + 4; EMIT;  R0 := #05130000; EMIT;
10 048C  07 0831   1044        LMINS; CHAININ;
```

```
10 04A4  07 0831  1045          R1 := LOGPOINTER;  R3 := LOGSTACK(R1) + 2;  R1 := R1 - 4;
10 04B4  07 0831  1046          LOGPOINTER := R1;  R0 := INSCOUNTER OR #E000;  PROGRAM(R3) := R0;
10 04C0  07 0831  1047          R1 := SAVER1;
10 04C8  07 0831  1048       END;
10 04CA  07 0831  1049
10 04CA  07 0831  1050    PROCEDURE CALLPROPROCWOPARAM(R1);
10 04CA  07 0831  1051       BEGIN INTEGER SAVER1;  SAVER1 := R1;
10 04CE  07 0838  1052          DUMPALLGENREG;  DUMPALLFLREG;  R3 := R3 - R3;
10 04E0  07 0838  1053          PROCCALLCODE;  R1 := SAVER1;
10 04F0  07 0838  1054       END;
10 04F2  07 0838  1055
10 04F2  07 0838  1056    PROCEDURE RECORDALLOCATE(R1);
10 04F2  07 0838  1057       BEGIN  LOGICAL ADDRESS,SAVER1;
10 04F2  07 0840  1058          SAVER1 := R1;  MARKRECORDS;
10 0502  07 0840  1059          R3 := 0;  IF R3 ¬= R(6) THEN DUMPALLGENREG;  R3 := #FFFFFFFF;
10 0516  07 0840  1060          IF R3 = R(4) THEN GENREG ELSE
10 052A  07 0840  1061          BEGIN  R(4) := R3;  GENREG;  R3 := 0;  R(4) := R3; END;
10 0546  07 0840  1062          R3 := 0;  IC(R3,TYPEINFO(R7+1));  R3 := R3 SHLL 4 ++ RECTABLE;
10 0556  07 0840  1063          R0 := #41300000 OR R3;  EMIT;
10 0568  07 0840  1064          R0 := @RECALLOCATE; R0 := NEG R0; EMITCALL;
10 057A  07 0840  1065          SETRECORD;  R1 := SAVER1;
10 058A  07 0840  1066       END;
10 058C  07 0840  1067
10 058C  07 0840  1068    PROCEDURE GETADDRESS(R1);  COMMENT GETS ADDRESS FROM NAMETABLE OR
10 058C  07 0840  1069       CONSTANT POINTER TABLE.  ENTRY INDEXED BY R7, RESULT IN R3
10 058C  07 0840  1070       HIERARCHY NUMBER INCLUDED FOR ARRAYS, SIMPLE VARIABLES
10 058C  07 0840  1071       FOR LITERALS HIERARCHY NUMBER IS E              ;
10 058C  07 0840  1072       BEGIN  ARRAY 2 LOGICAL SAVE01;  LOGICAL SAVER7;
10 058C  07 084C  1073
10 058C  07 084C  1074         PROCEDURE  PROCPARAM(R1);
10 058C  07 084C  1075          COMMENT SETS R3 AFTER NAME PARAM (R0=0) OR FUNCTION CALL AND
10 058C  07 084C  1076          ADJUSTS R7. FOR STRING PROCEDURES, MOVES RESULT TO LOCAL STACK;
10 058C  07 084C  1077          BEGIN  LOGICAL SAVER1;  SAVER1 := R1;
10 0594  07 0850  1078             R3 := R3-R3;  IC(R3,SIMPLETYPE(R7));
10 059A  07 0850  1079             IF R3 = 7 THEN
10 05A2  07 0850  1080             BEGIN  R1 := SIMTYPEINFO(R7);  IF R0 ¬= 0 THEN
10 05AC  07 0850  1081                BEGIN  COMMENT STRING PROCEDURE;
10 05AC  07 0850  1082                   R0 := R1 SHLL 16 OR NEXTADDR OR #D2000000;
10 05BA  07 0850  1083                   R3 := #30000000;  EMIT;
10 05CA  07 0850  1084                   R1 := SIMTYPEINFO(R7);  R3 := NEXTADDR;
10 05D2  07 0850  1085                   R0 := R3 + R1 + 1;  NEXTADDR := R0;
10 05DE  07 0850  1086                END ELSE
10 05DE  07 0850  1087                R3 := #3000;
10 05E6  07 0850  1088                R1 := R1 SHLL 16 OR #87000000;
10 05EE  07 0850  1089             END ELSE
10 05EE  07 0850  1090             IF R3 = 6 THEN
10 05FA  07 0850  1091             BEGIN  R1 := #86010000;  R3 := #3000;
10 06C2  07 0850  1092             END ELSE
10 06C2  07 0850  1093             BEGIN  IF R3 = 1 OR R3 >= 8 THEN SET(R3FLAG);
10 061A  07 0850  1094                R3 := R3 SHLL 24 OR SIGN;
10 0622  07 0850  1095                R1 := SAVER7 AND #00FFFFFF OR R3;  R3 := #3000;
10 0630  07 0850  1096             END;
10 0630  07 0850  1097             SAVER7 := R1;  R1 := SAVER1;
10 0638  07 0850  1098          END;
10 063A  07 0850  1099
10 063A  07 0850  1100          STM(R0,R1,SAVE01);  SAVER7 := R7;
10 0642  07 0850  1101          R7 := R7 AND #7F00FFFF;  COMMENT EXTRACT OPCODE, TABLE PTR;
10 0646  07 0850  1102          R0 := R7 SHRL 24;
```

```
10 064C  07 0850  1103          IF R0 = ID THEN
10 0654  07 0850  1104          BEGIN  R3 := IDLOC1(R7) SHLL 12 + IDLOC2(R7);
10 066C  07 0850  1105             R1 := 0;  IC(R1,TYPE(R7)); IF R1 = 16 THEN
10 0670  07 0850  1106             BEGIN COMMENT FORMAL NAME PARAMETER;
10 067C  07 0850  1107                R1 := STACK(R3) AND #FFFF;  R1 := TREE(R1);
10 067C  07 0850  1108                R0 := R1 SHRL 24 AND #7F;
10 0686  07 0850  1109                IF R0 >= 6 AND R0 <= 9 THEN R0 := 0 ELSE
10 069A  07 0850  1110                IF R0 >= 22 AND R0 <= 25 THEN R0 := 0 ELSE
10 06B2  07 0850  1111                R0 := #FF;
10 06BA  07 0850  1112                IF R0 = 0 THEN
10 06C0  07 0850  1113                BEGIN  COMMENT  := OR :=2, LEFT OR RIGHT;
10 06C0  07 0850  1114                   IF ARGFLAG THEN R1 := NEG R1;
10 06CA  07 0850  1115                   IF R1 < 0 THEN R0 := #FF;  COMMENT ON RIGHT;
10 06D4  07 0850  1116                END;
10 06D4  07 0850  1117                FPARCODE;   R0 := 0;  PROCPARAM;
10 06E0  07 0850  1118             END;
10 06EC  07 0850  1119          END ELSE
10 06E0  07 0850  1120          IF R0 = CONID OR R0 = ARRAYID THEN
10 06F4  07 0850  1121          R3 := IDLOC1(R7) SHLL 12 + IDLOC2(R7)  ELSE
1C C700  07 0850  1122          IF R0 = FUNCID THEN
10 070C  07 0850  1123          BEGIN  COMMENT PARAMETERLESS FUNCTION PROCEDURE;
10 070C  07 0850  1124             MARKRECORDS;  CALLPROPROCWOPARAM;  R0 := 1;  PROCPARAM;
10 0724  07 0850  1125          END ELSE
10 0724  07 0850  1126          IF R0 = RCCLID THEN
10 0730  07 0850  1127          BEGIN  COMMENT RECORD CREATOR, UNITIALIZED FIELDS;
10 0730  07 0850  1128             RECORDALLOCATE;  R3 := #80300000;  R6 := 9;
10 073C  07 0850  1129             RELEASE;  R5 := R5-4;
10 074C  07 0850  1130          END ELSE
10 074C  07 0850  1131          IF R0 = NULLL THEN R3 := NULLREF ELSE
10 075C  07 0850  1132          BEGIN  COMMENT NUMBER, STRYNG, BIT, TRUE, FALSE;
10 0760  07 0850  1133             R7 := R7 ++ CPTBASE; R3 := CONSPTTAB(R7) AND #FFF OR #E000;
10 0770  07 0850  1134          END;
10 0770  07 0850  1135          LM(R0,R1,SAVE01);  R7 := SAVER7;
10 0778  07 0850  1136       END;
10 077A  07 0850  1137
10 077A  07 0850  1138    PROCEDURE LOADREG(R1); COMMENT WRITES INSTRUCTION TO LOAD A REGISTER;
10 077A  07 0850  1139       BEGIN LOGICAL SAVER1,SAVER3,SAVER6,SAVEADD,TEMP;
10 077A  07 0864  1140          SAVER1 := R1; SAVER3 := R3; SAVER6 := R6;
10 0786  07 0864  1141          IF R7 >= 0 THEN
10 078C  07 0864  1142          BEGIN GETTYPE; R6 := R3; TEMP := R6; GETADDRESS; SAVEADD := R3;
10 07A6  07 0864  1143             R6 := TEMP;     R3 := R6;
10 07AC  07 0864  1144             IF R3 = 0 OR R3 > 5 THEN R3 := 1;
10 07BE  07 0864  1145             CASE R3 OF
10 07BE  07 0864  1146             BEGIN
10 07BE  07 0864  1147                BEGIN TEST(PAIRFLAG); IF = THEN PRGENREG ELSE GENREG; END;
10 07E2  07 0864  1148                BEGIN R0 := #C0000000; FLREG; END;
10 07F6  07 0864  1149                BEGIN R0 := #80000000; FLREG; END;
10 080A  07 0864  1150                BEGIN R0 := #40000000; PRFLREG; END;
10 081E  07 0864  1151                BEGIN R0 := #00000000; PRFLREG; END;
10 0832  07 0864  1152             END;
10 084A  07 0864  1153             TEMP := R0;
10 084E  07 0864  1154             R3 := SAVEADD;     R0 := R0 OR #08000000;
10 0856  07 0864  1155             ASSEMBLE;
10 0862  07 0864  1156             IF R6 = 1 AND DIVIDE THEN
10 0872  07 0864  1157             BEGIN  R0 := TEMP OR #8E000020;  EMIT;
10 0886  07 0864  1158             END;
10 0886  07 0864  1159             R7 := R6 SHLL 24 OR SIGN;
10 0890  07 0864  1160          END ELSE
```

```
10 0890  07 0864  1161          BEGIN COMMENT SET UP INSTRUCTION FOR LOADING A REGISTER FROM
10 0894  07 0864  1162             EITHER A COMPUTED ADDRESS OR A LOCAL STACK ADDRESS.  TYPE IN
10 0894  07 0864  1163             LOW BITS OF R6;
10 0894  07 0864  1164             LOGICAL SAVETYPE, SAVER3;
10 0894  07 086C  1165             R6 := R7 SHLL 1 SHRL 25;
10 089E  07 086C  1166             SAVETYPE := R6;
10 08A2  07 086C  1167             R3 := LSTACKM4(R5);  IF R3 >= 0 THEN
10 08AC  07 086C  1168             BEGIN  R0 := R3 SHRL 12; R5 := R5 - 4;
10 08B6  07 086C  1169                IF R0 < CLN THEN RELEASE; COMMENT COMPUTED ADDRESS;
10 08CA  07 086C  1170                SAVER3 := R3;  IF R6 > 5 THEN R6 := 1;
10 08DA  07 086C  1171                CASE R6 OF
10 08DA  07 086C  1172                BEGIN
10 08DA  07 086C  1173                   IF PAIRFLAG THEN PRGENREG ELSE GENREG;
10 08FE  07 086C  1174                   BEGIN R0 := #C0000000; FLREG; END;
10 0912  07 086C  1175                   BEGIN R0 := #80000000; FLREG; END;
10 0926  07 086C  1176                   BEGIN R0 := #40000000; PRFLREG; END;
10 093A  07 086C  1177                   BEGIN R0 := #00000000; PRFLREG; END;
10 094E  07 086C  1178                END;
10 0966  07 086C  1179                TEMP := R0;  R0 := R0 OR #08000000; R6 := SAVETYPE;
10 0972  07 086C  1180                R3 := SAVER3; ASSEMBLE;
10 0982  07 086C  1181                IF R6 = 1 AND DIVIDE THEN
10 0992  07 086C  1182                BEGIN  R0 := TEMP OR #8E000020;  EMIT;
10 09A6  07 086C  1183                END;
10 09A6  07 086C  1184             END;
10 09A6  07 086C  1185          END;
10 09A6  07 086C  1186          R6 := SAVER6;  R3 := SAVER3;  R1 := SAVER1;
10 09B2  07 086C  1187       END;
10 09B4  07 086C  1188
10 09B4  07 086C  1189    SEGMENT PROCEDURE  LEVEL1(R1);
10 09B4  07 086C  1190     BEGIN  LOGICAL SAVER1;
11 0000  07 0870  1191
11 0000  07 0870  1192     PROCEDURE INTREALCODE(R1);
11 0000  07 0870  1193        BEGIN  COMMENT GENERATE IN-LINE CODE TO FLOAT AN INTEGER;
11 0004  07 0870  1194           INTEGER SAVER1, SAVEFL, GPR, FPR;  SAVER1 := R1; SAVEFL := R0;
11 000C  07 0880  1195           LOADREG; R3 := LSTACKM4(R5); R0 := R3 XOR SIGN; GPR := R0;
11 0026  07 0880  1196           R6 := 1;  RELEASE; R5 := R5-4;  R0 := SAVEFL; FLREG; FPR := R0;
11 004E  07 0880  1197           R0 := R0 SHRL 4 OR FPR OR #2B000000;  EMIT;
11 0066  07 0880  1198           R0 := GPR SHRL 4 OR GPR OR #12000000;  EMIT;
11 0082  07 0880  1199           R0 := #47A0E008 + INSCOUNTER;  EMIT;
11 0096  07 0880  1200           R0 := #68000000 OR FPR OR FCON2;  EMIT;
11 00AE  07 0880  1201           R0 := #50000000 OR GPR OR FCON1 + 4;  EMIT;
11 00CA  07 0880  1202           R0 := #6A000000 OR FPR OR FCON1;  EMIT;
11 00E2  07 0880  1203           R1 := FSTACK(R4-4) AND MASK OR SAVEFL; FSTACK(R4-4) := R1;
11 00F2  07 0880  1204           R1 := SAVER1;
11 00F6  07 0880  1205        END;
11 00F8  07 0880  1206
11 00F8  07 0880  1207     SEGMENT PROCEDURE  CONVERT(R1);
11 00F8  07 0880  1208        BEGIN  COMMENT EMIT CODE TO CONVERT TERMINALS FROM DECLARED TYPE TO
12 0000  07 0880  1209                       TYPE INDICATED IN 2ND BYTE OF TERMINAL NODE;
12 0000  07 0880  1210
12 0000  07 0880  1211           LOGICAL SAVER6,SAVERA,SAVER0,SAVER3;
12 0000  07 0890  1212           LOGICAL SAVEFL;
12 0000  07 0894  1213           PROCEDURE INTREAL(R1);
12 0000  07 0894  1214              BEGIN  LOGICAL SAVER1; SAVER1 := R1;
12 00C8  07 0898  1215                 R0 := SAVEFL; INTREALCODE; R7 := SAVER6; R1 := SAVER1;
12 00C0  07 0898  1216              END;
12 0022  07 0898  1217
12 0022  07 0898  1218           PROCEDURE INTCOMPLEX(R1);
```

```
12 0022  07 0898  1219          BEGIN  LOGICAL SAVER1;  SAVER1 := R1;
12 0026  07 089C  1220             INTREAL;  R0:= SAVEFL;  FLREG;  R5:=R5-4; R3:=LSTACK(R5);
12 0042  07 089C  1221             R0 := R3;  R3 := R3 XOR SIGN SHRL 4 OR LSTACKM4(R5);
12 0050  07 089C  1222             LSTACKM4(R5) := R3;  R3 := R0;
12 0056  07 089C  1223             R0 := R0 XOR SIGN OR S;  R6 := 3;  ASSEMBLE;  R4 := R4 - 4;
12 006E  07 089C  1224             R1:=FSTACK(R4-4) AND MASK OR SAVEFL; FSTACK(R4-4):=R1;
12 0082  07 089C  1225             R7 := SAVER6;  R1 := SAVER1;
12 008A  07 089C  1226          END;
12 008C  07 089C  1227
12 008C  07 089C  1228       PROCEDURE REALCOMPLEX(R1);
12 008C  07 089C  1229          BEGIN  LOGICAL SAVER1;  SAVER1 := R1;
12 0090  07 08A0  1230             LOADREG; R0:=SAVEFL; FLREG;  R3:=R0 OR SIGN; R0:=R0 OR S;
12 00B6  07 08A0  1231             R6 := 3;  ASSEMBLE;  R3 := LSTACKM4(R5) XOR SIGN SHRL 4 OR
12 00D2  07 08A0  1232             LSTACKM8(R5);  LSTACKM8(R5) := R3;  R5 := R5 - 4;
12 00DE  07 08A0  1233             R1:=FSTACK(R4-8) AND MASK OR SAVEFL;  FSTACK(R4-8):=R1;
12 00EE  07 08A0  1234             R4 := R4 - 4;  R7 := SAVER6;  R1 := SAVER1;
12 00FA  07 08A0  1235          END;
12 00FC  07 08A0  1236
12 00FC  07 08A0  1237       PROCEDURE RLONGREAL(R1);
12 00FC  07 08A0  1238          BEGIN  LOGICAL SAVER1,TEMP;  SAVER1 := R1;
12 0100  07 08A8  1239             GETADDRESS;  R0 := SAVEFL;  TEMP := R3;
12 0114  07 08A8  1240             FLREG;  R0 := R0 OR S;  R3 := LSTACKM4(R5);  R6 := 3;
12 012C  07 08A8  1241             ASSEMBLE;  R0 := LSTACKM4(R5) XOR SIGN OR #78000000 OR
12 0144  07 08A8  1242             TEMP;  EMIT;  R1 := SAVER1;  R7 := SAVER6;
12 015C  07 08A8  1243          END;
12 015E  07 08A8  1244       PROCEDURE LONGSHORT(R1);
12 015E  07 08A8  1245          BEGIN  INTEGER SAVER1;  SAVER1 := R1;  LOADREG;
12 016E  07 08AC  1246             R1:=FSTACK(R4-4) AND MASK OR SAVEFL; FSTACK(R4-4):=R1;
12 017E  07 08AC  1247             R7 := SAVER6;  R1 := SAVER1;
12 0186  07 08AC  1248          END;
12 0188  07 08AC  1249
12 0188  07 08AC  1250       PROCEDURE CLONGCOMPLEX(R1);
12 0188  07 08AC  1251          BEGIN  LOGICAL SAVER1,TEMP;  SAVER1 := R1;
12 018C  07 08B4  1252             GETADDRESS;  TEMP := R3;  R0 := SAVEFL;
12 01A0  07 08B4  1253             FLREG;  R0 := R0 OR S;  R3 := LSTACKM4(R5);  R6 := 3;
12 01B8  07 08B4  1254             ASSEMBLE; R0:=SAVEFL; FLREG; R0:=R0 OR S; R3:=LSTACKM4(R5);
12 01D8  07 08B4  1255             R6 := 3;  ASSEMBLE;  R0 := LSTACKM4(R5) XOR SIGN
12 01F0  07 08B4  1256             OR #78000000 OR TEMP ++ 4;  EMIT;
12 020C  07 08B4  1257             R0 := LSTACKM8(R5) XOR SIGN OR #78000000 OR TEMP;  EMIT;
12 0228  07 08B4  1258             R0 := LSTACKM4(R5) XOR SIGN SHRL 4 OR LSTACKM8(R5);
12 0238  07 08B4  1259             LSTACKM8(R5) := R0;  R5 := R5 - 4;  R4 := R4 - 4;
12 0244  07 08B4  1260             R1:=FSTACK(R4-4) AND MASK OR SAVEFL; FSTACK(R4-4):=R1;
12 0254  07 08B4  1261             R7 := SAVER6;  R1 := SAVER1;
12 025C  07 08B4  1262          END;
12 025E  07 08B4  1263       PROCEDURE REALLONGCOMPLEX(R1);
12 025E  07 08B4  1264          BEGIN  LOGICAL SAVER1;  SAVER1 := R1;
12 0262  07 08B8  1265             RLONGREAL; R0:=SAVEFL; FLREG; R3:=R0 OR SIGN;
12 027C  07 08B8  1266             R0 := R0 OR S;  R6 := 3;  ASSEMBLE;
12 0290  07 08B8  1267             R0 := LSTACKM4(R5) XOR SIGN SHRL 4 OR LSTACKM8(R5);
12 02A0  07 08B8  1268             LSTACKM8(R5) := R0;  R5 := R5 - 4;
12 02A8  07 08B8  1269             R1:=FSTACK(R4-4) AND MASK OR SAVEFL; FSTACK(R4-4):=R1;
12 02B8  07 08B8  1270             R4 := R4 - 4;  R7 := SAVER6;  R1 := SAVER1;
12 02C4  07 08B8  1271          END;
12 02C6  07 08B8  1272
12 02C6  07 08B8  1273       SAVER3 := R3;  SAVERA := R10;  SAVER0 := R0;
12 02D2  07 08B8  1274       IF R7 >= 0 THEN
12 02D8  07 08B8  1275       BEGIN  LOGICAL SAVER1;  SAVER1 := R1;
12 02DC  07 08BC  1276          R10 := R7 SHLL 8 SHRL 24;  IF R10 ¬= 0 THEN
```

```
12 02EC  07 08BC  1277          BEGIN  GETTYPE;
12 02F8  07 08BC  1278             R0 := R10 SHLL 24 OR SIGN;  R6 := R6 SHLL 8 SHRL 8 OR R0;
12 030C  07 08BC  1279             R1 := 5 - R10 SHLL 30;  SAVEFL := R1;
12 031A  07 08BC  1280             SAVER6 := R6;
12 031E  07 08BC  1281             IF R3 <= 5 THEN
12 0326  07 08BC  1282             CASE R10 OF
12 0326  07 08BC  1283             BEGIN
12 0326  07 08BC  1284             BEGIN END;
12 032E  07 08BC  1285             BEGIN COMMENT CONVERT TO REAL;
12 0332  07 08BC  1286                CASE R3 OF
12 0332  07 08BC  1287                   BEGIN INTREAL;  NULL;  LONGSHORT;
12 034A  07 08BC  1288                END;
12 035A  07 08BC  1289             END;
12 035A  07 08BC  1290             BEGIN  COMMENT CONVERT TO LONG REAL;
12 035E  07 08BC  1291                CASE R3 OF
12 035E  07 08BC  1292                BEGIN  INTREAL;  RLONGREAL;  NULL;
12 0376  07 08BC  1293                END;
12 0386  07 08BC  1294             END;
12 0386  07 08BC  1295             BEGIN  COMMENT CONVERT TO COMPLEX;
12 038A  07 08BC  1296                CASE R3 OF
12 038A  07 08BC  1297                BEGIN  INTCOMPLEX;  REALCOMPLEX;  REALCOMPLEX;  NULL;
12 03AA  07 08BC  1298                   LONGSHORT;
12 03B2  07 08BC  1299                END;
12 03CA  07 08BC  1300             END;
12 03CA  07 08BC  1301             BEGIN  COMMENT CONVERT TO LONG COMPLEX;
12 03CE  07 08BC  1302                CASE R3 OF
12 03CE  07 08BC  1303                BEGIN  INTCOMPLEX;  REALLONGCOMPLEX;  REALCOMPLEX;
12 03EA  07 08BC  1304                   CLONGCOMPLEX;  NULL;
12 03F6  07 08BC  1305                END;
12 040E  07 08BC  1306             END;
12 040E  07 08BC  1307             END;
12 0426  07 08BC  1308             R6 := SAVER6;
12 042A  07 08BC  1309          END;
12 042A  07 08BC  1310          R1 := SAVER1;
12 042E  07 08BC  1311       END;
12 042E  07 08BC  1312       R10 := SAVERA;  R3 := SAVER3;  R0 := SAVER0;
12 043A  07 08BC  1313    END;

                 SEGMENT 12  NAME = SEG#12      LENGTH = 0458  BASE REG = 15

11 00F8  07 08BC  1314
11 00F8  07 08BC  1315    SEGMENT PROCEDURE  CONVERTRESULT(R1);
11 00F8  07 08BC  1316       BEGIN  LOGICAL SAVER1,SAVETYPE;   BYTE KEEPREG;
13 0000  07 08C5  1317          LOGICAL SAVER3,SAVEFL;
13 0000  07 08D0  1318          PROCEDURE INTREAL(R1);
13 0000  07 08D0  1319             BEGIN  LOGICAL SAVER1; SAVER1 := R1;
13 00C8  07 08D4  1320                R0 := SAVEFL; INTREALCODE;
13 0018  07 08D4  1321                R7 := SAVETYPE SHLL 24 OR SIGN; R1 := SAVER1;
13 0028  07 08D4  1322             END;
13 002A  07 08D4  1323
13 002A  07 08D4  1324          PROCEDURE INTCOMPLEX(R1);
13 002A  07 08D4  1325             BEGIN  LOGICAL SAVER1;  SAVER1 := R1;
13 002E  07 08D8  1326                INTREAL;  R0 := SAVEFL;  FLREG;  R5 := R5 - 4;
13 0046  07 08D8  1327                R3 := LSTACK(R5) ;  R0 := R3;
13 004C  07 08D8  1328                R3 := R3 XOR SIGN SHRL 4 OR LSTACKM4(R5) OR SIGN;
13 005C  07 08D8  1329                LSTACKM4(R5) := R3;  R3 := R0;  R4 := R4 - 4;
13 0066  07 08D8  1330                R0 := R0 XOR SIGN OR S;  R6 := 3;  ASSEMBLE;
13 007E  07 08D8  1331             R1 := FSTACK(R4-4) AND MASK OR SAVEFL;  FSTACK(R4-4) :=R1;
```

```
13 008E  07 08D8  1332              R1 := SAVER1;  R7 := SAVETYPE SHLL 24 OR SIGN;
13 009E  07 08D8  1333           END;
13 00A0  07 08D8  1334
13 00A0  07 08D8  1335        PROCEDURE RLONGREAL(R1);
13 00A0  07 08D8  1336           BEGIN LOGICAL SAVER1,TEMP;  SAVER1 := R1;
13 00A4  07 08E0  1337              R3 := LSTACKM4(R5);
13 00A8  07 08E0  1338              IF R3 >= 0 THEN
13 00AE  07 08E0  1339              BEGIN  TEMP := R3;  R3 := R3 AND #F000 SHRL 12;
13 00BA  07 08E0  1340                 IF R3 < CLN AND ¬KEEPREG THEN
13 00CA  07 08E0  1341                 BEGIN  R10 := R7;  R6 := 1;  R3 := TEMP;  RELEASE;
13 00E0  07 08E0  1342                 END;
13 00EC  07 08E0  1343                 R0 := SAVEFL;  R5 := R5 - 4;
13 00E8  07 08E0  1344                 FLREG;  R0 := R0 OR S;  R3 := LSTACKM4(R5);  R6 := 3;
13 0100  07 08E0  1345                 ASSEMBLE;  R0 := LSTACKM4(R5) XOR SIGN OR #78000000 OR
13 0118  07 08E0  1346                 TEMP;  EMIT;
13 0128  07 08E0  1347              END ELSE
13 0128  07 08E0  1348              BEGIN R3 := NEXTADDR + 3 AND SMASK;  NEXTADDR := R3;
13 013C  07 08E0  1349                 R6 := 2;    R0 := LSTACKM4(R5) SHLL
13 0144  07 08E0  1350                 1 SHRL 1; R10:=R0; ASSEMBLE; R0:=R10 OR #2F000000;
13 0160  07 08E0  1351                 R3 := R10 SHRL 4;  R0 := R0 OR R3;  EMIT;
13 0174  07 08E0  1352                 R3 := NEXTADDR;  R6 := 2;  R0 := R10 OR #08000000;
13 0182  07 08E0  1353                 R6 := 2;  ASSEMBLE;
13 0192  07 08E0  1354
13 0192  07 08E0  1355                 R10 := 4;  INCRADDR;
13 01A2  07 08E0  1356              END;
13 01A2  07 08E0  1357           R1 := FSTACK(R4-4) AND MASK OR SAVEFL;  FSTACK(R4-4) :=R1;
13 01B2  07 08E0  1358              R1 := SAVER1;  R7 := SAVETYPE SHLL 24 OR SIGN;
13 01C2  07 08E0  1359           END;
13 01C4  07 08E0  1360        PROCEDURE REALCOMPLEX(R1);
13 01C4  07 08E0  1361           BEGIN  LOGICAL SAVER1;  SAVER1 := R1;
13 01C8  07 08E4  1362              LOADREG;  R0 := SAVEFL;  FLREG;  R3 := R0 OR SIGN;
13 01EA  07 08E4  1363              R0 := R0 OR S;  R6 := SAVEFL SHLL 1;
13 01F6  07 08E4  1364              IF R6 < 0 THEN R6 := 2 ELSE R6 := 3;  ASSEMBLE;
13 0214  07 08E4  1365              R3 := LSTACKM4(R5) XOR SIGN SHRL 4 OR LSTACKM8(R5);
13 0224  07 08E4  1366              LSTACKM8(R5) := R3;  R5 := R5 - 4;  R4 := R4 - 4;
13 0230  07 08E4  1367           R1 := FSTACK(R4-4) AND MASK OR SAVEFL;  FSTACK(R4-4) :=R1;
13 0240  07 08E4  1368              R7 := SAVETYPE SHLL 24 OR SIGN;  R1 := SAVER1;
13 0250  07 08E4  1369           END;
13 0252  07 08E4  1370
13 0252  07 08E4  1371        PROCEDURE LONGSHORT(R1);
13 0252  07 08E4  1372           BEGIN  LOGICAL SAVER1;  SAVER1 := R1;  LOADREG;
13 0262  07 08E8  1373              R1 := FSTACK(R4-4) AND MASK OR SAVEFL;
13 026E  07 08E8  1374              FSTACK(R4-4) := R1;  R7 := SAVETYPE SHLL 24 OR SIGN;
13 027E  07 08E8  1375              R1 := SAVER1;
13 0282  07 08E8  1376           END;
13 0284  07 08E8  1377
13 0284  07 08E8  1378        PROCEDURE CLONGCOMPLEX(R1);
13 0284  07 08E8  1379           BEGIN  LOGICAL SAVER1,TEMP;  SAVER1 := R1;
13 0288  07 08F0  1380              R3 := LSTACKM4(R5);  IF R3 < 0 THEN
13 0292  07 08F0  1381              BEGIN  R0 := R3 AND MASK5 OR SIGN;  LSTACKM4(R5) := R0;
13 02A0  07 08F0  1382                 R3 := R3 SHLL 4 AND MASK5 OR SIGN;  R5 := R5 + 4;
13 02B0  07 08F0  1383                 LSTACKM4(R5) := R3;
13 02B4  07 08F0  1384              END ELSE
13 02B4  07 08F0  1385              BEGIN  R3 := R3 + 4;  R5 := R5 + 4;  LSTACKM4(R5) := R3;
13 02C4  07 08F0  1386              END;
13 02C4  07 08F0  1387              SET(KEEPREG);  RLONGREAL;  R0 := LSTACKM4(R5);  TEMP := R0;
13 02D0  07 08F0  1388              R0 := LSTACKM8(R5);  IF R0 > 0 THEN R4 := R4 - 4;
13 02E2  07 08F0  1389              R5 := R5 - 4;  RESET(KEEPREG);  RLONGREAL;
```

```
13 02EE  07 08F0  1390              R0 := TEMP SHLL 1 SHRL 5 OR LSTACKM4(R5) OR SIGN;
13 0302  07 08F0  1391              LSTACKM4(R5) := R0;
13 03C6  07 08F0  1392           R1 := FSTACK(R4-4) AND MASK OR SAVEFL;  FSTACK(R4-4) :=R1;
13 0316  07 08F0  1393              R1 := SAVER1;  R7 := SAVETYPE SHLL 24 OR SIGN;
13 0326  07 08F0  1394           END;
13 0328  07 08F0  1395
13 0328  07 08F0  1396        PROCEDURE REALLONGCOMPLEX(R1);
13 0328  07 08F0  1397           BEGIN  LOGICAL SAVER1;  SAVER1 := R1;
13 032C  07 08F4  1398           RLONGREAL;  R0 := SAVEFL;  FLREG;  R3 := R0 OR SIGN;
13 0346  07 08F4  1399              R0 := R0 OR S;  R6 := 3;  ASSEMBLE;
13 035A  07 08F4  1400              R0 := LSTACKM4(R5) XOR SIGN SHRL 4 OR LSTACKM8(R5);
13 036A  07 08F4  1401              LSTACKM8(R5) := R0;  R5 := R5 - 4;  R4 := R4 - 4;
13 0376  07 08F4  1402           R1 := FSTACK(R4-4) AND MASK OR SAVEFL;  FSTACK(R4-4) :=R1;
13 0386  07 08F4  1403              R7 := SAVETYPE SHLL 24 OR SIGN;  R1 := SAVER1;
13 0396  07 08F4  1404           END;
13 C398  07 08F4  1405
13 0398  07 08F4  1406        R6 := SAVEL;  SAVER3 := R3;    RESET(KEEPREG);
13 03A4  07 08F4  1407        R10 := TREE(R6) SHLL 8 SHRL 24;
13 C3B0  07 08F4  1408        IF R10 ¬= 0 THEN
13 03B6  07 08F4  1409        BEGIN  COMMENT CONVERT TO TYPE IN RA;
13 03B6  07 08F4  1410           SAVER1 := R1;  SAVETYPE := R10;
13 03BE  07 08F4  1411           R1 := 5 - R10 SHLL 30;  SAVEFL := R1;
13 03CC  07 08F4  1412           R3 := R7 SHLL 1 SHRL 25;  COMMENT R3 HAS CURRENT TYPE;
13 03D6  07 08F4  1413           IF R3 <= 5 THEN
13 03DE  07 08F4  1414           CASE R10 OF
13 03DE  07 08F4  1415           BEGIN
13 03DE  07 08F4  1416           NULL;
13 03E6  07 08F4  1417           BEGIN  CASE R3 OF
13 03EA  07 08F4  1418                    BEGIN  INTREAL;  NULL;  LONGSHORT;
13 0402  07 08F4  1419              END;
13 0412  07 08F4  1420           END;
13 0412  07 08F4  1421           BEGIN  CASE R3 OF
13 0416  07 08F4  1422              BEGIN  INTREAL;  RLONGREAL;  NULL;
13 042E  07 08F4  1423              END;
13 043E  07 08F4  1424           END;
13 043E  07 08F4  1425           BEGIN  CASE R3 OF
13 0442  07 08F4  1426              BEGIN  INTCOMPLEX;  REALCOMPLEX;  REALCOMPLEX;  NULL;
13 0462  07 08F4  1427                    LONGSHORT;
13 046A  07 08F4  1428              END;
13 0482  07 08F4  1429           END;
13 0482  07 08F4  1430           BEGIN  CASE R3 OF
13 0486  07 08F4  1431              BEGIN  INTCOMPLEX;  REALLONGCOMPLEX;  REALCOMPLEX;
13 04A2  07 08F4  1432                 CLONGCOMPLEX;  NULL;
13 04AE  07 08F4  1433              END;
13 04C6  07 08F4  1434           END;
13 04C6  07 08F4  1435           END;
13 04DE  07 08F4  1436           R1 := SAVER1;
13 04E2  07 08F4  1437        END;
13 04E2  07 08F4  1438        R3 := SAVER3;
13 04E6  07 08F4  1439     END;

                         SEGMENT 13  NAME = SEG#13      LENGTH = 0518  BASE REG = 15

11 0CF8  07 08F4  1440
11 00F8  07 08F4  1441     PROCEDURE DIVREM(R1);  COMMENT GENERATES CODE FOR DIVISION OF INTEGER
11 00F8  07 08F4  1442        ARGUMENTS;
11 00F8  07 08F4  1443        BEGIN LOGICAL SAVER1;   SAVER1 := R1;
11 00FC  07 08F8  1444           SET(PAIRFLAG); SET(DIVIDE);
```

```
11 0104   07 08F8   1445          R10 := R6; R6 := 1;
11 010A   07 08F8   1446          IF R7 >= 0 THEN
11 0110   07 08F8   1447          BEGIN R10 := TREE(R10); IF R10 < 0 THEN
11 011A   07 08F8   1448             BEGIN  LOADREG;
11 0126   07 08F8   1449                R0 := LSTACKM4(R5); FORCEPAIR;
11 0136   07 08F8   1450                R5 := R5-4; R3 := LSTACKM4(R5);
11 013E   07 08F8   1451                IF R3 < 0 THEN RELEASE;
11 0150   07 08F8   1452                R10:=R3 SHRL 12; IF R3>=0 AND R10>=CLN THEN ADJSTACKS;
11 0170   07 08F8   1453                R1 := LSTACK(R5);  LSTACKM4(R5) := R1;
11 0178   07 08F8   1454             END ELSE
11 0178   07 08F8   1455             BEGIN LOGICAL SAVEADD;  GETADDRESS;  SAVEADD := R3;
11 018C   07 08FC   1456                R0 := LSTACKM4(R5);  IF R0 < 0 THEN FORCEPAIR ELSE
11 01A2   07 08FC   1457                BEGIN COMMENT LEFT SIDE STORED WHILE PROCESSING RIGHT;
11 01A6   07 08FC   1458                   LOADREG;  R0 := LSTACKM4(R5) XOR SIGN;
11 01EA   07 08FC   1459                END;  R3 := SAVEADD;
11 01BE   07 08FC   1460             END;
11 01BE   07 08FC   1461          END ELSE
11 01BE   07 08FC   1462          BEGIN R10 := TREE(R10); IF R10 < 0 THEN
11 01CC   07 08FC   1463             BEGIN
11 01CC   07 08FC   1464                LOADREG; R0 := LSTACKM4(R5); FORCEPAIR; R5 := R5-4;
11 01EC   07 08FC   1465                R3 := LSTACKM4(R5); IF R3 < 0 THEN RELEASE;
11 0202   07 08FC   1466                R0 := R0 OR SIGN; R10 := LSTACK(R5); LSTACKM4(R5) := R10;
11 020E   07 08FC   1467                ADJSTACKS;
11 021A   07 08FC   1468             END ELSE
11 021A   07 08FC   1469             BEGIN R5 := R5-4; LOADREG; R0 := LSTACKM4(R5); FORCEPAIR;
11 023E   07 08FC   1470                R3 := LSTACK(R5); RELEASE;
11 024E   07 08FC   1471             END;
11 024E   07 08FC   1472          END;
11 024E   07 08FC   1473          R0 := R0 OR D; ASSEMBLE;
11 025E   07 08FC   1474          R7 := R6 SHLL 24 OR SIGN;
11 0268   07 08FC   1475          RESET(PAIRFLAG); RESET(DIVIDE);
11 0270   07 08FC   1476          R1 := SAVER1;
11 0274   07 08FC   1477       END;
11 0276   07 08FC   1478    PROCEDURE BITSANDORARG2(R1); COMMENT SETS UP INSTRUCTION FOR ASSEMBLE
11 0276   07 08FC   1479       EXCEPT OPCODE PART.  MAY ALSO WRITE INSTRUCTION TO RELOAD BITS
11 0276   07 08FC   1480       VALUE;
11 0276   07 08FC   1481       BEGIN
11 0276   07 08FC   1482          LOGICAL SAVER1; SAVER1 := R1;
11 027A   07 0900   1483          R6 := R6 SHLL 1 SHRL 25;
11 0282   07 0900   1484          IF R7 >= 0 THEN
11 0288   07 0900   1485          BEGIN GETADDRESS;    R0 := LSTACKM4(R5);
11 0298   07 0900   1486             IF R7 < 0 THEN
11 029E   07 0900   1487             BEGIN R7 := #88000000;  LOADREG;  R0 := LSTACKM4(R5);  END;
11 02B2   07 0900   1488          END ELSE
11 02B2   07 0900   1489          BEGIN R0 := LSTACKM8(R5); R3 := LSTACKM4(R5);
11 02BE   07 0900   1490             IF R0 >= 0 THEN
11 02C4   07 0900   1491             BEGIN
11 02C4   07 0900   1492                IF R3 >= 0 THEN LOADREG;   R3 := LSTACKM8(R5);
11 02DA   07 0900   1493                R0 := LSTACKM4(R5);  LSTACKM8(R5) := R0;
11 02E2   07 0900   1494                R6 := 1;  ADJSTACKS;  R6 := 8;
11 02F6   07 0900   1495             END ELSE RELEASE; R5 := R5-4;
11 030A   07 0900   1496          END;
11 030A   07 0900   1497          R1 := SAVER1;
11 030E   07 0900   1498       END;
11 0310   07 0900   1499    PROCEDURE SHIFTAMOUNT(R1); COMMENT GENERATES INSTRUCTIONS TO LOAD
11 0310   07 0900   1500       ABSOLUTE VALUE OF SHIFT EXPRESSION;
11 0310   07 0900   1501       BEGIN LOGICAL SAVER1;   SAVER1 := R1;
11 0314   07 0904   1502          LOADREG;
```

```
11 0320  07 090C  1503          R0 := LSTACKM4(R5); R3 := R0; ASSEMBLE;
11 0332  07 0904  1504          R1 := SAVER1;
11 0336  07 0904  1505       END;
11 0338  07 0904  1506    PROCEDURE BITSSHIFTARG2(R1); COMMENT GENERATES INSTRUCTIONS TO SET UP
11 0338  07 0904  1507       SHIFT OF BITS VALUE.  R0 IS AN OUTPUT CONTAINING THE SHIFT INSTRUC-
11 0338  07 0904  1508       TION MINUS THE OPCODE;
11 0338  07 0904  1509       BEGIN LOGICAL SAVER1, TEMP;   SAVER1 := R1;
11 033C  07 090C  1510          R10 := R6;       R6 := 8;
11 0342  07 090C  1511          R10 := TREE(R10); IF R10 < 0 THEN
11 034C  07 090C  1512          BEGIN
11 034C  07 090C  1513             LOADREG; R0 := LSTACKM4(R5); TEMP := R0;
11 0360  07 090C  1514             R5 := R5-4; LOADREG; R3 := LSTACKM4(R5); RELEASE;
11 0380  07 090C  1515             R0 := TEMP; LSTACKM4(R5) := R0;
11 0388  07 090C  1516             R3 := R3 AND MASK5 SHRL 8; R0 := R0 AND MASK5 OR R3;
11 0396  07 090C  1517          END ELSE
11 0396  07 090C  1518          BEGIN
11 039A  07 090C  1519             SHIFTAMOUNT; R0 := LSTACKM4(R5) AND MASK5 SHRL 8;
11 03AA  07 090C  1520             TEMP := R0; R5 := R5-4; LOADREG;
11 03BE  07 090C  1521             R3 := TEMP; RELEASE;
11 03CE  07 090C  1522             R0 := LSTACKM4(R5) AND MASK5 OR TEMP;
11 03DA  07 090C  1523          END;
11 03DA  07 090C  1524          R1 := RSTACKM4(R2) + 4;  IF R1 ¬= R5 THEN ADJSTACKS;
11 03F4  07 090C  1525          R1 := SAVER1;
11 03F8  07 090C  1526       END;
11 03FA  07 090C  1527
11 03FA  07 090C  1528    PROCEDURE CALLALLOCATE(R1); COMMENT THIS PROCEDURE EMITS THE CODE
11 03FA  07 090C  1529       WHICH CALLS THE ALLOCATE ROUTINE;
11 03FA  07 090C  1530       BEGIN INTEGER SAVER1;  SAVER1 := R1;
11 03FE  07 0910  1531          R0 := #58602000 OR FP;      EMIT;
11 0412  07 0910  1532          R0 := #41606007; EMIT;  COMMENT ADDS 7 TO R6;
11 0422  07 0910  1533          R0 := #54600000 OR DUBLMASK;    EMIT;
11 0436  07 0910  1534          COMMENT SAVE INSCOUNTER FOR LA INSTRUCTION TO BE FIXED UP;
11 0436  07 0910  1535          R1 := LOGPOINTER + 4;    LOGPOINTER := R1;
11 0442  07 0910  1536          R0 := INSCOUNTER;    LOGSTACK(R1) := R0;
11 044A  07 0910  1537          R0 :=  #41006000;    EMIT;
11 045A  07 0910  1538          R0 := #45400000 OR ALLOCERR;       EMIT;
11 046E  07 0910  1539          R1 := SAVER1;
11 0472  07 0910  1540       END;
11 0474  07 0910  1541
11 0474  07 0910  1542     COMMENT  SELECTION INDICES FOR PROCEDURE ARRAYS;
11 0474  07 0910  1543      BYTE  INITAREF SYN 1,        INDEXOP SYN 2,
11 0474  07 0910  1544            INITADCL SYN 6,        LBOUND  SYN 5,
11 0474  07 0910  1545            FILLDESCRIP SYN 3,     CLOSEDESCRIP SYN 4;
11 0474  07 0910  1546
11 0474  07 0910  1547    SEGMENT PROCEDURE  ARRAYS(R1); COMMENT THIS PROCEDURE CONTAINS ALL PROC
11 0474  07 0910  1548       RELEVANT TO ARRAY DECLARATIONS AND REFERENCES.  THE APPROPRIATE
11 0474  07 0910  1549       PROCEDURE IS SELECTED BY THE PARAMETER R3;
11 0474  07 0910  1550       BEGIN
14 0000  07 0910  1551
14 0000  07 0910  1552        PROCEDURE INITAREF(R1);
14 0000  07 0910  1553          COMMENT SET UP STACK FOR AN ARRAY REFERENCE (ARG1 - INDX);
14 0000  07 0910  1554          BEGIN  LOGICAL SAVER1;  SAVER1 := R1;
14 0008  07 0914  1555             GETADDRESS;  R0 := TYPEINFO(R7) AND #F SHLL 4;
14 0020  07 0914  1556             R1 := R3 SHLL 8 OR R0 OR #1; STACKP4(R8) := R1;
14 0030  07 0914  1557             IC(R0,SIMPLETYPE(R7));  STC(R0,STACKP4(R8));
14 0038  07 0914  1558             COMMENT  STACKP4 FORMAT:
14 0038  07 0914  1559               SIMPLETYPE(0:7), ADDR(8:23), #DIMEN(24:27), CURR DIM(28:31);
14 0038  07 0914  1560             IF R0 ¬= 7 THEN R1 := 0 ELSE R1 := SIMTYPEINFO(R7) SHLL 16;
```

```
14 0050  C7 0914  1561            IC(R0,TYPE(R7));  IF R0 = 18 THEN R1 := R1 OR SIGN;
14 0060  C7 0914  1562            STACKP8(R8) := R1;
14 0064  C7 0914  1563            COMMENT  STACKP8 FORMAT:
14 0064  07 0914  1564              FORMAL SWITCH(0:0), STRING LEN(8:15), SUBARRAY MASK(16:31);
14 0064  07 0914  1565            R1 := SAVER1;
14 0068  C7 0914  1566          END;
14 006A  C7 0914  1567
14 006A  07 0914  1568        PROCEDURE INDEXOP(R1);
14 006A  07 0914  1569          COMMENT PERFORM AN INDEXING OPERATION (ARG2 - INDX);
14 006A  C7 0914  1570          BEGIN  LOGICAL SAVER1; INTEGER NDIMEN, DESPOINT;  SAVER1 := R1;
14 006E  C7 0920  1571             R0 := STACKP4(R8) AND #F0 SHRL 4;  NDIMEN := R0;
14 007E  C7 0920  1572             R0 := STACKP4(R8) SHRL 8 AND #FFFF;
14 008A  07 0920  1573             R1 := STACKP4(R8) AND #F * 12S - 8 + R0;  DESPOINT := R1;
14 00A0  C7 0920  1574             COMMENT  R1 -> (DELTA(I), L(I), U(I));
14 00A0  07 0920  1575             R1 := R7 SHRL 24;  IF R1 = ARSTAR THEN
14 00AE  07 0920  1576             BEGIN  COMMENT SUBARRAY DESIGNATOR;
14 00AE  07 0920  1577                R1 := STACKP4(R8) AND #F - 1;
14 00BA  07 0920  1578                R0 := #1 SHLL R1 OR STACKP8(R8);  STACKP8(R8) := R0;
14 00CA  07 0920  1579                IF R1 = 0 THEN
14 00D0  C7 0920  1580                BEGIN  COMMENT FIRST DIMENSION, ESTABLISH REGISTER;
14 00D0  07 0920  1581                   GENREG;  R0 := R0 SHRL 4 OR LSTACKM4(R5);
14 00E4  07 0920  1582                   R0 := R0 AND #00FF0000 OR #18000000; EMIT;
14 00F8  07 0920  1583                END;
14 00F8  07 0920  1584             END ELSE
14 00F8  07 0920  1585             BEGIN  LOADREG;  R10 := LSTACKM4(R5) AND #00F00000;
14 0110  07 0920  1586                IF CHECKFLAG THEN
14 0118  07 0920  1587                BEGIN  R0 := #59000008 ++ DESPOINT OR R10;  EMIT;
14 012E  07 0920  1588                   R0 := #45100000 OR ARRAYERR;  EMIT;
14 0142  07 0920  1589                END;
14 0142  07 0920  1590             R0 := #5B000004 ++ DESPOINT OR R10;  EMIT;
14 0158  07 0920  1591             R0 := STACKP4(R8) AND #F; IF R0 = 1 THEN
14 0168  07 0920  1592             BEGIN  COMMENT FIRST DIMENSION, TYPE TO R3;
14 0168  C7 0920  1593                R3 := 0; IC(R3,STACKP4(R8));  R0 := STACKP8(R8);
14 0174  07 0920  1594                IF R3 = 7 OR R0 < 0 THEN
14 0182  C7 0920  1595                BEGIN  COMMENT STRING OR FORMAL - MULTIPLY;
14 0182  07 0920  1596                   R0 := #4C000002 ++ DESPOINT OR R10; EMIT;
14 0198  07 0920  1597                END ELSE
14 0198  07 0920  1598                IF R3 ¬= 6 THEN
14 01A4  C7 0920  1599                BEGIN  COMMENT SHIFT;
14 01A4  07 0920  1600                   IF R3 <= 2 OR R3 >= 8 THEN R0 := 2 ELSE
14 01B8  07 0920  1601                   IF R3 = 5 THEN R0 := 4 ELSE R0 := 3;
14 01D0  07 0920  1602                   R0 := R0 OR #89000000 OR R10;  EMIT;
14 01E2  07 0920  1603                END;
14 01E2  07 0920  1604             END ELSE
14 01E2  07 0920  1605             BEGIN  COMMENT NOT FIRST DIMENSION;
14 01E6  07 0920  1606                R0 := #4C000002 ++ DESPOINT OR R10;  EMIT;
14 01FC  07 0920  1607                R3 := LSTACKM4(R5-4);
14 0200  C7 0920  1608                IF R3 < 0 THEN
14 0206  C7 0920  1609                BEGIN  COMMENT  ACCUMULATING REGISTER NOT DUMPED;
14 0206  C7 0920  1610                   R0 := R10 SHRL 4 OR R3 XOR SIGN OR #1A000000; EMIT;
14 0222  07 0920  1611                   R3 := LSTACKM4(R5);  R6 := 1;  RELEASE;  R5 := R5-4;
14 023A  C7 0920  1612                END ELSE
14 023A  C7 0920  1613                BEGIN  COMMENT ACCUMULATING REGISTER DUMPED;
14 023E  07 0920  1614                   R0 := R10 OR R3 OR #5A000000;  EMIT;
14 0252  07 0920  1615                   R0 := LSTACKM4(R5); R5 := R5-4; LSTACKM4(R5) := R0;
14 025E  C7 0920  1616                   R0 := RSTACKM4(R2) - 4; RSTACKM4(R2) := R0;
14 026A  07 0920  1617                END;
14 026A  C7 0920  1618             END;
```

```
14 026A  07 0920  1619            END;
14 026A  07 0920  1620            R1 := STACKP4(R8) AND #F;  IF R1 = NDIMEN THEN
14 027A  07 0920  1621            BEGIN  COMMENT END OF INDEXING OPERATION;
14 027A  07 0920  1622               R1 := STACKP4(R8) SHRL 8 AND #FFFF;  R3 := LSTACKM4(R5);
14 028A  07 0920  1623               R0 := R3 AND #00F00000 OR R1 OR #5A000000;  EMIT;
14 02A2  07 0920  1624               R7 := STACKP4(R8) AND #FF000000 OR SIGN;
14 02AE  07 0920  1625               R10 := STACKP8(R8) AND #FFFF; IF = THEN
14 02BA  07 0920  1626               BEGIN  R3 := R3 SHRL 8 AND #FFFF; LSTACKM4(R5) := R3;
14 02C6  07 0920  1627                  R0 := 0; IC(R0,STACKP4(R8));
14 02CE  07 0920  1628                  IF R0 = 6 THEN R7 := R7 OR #10000 ELSE
14 02DA  07 0920  1629                  IF R0 = 7 THEN
14 02E6  07 0920  1630                  BEGIN  R0 := STACKP8(R8) AND #00FF0000; R7 := R7 OR R0;
14 02F0  07 0920  1631                  END;
14 02F0  07 0920  1632                  CONVERTRESULT;
14 02FA  07 0920  1633               END ELSE
14 02FA  07 0920  1634               BEGIN  COMMENT  PROCESS SUBARRAY ACTUAL PARAMETER;
14 02FE  07 0920  1635                  INTEGER LDESCRIP;
14 02FE  07 0924  1636                  R0 := STACKP4(R8) SHRL 8 AND #FFFF + 4; DESPOINT := R0;
14 0312  07 0924  1637                  R0 := NEXTADDR + 3 AND #FFFC; NEXTADDR := R0;
14 0322  07 0924  1638                  R0 := R0 + 4; LDESCRIP := R0;  R0 := 0; NDIMEN := R0;
14 0332  07 0924  1639                  WHILE R10 ¬= 0 DO
14 0338  07 0924  1640                  BEGIN  R0 := R10 AND #1;  IF ¬= THEN
14 0342  07 0924  1641                     BEGIN  COMMENT STARRED DIMENSION;
14 0342  07 0924  1642                        R0 := NDIMEN + 1;  NDIMEN := R0;  R1 := LDESCRIP;
14 0352  07 0924  1643                        R0 := #D20B0000 OR R1;  R3 := DESPOINT SHLL 16;
14 0360  07 0924  1644                        R1 := R1 + 12;  LDESCRIP := R1;  EMIT;
14 0374  07 0924  1645                     END;
14 0374  07 0924  1646                     R1 := DESPOINT + 12; DESPOINT := R1;
14 0380  07 0924  1647                     R10 := R10 SHRL 1;
14 0384  07 0924  1648                  END;
14 0388  07 0924  1649                  R0 := NEXTADDR;  LDESCRIP := R0;
14 0390  07 0924  1650                  R10 := NDIMEN * 12S + 4;  INCRADDR;
14 03A8  07 0924  1651                  R10 := LSTACKM4(R5) AND #00F00000;
14 03B0  07 0924  1652                  R0 := #50000000 OR R10 OR LDESCRIP;  EMIT;
14 03C6  07 0924  1653                  R0 := R10 SHLL 4 OR SIGN;  R1 := NDIMEN SHLL 16 OR R0;
14 03DA  07 0924  1654                  R0 := R7 SHRL 24 AND #7F;
14 03E4  07 0924  1655                  IF R0 = 7 THEN R0 := STACKP8(R8) AND #FF0000 SHRL 8 + 7;
14 03F8  07 0924  1656                  R0 := R0 OR R1;  LSTACKM4(R5) := R0;
14 0402  07 0924  1657                  R0 := #41300000 OR LDESCRIP;  EMIT;  SET(SUBARFLAG);
14 041A  07 0924  1658               END;
14 041A  07 0924  1659            END ELSE
14 041A  07 0924  1660            BEGIN  COMMENT NOT LAST SUBSCRIPT, UPDATE STACK;
14 041E  07 0924  1661               R0 := STACKP4(R8) ++ 1; R1 := STACKP8(R8);
14 042A  07 0924  1662               R3 := R8 - SSIZE;  STACKP4(R3) := R0; STACKP8(R3) := R1;
14 0438  07 0924  1663            END;
14 0438  07 0924  1664            R1 := SAVER1;
14 043C  07 0924  1665         END;
14 043E  07 0924  1666
14 043E  07 0924  1667         PROCEDURE INITADCL(R1);
14 043E  07 0924  1668           COMMENT  SET UP STACK, STORE DELTA(1), FOR ARRAY DECL (AR,-ARG1).
14 043E  07 0924  1669             CALLED FROM CARD ARG2 - R8 ALREADY ADJUSTED;
14 043E  07 0924  1670           BEGIN  LOGICAL SAVER1;  SAVER1 := R1;
14 0442  07 0928  1671              R1 := R7 AND #FFFF;  R3 := IDLOC1(R1) SHLL 12 + IDLOC2(R1);
14 0454  07 0928  1672              R0 := R1 SHLL 16 OR R3 ++ 4;  STACKP4(R8) := R0;
14 0464  07 0928  1673              COMMENT  STACKP4 FORMAT:
14 0464  07 0928  1674                NT PTR(0:15), DESCRIPTOR ADDR(16:31);
14 0464  07 0928  1675              IC(R10,SIMPLETYPE(R1));  R10 := R10 AND #FF;
14 046C  07 0928  1676              IF R10 <= 2 OR R10 >= 8 THEN  R0 := 4  ELSE
```

```
14 0480  07 0928  1677            IF R10 < 5 THEN  R0 := 8  ELSE
14 0490  07 0928  1678            IF R10 = 5 THEN  R0 := 16  ELSE
14 04A0  07 0928  1679            IF R10 = 6 THEN  R0 := 1  ELSE  R0 := SIMTYPEINFO(R1) + 1;
14 04B0  07 0928  1680            R0 := R0 OR #41000000;  EMIT;
14 04C0  07 0928  1681            R0 := #50000004 ++ R3;  EMIT;
14 04DE  07 0928  1682            R0 := R7 SHRL 16 AND #FF;  R10 := R10 SHLL 16;
14 04EC  07 0928  1683            R0 := R0 OR R10 OR SIGN;  STACKP8(R8) := R0;
14 04F6  07 0928  1684            COMMENT  STACKP8 FORMAT:
14 04F6  07 0928  1685              FIRST DIMEN SW(0:0), SI TYPE(1:15), #ID'S(16:31);
14 04F6  07 0928  1686            R1 := SAVER1;
14 04FA  07 0928  1687          END;
14 04FC  07 0928  1688
14 04FC  07 0928  1689        PROCEDURE PUTITIN(R1);
14 04FC  07 0928  1690          COMMENT  PUT U(I), DELTA(I+1) IN DESCRIPTOR;
14 04FC  07 0928  1691          BEGIN  LOGICAL SAVER1;  SAVER1 := R1;
14 0500  07 092C  1692             R3 := LSTACKM4(R5) AND #00F00000;
14 0508  07 092C  1693             R10 := STACKP4(R8) AND #FFFF;  COMMENT ADDR OF DELTA(I);
14 0510  07 092C  1694             R0 := #50000008 ++ R10 OR R3;  EMIT;  COMMENT U(I);
14 0524  07 092C  1695             R0 := #5B000004 ++ R10 OR R3;  EMIT;
14 0538  07 092C  1696             R0 := #45100000 OR UBLBERR;  EMIT;
14 054C  07 092C  1697             R1 := R3 SHRL 8;  R0 := #41000001 OR R3 OR R1;  EMIT;
14 0566  07 092C  1698             R1 := STACKP8(R8);  IF R1 < 0 THEN
14 0570  07 092C  1699             BEGIN  COMMENT FIRST DIMENSION;
14 0570  07 092C  1700                R1 := R1 AND #7FFFFFFF;  STACKP8(R8) := R1;
14 0578  07 092C  1701                R1 := R1 SHRL 16;  IF R1 ¬= 6 THEN
14 0584  07 092C  1702                BEGIN  IF R1 = 7 THEN R0 := #4C000002 ++ R10 OR R3 ELSE
14 0594  07 092C  1703                   BEGIN  COMMENT SHIFT;
14 0598  07 092C  1704                      IF R1 <= 2 OR R1 >= 8 THEN R0 := 2 ELSE
14 05AC  07 092C  1705                      IF R1 = 5 THEN R0 := 4 ELSE R0 := 3;
14 05C4  07 092C  1706                   R0 := R0 OR #89000000 OR R3;
14 05CA  07 092C  1707                   END;
14 05CA  07 092C  1708                   EMIT;
14 05D6  07 092C  1709                END;
14 05D6  07 092C  1710             END ELSE
14 05D6  07 092C  1711             BEGIN  R0 := #4C000002 ++ R10 OR R3;  EMIT;
14 05EE  07 092C  1712             END;
14 05EE  07 092C  1713             R0 := #5000000C ++ R10 OR R3;  EMIT;    R1 := SAVER1;
14 0606  07 092C  1714          END;
14 0608  07 092C  1715
14 0608  07 092C  1716        PROCEDURE FILLDESCRIP(R1);
14 0608  07 092C  1717          COMMENT  COMPLETE BOUND PAIR, UPDATE STACK (AR, - ARG2);
14 0608  07 092C  1718          BEGIN  LOGICAL SAVER1;  SAVER1 := R1;
14 060C  07 0930  1719             PUTITIN;   R3 := LSTACKM4(R5);
14 0614  07 0930  1720             R0 := R3 AND #00F00000 OR #49000000 OR DELTALIMIT; EMIT;
14 062E  07 0930  1721             R6 := 1;  RELEASE;  R5 := R5 - 4;
14 0642  07 0930  1722             R0 := #45100000 OR DESCERROR;  EMIT;
14 0656  07 0930  1723             R0 := STACKP4(R8) ++ 12;  R1 := STACKP8(R8);
14 0662  07 0930  1724             R3 := R8 - SSIZE;  STACKP4(R3) := R0;  STACKP8(R3) := R1;
14 0670  07 0930  1725             R7 := R7 OR SIGN;   R1 := SAVER1;
14 0678  07 0930  1726          END;
14 067A  07 0930  1727
14 067A  07 0930  1728        PROCEDURE CLOSEDESCRIP(R1);
14 067A  07 0930  1729          COMMENT  COMPLETE LAST BOUND PAIR, DECLARATION (AR) - ARG2);
14 067A  07 0930  1730          BEGIN  LOGICAL SAVER1; INTEGER DESLENGTH, NARRAYS;
14 067A  07 093C  1731             SAVER1 := R1;   PUTITIN;
14 0682  07 093C  1732             R3 := STACKP8(R8) SHRL 16;   COMMENT SIMPLE TYPE;
14 068A  07 093C  1733             R0 := CLN SHLL 12 OR FP;
14 0696  07 093C  1734             IF R3 = 6 OR R3 = 7 THEN  R0 := R0 OR #58000000  ELSE
```

```
14 06AA  07 093C  1735          BEGIN  R0 := R0 OR #58100000;  EMIT;
14 06BE  07 093C  1736             IF R3 = 3 OR R3 = 5 THEN R3 := 7 ELSE R3 := 3;
14 06CA  07 093C  1737             R0 := #41001000 OR R3;  EMIT;
14 06EC  07 093C  1738             IF R3 = 7 THEN R0 := DUBLMASK ELSE R0 := SINGLMASK;
14 0700  07 093C  1739             R0 := R0 OR #54000000;
14 0704  07 093C  1740          END;
14 0704  07 093C  1741          EMIT;
14 0710  07 093C  1742          R7 := STACKP4(R8) SHRL 16;  R1 := TYPEINFO(R7)*12S + 8;
14 0724  07 093C  1743          DESLENGTH := R1;  R0 := STACKP8(R8) AND #FFFF; NARRAYS := R0;
14 0734  07 093C  1744          R3 := IDLOC1(R7) SHLL 12 + IDLOC2(R7);
14 0740  07 093C  1745          R7 := R3;   R3 := R3 + 4 SHLL 16;
14 074A  07 093C  1746          FOR R10 := 1 STEP 1 UNTIL NARRAYS DO
14 074E  07 093C  1747          BEGIN  COMMENT COPY DESCRIPTOR, STORE AND UPDATE FREE POINTER;
14 0752  07 093C  1748             R0 := #50000000 OR R7;  EMIT;
14 0764  07 093C  1749             R0 := LSTACKM4(R5) AND #00F00000 SHRL 4 OR #1A000000; EMIT;
14 0774  07 093C  1750             R1 := STACKP8(R8) SHRL 16;  IF R1 = 9 THEN
14 0790  07 093C  1751             BEGIN  COMMENT REFERENCE ARRAY, SUPPLY # DIMENSIONS;
14 0790  07 093C  1752                R1 := STACKP4(R8) SHRL 16;
14 0798  07 093C  1753                R0 := TYPEINFO(R1) SHLL 16 OR #92000000 OR R7;  EMIT;
14 07B2  07 093C  1754             END;
14 07B2  07 093C  1755             R7 := R7 + DESLENGTH;
14 07B6  07 093C  1756             R0 := DESLENGTH - 5 SHLL 16 OR #D2000004 ++ R7;  EMIT;
14 07D4  07 093C  1757          END;
14 07E0  07 093C  1758          R0 := INSCOUNTER - 6;  INSCOUNTER := R0;
14 07EC  07 093C  1759          R3 := LSTACKM4(R5);  R6 := 1;  RELEASE;  R5 := R5 - 4;
14 0804  07 093C  1760          R0 := #45400000 OR ALLOCERR;  EMIT;
14 0818  07 093C  1761          R0 := CLN SHLL 12 OR FP OR #50000000;  EMIT;
14 0834  07 093C  1762          R7 := SIGN;    R1 := SAVER1;
14 083C  07 093C  1763       END;
14 083E  07 093C  1764
14 083E  07 093C  1765       PROCEDURE LBOUND(R1);
14 083E  07 093C  1766          COMMENT  PUT L(I) INTO DESCRIPTOR  (:: - ARG1);
14 083E  07 093C  1767          BEGIN  LOGICAL SAVER1, SAVER6;  SAVER1 := R1;  SAVER6 := R6;
14 0846  07 0944  1768             R3 := R8 - SSIZE;  R10 := STACKP4(R3) AND #FFFF + 4;
14 0858  07 0944  1769             R0 := LSTACKM4(R5) AND #00F00000 OR R10 OR #50000000; EMIT;
14 0872  07 0944  1770             R3 := LSTACKM4(R5); R6 := 1; RELEASE;  R5 := R5 - 4;
14 088A  07 0944  1771             R6 := SAVER6;  R1 := SAVER1;
14 0892  07 0944  1772          END;
14 0894  07 0944  1773
14 0894  07 0944  1774          INTEGER SAVER1;    SAVER1 := R1;
14 0898  07 0948  1775          CASE R3 OF
14 0898  07 0948  1776          BEGIN
14 0898  07 0948  1777             INITAREF;  INDEXOP;  FILLDESCRIP;  CLOSEDESCRIP;
14 08BC  07 0948  1778             LBOUND;  INITADCL;
14 08CC  07 0948  1779          END;
14 08E8  07 0948  1780          R1 := SAVER1;
14 08EC  07 0948  1781       END;

                SEGMENT 14  NAME = SEG#14      LENGTH = 09C0  BASE REG = 15

11 0474  07 0948  1782
11 0474  07 0948  1783    PROCEDURE TERMPROCEXIT(R1);  COMMENT THIS PROCEDURE HANDLES
11 0474  07 0948  1784       TERMINAL NODE VARIABLES OR LITERALS FOR TYPED PROCEDURE EXIT;
11 0474  07 0948  1785       BEGIN INTEGER SAVER1;    SAVER1 := R1;    GETTYPE;
11 0484  07 094C  1786          R1 := R7 SHRL 24;
11 048A  07 094C  1787          IF R3 = 6 THEN
11 0492  07 094C  1788          BEGIN COMMENT LOGICAL;
11 0492  07 094C  1789             GETADDRESS;
```

```
11 049E  07 094C  1790          R0 := #43300000 OR R3;    EMIT;    COMMENT VALUE IN R3;
11 04B0  07 094C  1791          R7 := #86000000;  R6 := 6;  SET(TERMNODE);
11 04BC  07 094C  1792       END ELSE IF R3 = 7 THEN
11 04C8  07 094C  1793       BEGIN COMMENT STRING;      GETADDRESS;
11 04D4  07 094C  1794          R0 := #41300000 OR R3;  EMIT; COMMENT ADDRESS OF STRING IN R3;
11 04E6  07 094C  1795          R7 := #87000000;    R6 := 7;        SET(TERMNODE);
11 04F2  07 094C  1796       END ELSE IF R1 =  RCCLID THEN RECORDALLOCATE ELSE
11 050A  07 094C  1797       BEGIN    CONVERT;    LOADREG;         END;
11 0524  07 094C  1798       R1 := SAVER1;
11 0528  07 094C  1799    END;
11 052A  07 094C  1800
11 052A  07 094C  1801     SEGMENT PROCEDURE  PARAMETERS(R1); COMMENT THIS PROCEDURE CALLS THE
11 052A  07 094C  1802       APPROPRIATE PROCEDURE TO HANDLE ACTUAL AND FORMAL PARAMETERS;
11 052A  07 094C  1803       BEGIN
15 0000  07 094C  1804
15 0000  07 094C  1805    PROCEDURE EMITIT(R1);  COMMENT EMITS SAPDS;
15 0000  07 094C  1806       BEGIN INTEGER SAVEINS;   ARRAY  2  INTEGER TEMP;
15 0004  07 0958  1807          STM(R0,R1,TEMP);      MVC(3,SAVEINS,INSCOUNTER);
15 000E  07 0958  1808          R1 := STACKP8(R8) AND MASK;     INSCOUNTER := R1;
15 001A  07 0958  1809          EMIT;
15 0026  07 0958  1810          R1 := STACKP8(R8) + 4;    STACKP8(R8) := R1;
15 0032  07 0958  1811          MVC(3,INSCOUNTER,SAVEINS);    LM(R0,R1,TEMP);
15 003C  07 0958  1812       END;
15 003E  07 0958  1813
15 003E  07 0958  1814    PROCEDURE SUBRENTRY(R1); COMMENT THIS PROCEDURE EMITS THE CODE FOR
15 003E  07 0958  1815       IMPLICIT SUBROUTINE ENTRY;
15 003E  07 0958  1816       BEGIN INTEGER SAVER1;      SAVER1 := R1;
15 0042  07 095C  1817          R0 := #58200000 OR MP;      EMIT;
15 0056  07 095C  1818          CALLALLOCATE;
15 0062  07 095C  1819          R0 := #1B330000;  EMIT;    R0 := #1B440000;  EMIT;
15 0082  07 095C  1820          R0 := #90046000;      EMIT;
15 0092  07 095C  1821          R0 := #50600000 OR MP;    EMIT;
15 00A6  07 095C  1822          R3 := CLN - 1;    CLN := R3;     R1 := 12 - R3 SHLA 2 + DSP;
15 00C0  07 095C  1823          R10 := R8 - SSIZE;    R10 := STACK(R10) AND MASK - 4;
15 00D2  07 095C  1824          R0 := TREE(R10) SHRL 24;
15 00DA  07 095C  1825          IF R0 = STACKADDR THEN R1 := TREE(R10) AND MASK;
15 00EA  07 095C  1826          R3 := R3 SHLL 12 OR R1;   NEXTADDR := R3;
15 00F4  07 095C  1827          R1 := CLN;    IF R1 = 10 THEN
15 0100  07 095C  1828          BEGIN R0 := #58B05000 OR DSP;    EMIT;    END ELSE
15 0114  07 095C  1829          IF R1 < 10 THEN
15 0120  07 095C  1830          BEGIN R0 := R1 + 1 SHLL 20 OR #980B5000 OR DSP;  EMIT;  END;
15 013E  07 095C  1831          R0 := CLN SHLL 20 OR #18060000;    EMIT;
15 0156  07 095C  1832          R1 := CLN;      IF R1 = 11 THEN
15 0162  07 095C  1833          BEGIN R0 := #50B06000 OR DSP;    EMIT;    END ELSE
15 0176  07 095C  1834          IF R1 < 11 THEN
15 0182  07 095C  1835          BEGIN R0 := R1 SHLL 20 OR #900B6000 OR DSP;    EMIT;    END;
15 019C  07 095C  1836          CHAININ; R1 := SAVER1;
15 01AC  07 095C  1837       END;
15 01AE  07 095C  1838
15 01AE  07 095C  1839    PROCEDURE SUBREXIT(R1); COMMENT THIS PROCEDURE EMITS THE CODE FOR
15 01AE  07 095C  1840       IMPLICIT SUBROUTINE EXIT;
15 01AE  07 095C  1841       BEGIN INTEGER SAVER1;      SAVER1 := R1;
15 01B2  07 0960  1842          R1 := NEXTADDR AND MASK SHRL 12;  R0 := NEXTADDR AND #FFF;
15 01C6  07 0960  1843          IF R1 ¬= CLN OR R0 >= #FF9 THEN BEGIN R3 := 5;  ERROR; END;
15 01E6  07 0960  1844          R0 := CLN SHLL 12 OR #98120000 OR RETA;    EMIT;
15 0202  07 0960  1845          R0 := #50200000 OR MP;      EMIT;
15 0216  07 0960  1846          R0 := #07F10000;      EMIT;
15 0226  07 0960  1847          COMMENT FIXUP LA INSTRUCTION IN SUBRENTRY;
```

```
15 0226  07 0960  1848          R1 := LOGPOINTER;    R3 := LOGSTACK(R1) + 2;
15 0232  07 0960  1849          R1 := R1 - 4;      LOGPOINTER := R1;
15 023A  07 0960  1850          R10 := NEXTADDR AND #FFF OR #6000 + 7 SHRL 3 SHLL 3;
15 0252  07 0960  1851          PROGRAM(R3) := R10;
15 0256  07 0960  1852          R1 := SAVER1;
15 025A  07 0960  1853       END;
15 025C  07 0960  1854
15 025C  07 0960  1855    PROCEDURE PROCEDURECALL(R1); COMMENT THIS PROCEDURE SETS UP
15 025C  07 0960  1856       PROCEDURE CALL CODE, COUNTS THE NUMBER OF ACTUAL PARAMETERS ('1),
15 025C  07 0960  1857       AND SETS UP POINTERS TO BASE OF THE SAPDS AND IMPLICIT SUBRS;
15 025C  07 0960  1858       BEGIN INTEGER SAVER1;    SAVER1 := R1;
15 0260  07 0964  1859          MARKRECORDS;    DUMPALLGENREG;    DUMPALLFLREG;
15 0284  07 0964  1860          COMMENT GET NUMBER OF ACTUAL PARAMETERS;
15 0284  07 0964  1861          R1 := 1;    R3 := R8;
15 028A  07 0964  1862       L1: R10 := STACK(R3) AND MASK;
15 0292  07 0964  1863          R10 := TREE(R10) SHLL 1 SHRL 25;
15 029E  07 0964  1864          IF R10 = APRPAREN THEN GOTO L2;
15 02A6  07 0964  1865          R1 := R1 + 1;    R3 := R3 - SSIZE;    GOTO L1;
15 02B2  07 0964  1866       L2: R3 := R1;  PROCCALLCODE;
15 02C0  07 0964  1867          COMMENT SAVE NEXTADDR (TO BE RESET AFTER SUBRS COMPILED)
15 02C0  07 0964  1868             AND INSCOUNTER OF BRANCH AROUND SUBRS;
15 02CC  07 0964  1869          R1 := LOGPOINTER + 4;    R0 := NEXTADDR;    LOGSTACK(R1) := R0;
15 02DC  07 0964  1870          R1 := R1 + 4;    R0 := INSCOUNTER;      LOGSTACK(R1) := R0;
15 02DC  07 0964  1871          LOGPOINTER := R1;    R1 := 0;    IC(R1,SIMPLETYPE(R7));
15 02E8  07 0964  1872          IF R1 = 7 THEN R0 := SIMTYPEINFO(R7) SHLL 16 + INSCOUNTER;
15 02FC  07 0964  1873          R1 := R1 SHLL 24 + R0;  R0 := CLN;
15 0306  07 0964  1874          IF R0 = 12 THEN R1 := R1-4 ELSE R1 := R1-8; STACKP4(R8) := R1;
15 031E  07 0964  1875          R0 := #47F0E000;            EMIT;
15 032E  07 0964  1876          R0 := CLN; IF R0 = 12 THEN
15 033A  07 0964  1877          BEGIN COMMENT  PAD TO CORRECT SAPD DISPLACEMENT;
15 033A  07 0964  1878             R0 := #47000000; EMIT;
15 034A  07 0964  1879          END;
15 034A  07 0964  1880          R1 := R3 SHLA 3;    R10 := INSCOUNTER;
15 0354  07 0964  1881          R1 := R1 + R10;        INSCOUNTER := R1;
15 035A  07 0964  1882          R1 := R1 SHLL 16 + R10;    STACKP8(R8) := R1;
15 0364  07 0964  1883          R8 := R8 + SSIZE;      SUBRENTRY;      R8 := R8 - SSIZE;
15 0370  07 0964  1884          R1 := SAVER1;
15 0374  07 0964  1885       END;
15 0376  07 0964  1886
15 0376  07 0964  1887    PROCEDURE ACPARAM(R1); COMMENT THIS PROCEDURE BUILDS A SAPD AND
15 0376  07 0964  1888       IMPLICIT SUBROUTINE (IF THERE IS TO BE ONE) FOR A PROCEDURE CALL;
15 0376  07 0964  1889       BEGIN INTEGER SAVER1;          SAVER1 := R1;
15 037A  07 0968  1890          R3 := CLN + 1;    CLN := R3;
15 0386  07 0968  1891          IF R7 = NULLST THEN R7 := SIGN;
15 0392  07 0968  1892          IF R7 >= 0 THEN
15 0398  07 0968  1893          BEGIN COMMENT ACTUAL PARAMETER IS ID OR CONSTANT.
15 0398  07 0968  1894                   NO IMPLICIT SUBROUTINE FOR ID;
15 0398  07 0968  1895             R7 := R7 AND MASK2;
15 039C  07 0968  1896             R1 := CLN;  IF R1 <= 6 THEN BEGIN R3 := 4;  ERROR;  END;
15 03B8  07 0968  1897             IF R1 = 12 THEN R1 := INSCOUNTER - NUMBYTESUBR
15 03C4  07 0968  1898             ELSE R1 := INSCOUNTER - NUMBYTESUBR - 4; INSCOUNTER := R1;
15 03DC  07 0968  1899             R1 := CHAIN(0);
15 03E0  07 0968  1900             WHILE R1 >= INSCOUNTER DO R1 := PROGRAM(R1+2);
15 03F0  07 0968  1901             CHAIN(0) := R1;
15 03F4  07 0968  1902             R1 := LOGPOINTER - 4;        LOGPOINTER := R1;
15 04C0  07 0968  1903             GETTYPE;      R0 := #A0000000 OR R3;
15 0412  07 0968  1904             IF R3 = 7 THEN
15 041A  07 0968  1905             BEGIN R1 := SIMTYPEINFO(R7) SHLL 8 OR R0; R0 := R1; END;
```

```
15 0426   07 0968   1906          R1 := R7 SHRL 24;
15 042C   07 0968   1907          IF R1 = NUMBER OR R1 = BIT OR R1 = STRYNG OR R1 = TRUE OR
15 044C   07 0968   1908             R1 = FALSE OR R1 = NULLL THEN
15 045C   07 0968   1909          BEGIN COMMENT LITERAL;
15 045C   07 0968   1910             IF R3 = 7 THEN
15 0464   07 0968   1911             BEGIN COMMENT GET STRING LENGTH FROM CONSPTTAB;
15 0464   07 0968   1912               R10 := R7 AND MASK + CPTBASE;
15 046E   07 0968   1913               R0 := CONSPTTAB(R10) AND OPCODEMASK SHRL 16 OR #A2000007;
15 047A   07 0968   1914             END ELSE
15 047E   07 0968   1915             R0 := R0 OR #02000000;    EMITIT;
15 048A   07 0968   1916             R1 := STACKP4(R8) AND MASK;
15 0492   07 0968   1917             R0 := STACKP8(R8) SHRL 16 - R1 OR #41301000;  EMITIT;
15 04A4   07 0968   1918             R0 := #58200000 OR MP;  EMIT;
15 04B8   07 0968   1919             R1 := CLN; IF R1 <= 11 THEN
15 04C4   07 0968   1920             BEGIN  IF = THEN R0 := #58B05000 ELSE
15 04CC   07 0968   1921                              R0 := R1 SHLL 20 OR #980B5000;
15 04DA   07 0968   1922                R0 := R0 OR DSP; EMIT;
15 04EA   07 0968   1923             END;
15 04EA   07 0968   1924             CHAININ;   R10 := R7 AND MASK + CPTBASE;
15 05C0   07 0968   1925             R1 := R7 SHRL 24;
15 05C6   07 0968   1926             IF R1 = NULLL THEN R0 := #41300000 OR NULLREF ELSE
15 0516   07 0968   1927             R0 := CONSPTTAB(R10) AND MASK OR #4130E000;      EMIT;
15 0532   07 0968   1928             R0 := #07F10000;    EMIT;
15 0542   07 0968   1929             R1 := INSCOUNTER SHLL 16;
15 054A   07 0968   1930             R3 := STACKP8(R8) AND MASK OR R1;  STACKP8(R8) := R3;
15 0558   07 0968   1931          END ELSE
15 0558   07 0968   1932          BEGIN COMMENT ID. CHECK WHETHER FORMAL OR ACTUAL, AND IF
15 055C   07 0968   1933                FORMAL WHETHER VALUE AND/OR RESULT;
15 055C   07 0968   1934             R1 := 0;    IC(R1,TYPE(R7));
15 0564   07 0968   1935          IF R1 = 16 OR R1 = 19 THEN
15 0574   07 0968   1936          BEGIN COMMENT FORMAL NAME PARAMETER;
15 0574   07 0968   1937             R1 := R7 SHRL 24;
15 057A   07 0968   1938             IF R1 = FUNCID THEN R0 := R0 OR #00400000 ELSE
15 0586   07 0968   1939             IF R1 = ARRAYID THEN
15 0592   07 0968   1940             BEGIN R1 := TYPEINFO(R7) SHLL 16; R0 := R0 OR R1; END;
15 059C   07 0968   1941             EMITIT;    R1 := IDLOC2(R7);
15 05A4   07 0968   1942             R0 := IDLOC1(R7) SHLL 12 OR R1 OR #98340000;   EMITIT;
15 05B6   07 0968   1943          END ELSE
15 05B6   07 0968   1944          BEGIN COMMENT ACTUAL OR FORMAL BUT VALUE/RESULT. CHECK
15 05BA   07 0968   1945                WHETHER PROCEDURE OR SOME KIND OF VARIABLE;
15 05BA   07 0968   1946             R1 := R7 SHRL 24;
15 05C0   07 0968   1947             IF R1 = FUNCID THEN
15 05C8   07 0968   1948             BEGIN COMMENT PROCEDURE;
15 05C8   07 0968   1949                R0 := R0 OR #02400000;    EMITIT;
15 05D0   07 0968   1950                R1 := STACKP4(R8) AND MASK;
15 05D8   07 0968   1951                R0 := STACKP8(R8) SHRL 16 - R1 OR #41301000; EMITIT;
15 05EA   07 0968   1952                R0 := #07000000; EMIT;  COMMENT MARKER FOR CHECK;
15 05FA   07 0968   1953                R0 := #18430000; EMIT;  COMMENT SET R4 = R3 + 4096;
15 060A   07 0968   1954                R0 := #5A400000 OR ADDRSTEP;  EMIT;
15 061E   07 0968   1955                R0 := IDLOC2(R7) AND #FF; CHAINOUT;
15 0632   07 0968   1956                R0 := R0 OR #58400000; EMIT; R0 := #18F40000; EMIT;
15 0652   07 0968   1957                R0 := IDLOC2(R7) SHRL 8 AND #FF - CLN + 1 SHLA 2 +
15 066A   07 0968   1958                    DSP  OR #58505000;      EMIT;
15 067E   07 0968   1959                R0 := #07FF0000;      EMIT;
15 068E   07 0968   1960                R1 := INSCOUNTER SHLL 16;
15 0696   07 0968   1961                R3 := STACKP8(R8) AND MASK OR R1;  STACKP8(R8) := R3;
15 06A0   07 0968   1962             END ELSE
15 06A4   07 0968   1963             IF R1 = RCCLID THEN
```

```
15 06B0  C7 0968   1964                BEGIN COMMENT EMIT FULL SUBR - CALL RECORD ALLOCATE;
15 06B0  07 0963   1965                   R0 := #A2000009; EMITIT; R1 := STACKP4(R8) AND MASK;
15 06C0  C7 0968   1966                   R0 := STACKP8(R8) SHRL 16 - R1 OR #41301000; EMITIT;
15 06D2  07 0963   1967                   SUBRENTRY;    MARKRECORDS;    R3 := 0;
15 06E6  07 0968   1968                   IC(R3,TYPEINFO(R7+1));  R3:=R3 SHLL 4 ++ RECTABLE;
15 06F2  07 0968   1969                   R0 := #41300000 OR R3;      EMIT;
15 C704  C7 0968   1970                   R0 := @RECALLOCATE; R0 := NEG R0; EMITCALL;
15 0716  07 0968   1971                   R3 := NEXTADDR + 3 AND #FFFFFFFC;
15 C722  07 0968   1972                   R0 := #50300000 OR R3;    EMIT;
15 0734  07 0968   1973                   R0 := #41300000 OR R3;    EMIT;
15 0746  C7 0968   1974                   R3 := R3 + 4;    NEXTADDR := R3;    SUBREXIT;
15 0752  07 0968   1975                   R1 := CLN + 1;    CLN := R1;
15 C75E  07 0968   1976                END ELSE
15 075E  07 0968   1977                BEGIN  COMMENT VARIABLE, CON ID, OR ARRAY ID;
15 0762  07 0968   1978                   IF R1 = CONID THEN R0 := R0 OR #01000000 ELSE
15 C76E  07 0968   1979                   IF R1 = ARRAYID THEN
15 077A  07 0968   1980                   BEGIN  R1 := TYPEINFO(R7) SHLL 16; R0 := R0 OR R1;
15 0784  C7 0968   1981                   END;
15 0784  07 0968   1982                   EMITIT;
15 0788  C7 0968   1983                   R0 := IDLOC1(R7) SHLL 12 + IDLOC2(R7) OR #41300000;
15 C798  C7 0968   1984                   EMITIT;
15 079C  07 0963   1985                END;
15 C79C  07 0968   1986             END;
15 C79C  C7 0968   1987          END;
15 079C  07 0968   1988       END ELSE
15 079C  07 0968   1989       BEGIN COMMENT NONTERMINAL;
15 07A0  07 0968   1990          R1 := CLN;  IF R1 <= 7 THEN BEGIN R3 := 4;  EROR;  END;
15 07BC  07 0968   1991          R1 := R7 SHLL 1 SHRL 25;
15 07C6  07 0968   1992          IF R1 = 0 THEN
15 07CC  07 0968   1993          BEGIN COMMENT STATEMENT;
15 07CC  C7 0968   1994             R0 := #A2400000;      EMITIT;
15 07D4  07 0968   1995          END ELSE
15 07D4  07 0968   1996          BEGIN COMMENT EXPRESSION;
15 07D8  C7 0968   1997             R10 := LSTACKM4(R5);
15 07DC  C7 0968   1998             IF R1 = 6 THEN
15 C7E4  07 0968   1999             BEGIN COMMENT LOGICAL - CHECK IF ARRAY ELEMENT ;
15 C7E4  07 0968   2000                R3 := R7 SHLL 8 SHRL 24;
15 C7EE  C7 0968   2001                IF R3 = 0 THEN R10 := R10 OR SIGN;
15 07F8  07 0968   2002             END;
15 C7F8  07 0968   2003             IF R10 >= 0 THEN
15 07FE  C7 0968   2004             BEGIN COMMENT COMPUTED ADDRESS - INCLUDES
15 07FE  07 0968   2005                SUBSCRIPTED VARIABLES AND FIELD DESIGNATORS;
15 C7FE  07 0968   2006             R0 := R7 AND 1;    IF R0 = 1 THEN
15 080C  C7 0968   2007             BEGIN COMMENT STRING PROCEDURE (WITH PARAMS) AS ACTUAL
15 080C  C7 0968   2008                PARAM - MOVE RESULT INTO LOCAL STACK AT POINT OF CALL;
15 080C  07 0968   2009                R0 := R7 AND TYPEMASK SHRL 8 OR R1 OR #A2000000; EMITIT;
15 081C  C7 0968   2010                R1 := INSCOUNTER - 10;    INSCOUNTER := R1;
15 C82C  07 0968   2011                R10 := LOGPOINTER;  R1 := LOGSTACK(R10-8);
15 0834  07 0968   2012                R0 := R7 AND TYPEMASK OR #D2000000 OR R1;
15 0840  07 0968   2013                R3 := #30000000;    EMIT;
15 C850  07 0968   2014                R0 := R0 AND MASK OR #41300000;    EMIT;
15 0864  C7 0968   2015                R1 := R7 AND TYPEMASK SHRL 16 + LOGSTACK(R10-8) + 1;
15 0876  07 0968   2016                LOGSTACK(R10-8) := R1;
15 C87A  C7 0968   2017             END ELSE
15 087A  07 0968   2018             BEGIN COMMENT DONT MOVE;
15 087E  07 0968   2019             R0 := R1 OR #A3000000;
15 0884  C7 0968   2020             IF R1 = 7 THEN BEGIN R1 := R7 AND TYPEMASK SHRL 8 OR R0;
15 0898  07 0968   2021                                  R0 := R1;      END;
```

```
15 089A  07 0968  2022              EMITIT;    R0 := R10 AND #FFF;  IF R0 ¬= 0 THEN
15 08AA  07 0968  2023              BEGIN R0 := R10 AND MASK OR #41300000;  EMIT;  END ELSE
15 08C0  07 0968  2024              BEGIN
15 08C4  07 0968  2025                 R0 := R10 AND #F000 SHLL 4 OR #18300000;
15 08D2  07 0968  2026                 IF R0 ¬= #18330000 THEN EMIT;
15 08E6  07 0968  2027              END;       END;
15 08E6  07 0968  2028              END ELSE
15 08E6  07 0968  2029              BEGIN COMMENT RESULT IN REGISTER GIVEN BY TOP
15 08EA  07 0968  2030                 LSTACK ENTRY - STORE RESULT IN LOCAL STACK AND   LA  ;
15 08EA  07 0968  2031                 TEST(SUBARFLAG);  IF = THEN
15 08F2  07 0968  2032                 BEGIN COMMENT SUBARRAY;    RESET(SUBARFLAG);
15 08F6  07 0968  2033                    R0 := LSTACKM4(R5) AND #FFFFF OR #A2000000; EMITIT;
15 0906  07 0968  2034                    R3 := LSTACKM4(R5) AND #0F000000 SHRL 4 OR SIGN;
15 0916  07 0968  2035                    LSTACKM4(R5):=R3; R6:=1; RELEASE; R5:=R5-4; GOTO L1;
15 0932  07 0968  2036                 END;
15 0932  07 0968  2037                 R0 := R1 OR #A2000000;    EMITIT;
15 093C  07 0968  2038                 R3 := R7 SHLL 1 SHRL 25;
15 0946  07 0968  2039                 IF R3 >= 8 THEN R3 := 1;
15 0952  07 0968  2040                 STORERESULT;  IF R3 = 6 THEN GOTO L1;
15 0966  07 0968  2041              END;
15 0966  07 0968  2042           L:  R3:=LSTACKM4(R5); R6:=R7 SHLL 1 SHRL 25; RELEASE; R5:=R5-4;
15 0980  07 0968  2043        L1: END;
15 0984  07 0968  2044            RESETRECORD;
15 0990  07 0968  2045            R1 := STACKP4(R8) AND MASK;
15 0998  07 0968  2046            R0 := STACKP8(R8) SHRL 16 - R1 OR #41301000;     EMITIT;
15 09AA  07 0968  2047            R1 := CLN-1;  CLN := R1;  SUBREXIT;  R1 := CLN+1; CLN := R1;
15 09C6  07 0968  2048            R1 := INSCOUNTER SHLL 16;
15 09CE  07 0968  2049            R3 := STACKP8(R8) AND MASK OR R1;  STACKP8(R8) := R3;
15 09DC  07 0968  2050         END;
15 09DC  07 0968  2051            SUBRENTRY;    R7 := SIGN;
15 09E4  07 0968  2052            R1 := SAVER1;
15 09E8  07 0968  2053         END;
15 09EA  07 0968  2054
15 09EA  07 0968  2055      PROCEDURE LASTPARAM(R1); COMMENT THIS PROCEDURE CALLS ACPARAM FOR LAST
15 09EA  07 0968  2056         ACTUAL PARAMETER AND FINISHES THE PROCEDURE CALL;
15 09EA  07 0968  2057         BEGIN INTEGER SAVER1;      SAVER1 := R1;
15 09EE  07 096C  2058            ACPARAM;
15 09F2  07 096C  2059            R1 := CLN + 1;  CLN := R1;
15 09FE  07 096C  2060            IF R1 = 12 THEN R1 := INSCOUNTER - NUMBYTESUBR
15 0A0A  07 096C  2061            ELSE R1 := INSCOUNTER - NUMBYTESUBR - 4;  INSCOUNTER := R1;
15 0A22  07 096C  2062            R1 := CHAIN(0); WHILE R1 >= INSCOUNTER DO R1 := PROGRAM(R1+2);
15 0A36  07 096C  2063            CHAIN(0) := R1;
15 0A3A  07 096C  2064            COMMENT FIXUP BRANCH AROUND SAPDS-SUBRS AND RESTORE NEXTADDR;
15 0A3A  07 096C  2065            R1 := LOGPOINTER - 4;  R3 := LOGSTACK(R1) + 2;
15 0A4A  07 096C  2066            R10 := INSCOUNTER OR #E000;    PROGRAM(R3) := R10;
15 0A56  07 096C  2067            R1 := R1 - 4;    R0 := LOGSTACK(R1);
15 0A5E  07 096C  2068            NEXTADDR := R0;    R1 := R1 - 4;  LOGPOINTER := R1;
15 0A6A  07 096C  2069            COMMENT PUT PROCEDURE SIMPLETYPE IN UPPER BYTE OF R7;
15 0A6A  07 096C  2070            R3 := STACKP4(R8) AND OPCODEMASK;    R7 := R3 OR SIGN;
15 0A78  07 096C  2071            IF R3 ¬= 0 THEN
15 0A7E  07 096C  2072            BEGIN COMMENT TYPED PROCEDURE. MARK APPROPRIATE REGISTER USED -
15 0A7E  07 096C  2073               PROCEDURE RETURNS RESULT IN THAT REGISTER;
15 0A7E  07 096C  2074               R3 := R3 SHRL 24;      IF R3 >= 6 THEN R3 := 1;
15 0A8E  07 096C  2075               R0 := FLAG;  R( 4):= R0;    R0 := 5 - R3 SHLL 30;
15 0AA0  07 096C  2076               CASE R3 OF
15 0AA0  07 096C  2077               BEGIN GENREG;  FLREG;  FLREG;  PRFLREG;  PRFLREG;   END;
15 0B0C  07 096C  2078               R0 := 0;   R( 4):= R0;
15 0B14  07 096C  2079               R3 := STACKP4(R8) SHRL 24;  IF R3 <= 5 THEN CONVERTRESULT
```

```
15 0B24  07 096C   2080            ELSE IF R3 = 6 THEN
15 0B3A  07 096C   2081            BEGIN COMMENT LOGICAL;
15 0B3A  07 096C   2082               R3 := LSTACKM4(R5) SHLL 1 SHRL 9;  LSTACKM4(R5) := R3;
15 0B4A  07 096C   2083               R7 := R7 OR #10000;
15 0B4E  07 096C   2084            END ELSE IF R3 = 7 THEN
15 0B5A  07 096C   2085            BEGIN COMMENT STRING;
15 0B5A  07 096C   2086               R3 := LSTACKM4(R5) SHLL 1 SHRL 9;  LSTACKM4(R5) := R3;
15 0B6A  07 096C   2087               R0 := STACKP4(R8) AND TYPEMASK OR NEXTADDR OR #D2000000;
15 0B7A  07 096C   2088               R3 := #30000000;    EMIT;
15 0B8A  07 096C   2089               R10 := STACKP4(R8) AND TYPEMASK SHRL 16 + 1 + NEXTADDR;
15 0B9E  07 096C   2090               R0:=LSTACKM4(R5) SHLL 8 AND MASK5 OR #41000000 OR NEXTADDR;
15 0BAE  07 096C   2091               NEXTADDR := R10;      EMIT;
15 0BC2  07 096C   2092               R1 := STACKP4(R8) AND TYPEMASK OR R7 + 1;  R7 := R1;
15 0BD2  07 096C   2093            END ELSE IF R3 = 9 THEN
15 0BDE  07 096C   2094            BEGIN COMMENT REFERENCE;
15 0BDE  07 096C   2095               R5 := R5 + 4;    SETRECORD:    R5 := R5 - 4;
15 0BE6  07 096C   2096            END;
15 0BE6  07 096C   2097         END;
15 0BE6  07 096C   2098         R1 := SAVER1;
15 0BEA  07 096C   2099      END;
15 0BEC  07 096C   2100
15 0BEC  07 096C   2101      INTEGER SAVER1;      SAVER1 := R1;
15 0BFC  07 0970   2102      CASE R3 OF
15 0BF0  07 0970   2103      BEGIN   PROCEDURECALL;  ACPARAM;  LASTPARAM;  SUBREXIT;
15 0C14  07 0970   2104      END;
15 0C28  07 0970   2105      R1 := SAVER1;
15 0C2C  07 0970   2106   END;

                  SEGMENT 15  NAME = SEG#15      LENGTH = 0D50  BASE REG = 15

11 052A  07 0970   2107
11 052A  07 0970   2108   SEGMENT PROCEDURE  ENTRYEXIT(R1); COMMENT THIS PROCEDURE CONTAINS EN
11 052A  07 0970   2109      EXIT PROCEDURES FOR PROCEDURES AND BLOCKS.  THE APPROPRIATE
11 052A  07 0970   2110      PROCEDURE IS SELECTED BY THE PARAMETER R3;
11 052A  07 0970   2111      BEGIN INTEGER DPD, SAVE4, LIMT, HOLD4;
16 0000  07 0980   2112
16 0000  07 0980   2113   PROCEDURE CALLCONV(R1);  COMMENT THIS PROCEDURE SETS UP THE PARAMETERS
16 0000  07 0980   2114      FOR AMD CALLS THE CONVERT ROUTINE TO CONVERT VALUE AND RESULT
16 0000  07 0980   2115      PARAMETERS.  RA = 0,1 : RESULT,VALUE    ;
16 0000  07 0980   2116      BEGIN INTEGER SAVER1;      SAVER1 := R1;
16 0008  07 0984   2117         R0 := R3 SHLL 8 OR #41000000;      EMIT;
16 001E  07 0984   2118         R0 := CLN SHLL 12 OR DPD OR #43000000 + 4;     EMIT;
16 003E  07 0984   2119         IF R10 = 0 THEN BEGIN COMMENT RESULT - NEGATE R0;
16 0044  07 0984   2120               R0 := #11000000;    EMIT;      END;
16 0054  07 0984   2121         R0 := CLN SHLL 12 + IDLOC2(R4) OR #41200000;    EMIT;
16 0070  07 0984   2122         R0 := @FPARCONV; R0 := NEG R0; EMITCALL;
16 0082  07 0984   2123         R1 := SAVER1;
16 0086  07 0984   2124      END;
16 0088  07 0984   2125
16 0088  07 0984   2126
16 0088  07 0984   2127   SEGMENT PROCEDURE  PROCEDUREENTRY(R1); COMMENT THIS PROCEDURE EMITS THE
16 0088  07 0984   2128      FOR PROCEDURE ENTRY;
16 0088  07 0984   2129      BEGIN INTEGER SAVER1;      SAVER1 := R1;
17 0004  07 0988   2130         COMMENT PLACE SIMPLETYPE OF PROCEDURE, BASE AND LENGTH OF
17 0004  07 0988   2131            FPARS IN NAMETABLE, AND WHETHER OR NOT THERE ARE RESULT
17 0004  07 0988   2132            PARAMETERS IN STACK;
17 0004  07 0988   2133         R1 := 0;    IC(R1,SIMPLETYPE(R7));    STACKP4(R8) := R1;
17 0010  07 0988   2134         IF R1 = 7 THEN
```

```
17 0018  C7 0988  2135          BEGIN COMMENT STRING PROCEDURE;
17 C018  C7 0988  2136             R0 := SIMTYPEINFO(R7) SHLL 16 OR R1;
17 0022  07 0988  2137             STACKP4(R8) := R0;
17 0026  07 0988  2138          END;
17 0026  07 0988  2139          R1 := TYPEINFO(R7) SHLA 2;
17 002E  C7 0988  2140          IF R1 ¬= 0 THEN
17 0034  07 0988  2141          BEGIN COMMENT THERE ARE FPARS;
17 0034  07 0988  2142             R1 := BLOCKLIST(R1);    R3 := R1 AND MASK;
17 003E  07 0988  2143             R10 := R1 SHRL 16 - 12 + R3 SHLL 16 OR R3;
17 0050  07 0988  2144             STACKP8(R8) := R10;
17 0054  07 0988  2145          END ELSE
17 0054  07 0988  2146          BEGIN COMMENT NO FPARS;
17 0058  C7 0988  2147             R0 := 0;    STACKP8(R8) := R0;
17 C060  C7 0988  2148          END;
17 0060  07 0988  2149          COMMENT SET CLN (CURRENT LEVEL NUMBER), LOGSEG (CURRENT
17 0060  C7 0988  2150             PROGRAM SEGMENT NUMBER), AND NEXTADDR (LOCAL STACK ORIGIN)  ;
17 0060  07 0988  2151          R3 := R7 AND MASK;      R3 := NAMETABLE(R3);
17 006A  07 0988  2152          R0 := R3 AND #FF; LOGSEG := R0; IF R0 = 1 THEN
17 007C  C7 0988  2153          BEGIN  R10 := R10-R10; STACKP8(R8) := R10;
17 0082  07 0988  2154          END;
17 C082  07 0988  2155          SEGNO(0) := R0; R0 := 0; SEGTABIDX := R0;
17 008E  C7 0988  2156          R0 := ENDCHAIN; CHAIN(0) := R0;
17 0096  07 0988  2157          RESET(PRINT); MVI(0,TRACEIT); CLI(0,TRACE); IF ¬= THEN
17 00A6  07 0988  2158          BEGIN  R10 := LOGSEG; LM(R0,R1,TRACEBITS);
17 00AE  07 0988  2159             SLDL(R0,B10); IF R0 < 0 THEN
17 00B8  C7 0988  2160             BEGIN  COMMENT SET TRACE FLAG FOR SEGMENT;
17 00B8  C7 0988  2161                R0 := R0-R0; IC(R0,TRACE); STC(R0,TRACEIT);
17 00C2  C7 0988  2162                IF R0 = 2 OR R0 = 5 OR R0 = 8 THEN SET(PRINT);
17 00DE  07 0988  2163             END;
17 00DE  C7 0988  2164          END;
17 00DE  07 0988  2165          R10 := R3 AND #FF00 SHRL 8;         CLN := R10;
17 00EC  07 0988  2166          R10 := R10 SHLL 12;
17 00F0  07 0988  2167          R1 := R3 SHRL 16 OR R10 + 7 AND _8; BASELOC := R1;
17 0104  07 0988  2168          R1 := R1 + 8; NEXTADDR := R1;
17 010C  07 0988  2169          R14 := SAVE14;
17 0110  07 0988  2170          R0 := 0;  INSCOUNTER := R0;  NUMSTACKP := R0;
17 011C  07 0988  2171          NUMSTACK := R0;    REFPCOUNT := R0;
17 0124  07 0988  2172          FOR R1 := 0 STEP 2 UNTIL 200 DO    PROGRAM(R1) := R0;
17 013C  07 0988  2173          FOR R1 := 0 STEP 4 UNTIL 96 DO NUMSTACK(R1) := R0;
17 0154  07 0988  2174          R0 := LOADMARK OR #F008; EMIT;  R0 := #47F00000; EMIT;
17 0178  07 0988  2175          R0 := 0; WRITENUMBER;  R0 := 4096; WRITENUMBER;
17 0198  07 0988  2176          R0 := 0; WRITENUMBER;
17 01A8  07 0988  2177          COMMENT EMIT SFPD'S;
17 C1A8  C7 0988  2178          R3 := STACKP8(R8) SHRL 16;    LIMT := R3;
17 01B4  07 0988  2179          R3 := STACKP8(R8) AND MASK;
17 01BC  07 0988  2180          IF R3 = 0 THEN
17 01C2  07 0988  2181          BEGIN COMMENT NO FPARS;
17 01C2  07 0988  2182             R0 := 0;    WRITENUMBER;
17 01D2  07 0988  2183          END ELSE
17 01D2  07 0988  2184          BEGIN
17 01D6  07 0988  2185             R0 := 0;  R1 := TYPEINFO(R7) SHLL 2;
17 01E2  07 0988  2186             R1 := BLOCKLIST(R1) SHRL 16 / 12;  R0 := R1;  WRITENUMBER;
17 01FC  07 0988  2187          FOR R10 := R3 STEP 12 UNTIL LIMT DO
17 01FE  07 0988  2188          BEGIN          R1 := 0;
17 02C6  C7 0988  2189             IC(R1,SIMPLETYPE(R10));
17 02CA  07 0988  2190             BEGIN
17 02CA  07 0988  2191                R0 := R1;      IF R1 = 7 THEN
17 0214  07 0988  2192                BEGIN COMMENT STRING;
```

```
17 0214  07 0988  2193                R1 := SIMTYPEINFO(R10) SHLL 8 OR R0;  R0 := R1;
17 0220  07 0988  2194             END;
17 0220  07 0988  2195          END;
17 0220  07 0988  2196          R1 := 0;    IC(R1,TYPE(R10));
17 0228  07 0988  2197          IF R1 = 19 THEN R0 := R0 OR #00400000 ELSE
17 0234  07 0988  2198          IF R1 = 18 THEN
17 0240  07 0988  2199          BEGIN COMMENT ARRAY;
17 0240  07 0988  2200             R1 := TYPEINFO(R10) SHLL 16 OR R0;    R0 := R1;
17 024C  07 0988  2201          END;
17 024C  07 0988  2202          IC(R1,VR(R10));  R1 := R1 SHLL 24 OR R0;  R0 := R1;
17 0258  07 0988  2203          WRITENUMBER;
17 0264  07 0988  2204       END;      END;
17 0270  07 0988  2205       COMMENT LEAVE ROOM FOR LABEL TABLE, UPDATE CONSTANT POINTER
17 0270  07 0988  2206          TABLE, AND OUTPUT CONSTANT TABLE;
17 0270  07 0988  2207       R0 := LITORG;  INSCOUNTER := R0;
17 0278  07 0988  2208       R10 := CONSPTTABL - 4 + CPTBASE;
17 0284  07 0988  2209       FOR R1 := CPTBASE STEP 4 UNTIL R10 DO
17 0288  07 0988  2210       BEGIN
17 028C  07 0988  2211          R3 := CONSPTTAB(R1) + R0;
17 0292  07 0988  2212          CONSPTTAB(R1) := R3;
17 0296  07 0988  2213       END;
17 02A0  07 0988  2214       R10 := CONSTABL + 3 SHRL 2 SHLL 2 - 4;
17 02B4  07 0988  2215       FOR R3 := 0 STEP 4 UNTIL R10 DO
17 02B8  07 0988  2216       BEGIN
17 02BC  07 0988  2217          R1 := CONSTAB + R3;    R0 := B1;    WRITENUMBER;
17 02D2  07 0988  2218       END;
17 02DC  07 0988  2219       R0 := INSCOUNTER + 3 SHRL 2 SHLL 2;      INSCOUNTER := R0;
17 02F0  07 0988  2220       R0 := R0 OR #E000;  PROGRAM(6) := R0;
17 02F8  07 0988  2221       R0 := #58200000 OR MP;    EMIT;
17 030C  07 0988  2222       CALLALLOCATE;
17 0318  07 0988  2223       R1 := INSCOUNTER;    LAADD := R1;
17 0320  07 0988  2224       R0 := #41300000;  EMIT;    R0 := #41400000;    EMIT;
17 0340  07 0988  2225       R0 := #90046000;    EMIT;
17 0350  07 0988  2226       R0 := #50600000 OR MP;    EMIT;
17 0364  07 0988  2227       R3 := STACKP8(R8) SHRL 16;    LIMT := R3;
17 0370  07 0988  2228       R3 := STACKP8(R8) AND MASK;
17 0378  07 0988  2229       IF R3 ¬= 0 THEN
17 037E  07 0988  2230       FOR R10 := R3 STEP 12 UNTIL LIMT DO
17 0380  07 0988  2231       BEGIN        R0 := 0;    R1 := 0;
17 038C  07 0988  2232          IC(R1,SIMPLETYPE(R10));  IC(R0,VR(R10));
17 0394  07 0988  2233          IF R1 = 9 AND R0 >= 1 THEN
17 03A4  07 0988  2234          BEGIN COMMENT REF VALUE/RESULT PARAM;
17 03A4  07 0988  2235             R1 := REFPCOUNT + 1;    REFPCOUNT := R1;
17 03B0  07 0988  2236             IF R1 = 1 THEN
17 03B8  07 0988  2237             BEGIN COMMENT FIRST ONE;
17 03B8  07 0988  2238                R1 := LAADD;  R0 := IDLOC2(R10);  PROGRAM(R1+2) := R0;
17 03C4  07 0988  2239                R0 := #58300000 OR NULLREF;  EMIT;
17 03D8  07 0988  2240                R0 := #6000 + IDLOC2(R10) OR #50300000;  EMIT;
17 03F0  07 0988  2241             END;
17 03F0  07 0988  2242          END;
17 03F0  07 0988  2243       END;
17 03FC  07 0988  2244       R0 := REFPCOUNT; IF R0 ¬= 0 THEN
17 0406  07 0988  2245       BEGIN  COMMENT AT LEAST ONE REFERENCE VALUE/RESULT PARAMETER;
17 0406  07 0988  2246          IF R0 > 1 THEN
17 040E  07 0988  2247          BEGIN
17 040E  07 0988  2248             R1 := LAADD;  R1 := PROGRAM(R1+2);  R3 := #6000 OR R1;
17 041C  07 0988  2249             R0 := R0 SHLL 2 - 5 SHLL 16 OR R3 + 4 OR #D2000000;
17 0432  07 0988  2250             R3 := R3 SHLL 16;  EMIT;  R0 := REFPCOUNT;
```

```
17 0446  07 0988  2251          END;
17 0446  07 0988  2252          R0 := R0 SHLL 16 OR REFVAR+1 OR #92006000;  EMIT;
17 0462  07 0988  2253       END;
17 0462  07 0988  2254       COMMENT SAPD - DPD OPERATIONS;
17 0462  07 0988  2255       R1 := TYPEINFO(R7) SHLA 2;
17 046A  07 0988  2256       IF R1 ¬= 0 THEN
17 0470  07 0988  2257       BEGIN COMMENT THERE ARE FPARS;
17 0470  07 0988  2258          R0:=0; R1 := BLOCKLIST(R1) SHRL 16 / 12; R10 := R1 SHLA 3 - 8;
17 0486  07 0988  2259          SAVE4 := R4;    R3 := 12 - CLN SHLA 2 + DSP;    DPD := R3;
17 04A2  07 0988  2260          FOR R4 := 0 STEP 8 UNTIL R10 DO
17 04A6  07 0988  2261          BEGIN  R0 := #18420000;      EMIT;
17 04BA  07 0988  2262             R0 := #44001000 + R4 + 16;      EMIT;
17 04D0  07 0988  2263             R0 := #90346000 + DPD + R4;    EMIT;
17 04E6  07 0988  2264             R3 := #1000 + R4 + 12 SHLL 16;
17 04F4  07 0988  2265             R0 := #D6C06000 + DPD + R4;    EMIT;
17 050A  07 0988  2266             R3 := #1000 + R4 + 15 SHLL 16;
17 0518  07 0988  2267             R0 := #D2006000 + DPD + R4 + 4;      EMIT;
17 0532  07 0988  2268          END;
17 053C  07 0988  2269          R4 := SAVE4;
17 0540  07 0988  2270       END;
17 0540  07 0988  2271       R1 := CLN;
17 0544  07 0988  2272       IF R1 = 10 THEN
17 054C  07 0988  2273       BEGIN R0 := #58B05000 OR DSP;    EMIT;    END ELSE
17 0560  07 0988  2274       IF R1 < 10 THEN
17 056C  07 0988  2275       BEGIN R0 := R1 + 1 SHLL 20 OR #980B5000 OR DSP;    EMIT;    END;
17 058A  07 0988  2276       R0 := CLN SHLL 20 OR #18060000;      EMIT;
17 05A2  07 0988  2277       R1 := CLN;
17 05A6  07 0988  2278       IF R1 = 11 THEN
17 05AE  07 0988  2279       BEGIN R0 := #50B06000 OR DSP;      EMIT;    END ELSE
17 05C2  07 0988  2280       IF R1 < 11 THEN
17 05CE  07 0988  2281       BEGIN R0 := R1 SHLL 20 OR #900B6000 OR DSP;    EMIT;    END;
17 05E8  07 0988  2282       R0 := INSCOUNTER; BASESTLOC := R0;
17 05F0  07 0988  2283       R0 := STOREMARK OR BASELOC; EMIT;
17 0604  07 0988  2284       COMMENT DPD - PV OPERATIONS;
17 06C4  07 0988  2285       R3 := STACKP8(R8) SHRL 16;    LIMT := R3;
17 0610  07 0988  2286       R3 := STACKP8(R8) AND MASK;    SAVE4 := R4;
17 061C  07 0988  2287       R4 := 6;    CNTP := R4;
17 0624  07 0988  2288       IF R3 ¬= 0 THEN
17 062A  07 0988  2289       FOR R4 := R3 STEP 12 UNTIL LIMT DO
17 062C  07 0988  2290       BEGIN
17 0630  07 0988  2291          R3 := CNTP + 8;  CNTP := R3;
17 063C  07 0988  2292          R3 := TYPEINFO(R4) SHRL 8;    IF R3 = 3 THEN R3 := 1;
17 0650  07 0988  2293          IF R3 = 1 THEN
17 0658  07 0988  2294          BEGIN  COMMENT VALUE;  HOLD4 := R4; R4 := SAVE4;
17 0660  07 0988  2295             R0 := #FF;  R3 := CLN SHLL 12 OR DPD;  FPARCODE;
17 067C  07 0988  2296             SAVE4 := R4;  R4 := HOLD4;
17 0684  07 0988  2297             R3 := 0;  IC(R3,SIMPLETYPE(R4));  IF R3 = 1 THEN R3 := 8;
17 0698  07 0988  2298             IF R3 <= 5 THEN
17 06A0  07 0988  2299             BEGIN COMMENT CALL CONVERT; R10 := 1;  CALLCONV;  END ELSE
17 06B0  07 0988  2300             BEGIN  R3 := R3 - 5;
17 06B8  07 0988  2301             CASE R3 OF
17 06B8  07 0988  2302             BEGIN
17 06B8  07 0988  2303                R0 := #D2000000;
17 06C4  07 0988  2304                BEGIN
17 06C8  07 0988  2305                   R1 := IDLOC2(R4);  R10 := CLN SHLL 12 OR R1;
17 06D6  07 0988  2306                   R3 := SIMTYPEINFO(R4);  R1 := TYPEINFO(R4) SHRL 8;
17 06E2  07 0988  2307                   IF R1 = 3 THEN
17 06EA  07 0988  2308                   BEGIN COMMENT VALUE/RESULT - NO BLANKING NEEDED;
```

```
17 06EA  07 0988  2309                        R0 := R3 SHLL 16 OR #D2000000;  GOTO L2;
17 06F8  07 0988  2310                     END;
17 06F8  07 0988  2311                     IF R3 = 0 THEN GOTO L1;
17 06FE  07 0988  2312                     R0 := #92400000 OR R10;     EMIT;
17 0710  07 0988  2313                     IF R3 = 1 THEN
17 0718  07 0988  2314                     BEGIN  R0 := R10 + 1 OR #92400000;   EMIT;
17 072E  07 0988  2315                     END ELSE
17 072E  07 0988  2316                     BEGIN
17 0732  07 0988  2317                        R0 := R3 - 1 SHLL 16 OR R10 + 1 OR #D2000000;
17 0746  07 0988  2318                        R3 := R10 SHLL 16;     EMIT;
17 0758  07 0988  2319                     END;
17 0758  07 0988  2320                  L1: R0 := CLN SHLL 12 OR RETA OR #58200000;  EMIT;
17 0774  07 0988  2321                     R0 := #43202000 OR CNTP;     EMIT;
17 0788  07 0988  2322                     R0 := INSCOUNTER + 10 OR #47F0E000;  EMIT;
17 07A0  07 0988  2323                     R0 := #D2000000 OR R10;  R3 := #30000000;   EMIT;
17 07B6  07 0988  2324                     R0 := INSCOUNTER - 6 OR #4420E000;    EMIT;
17 07CE  07 0988  2325                     GOTO L;
17 07D2  07 0988  2326                   END;
17 07D2  07 0988  2327                   R0 := #D2030000;       R0 := #D2030000;
17 07E2  07 0988  2328                END;        L2:
17 07F6  07 0988  2329                R10 := IDLOC2(R4);  R1 := CLN SHLL 12 OR R10;
17 0804  07 0988  2330                R0 := R0 OR R1;  R3 := #30000000;      EMIT;
17 0816  07 0988  2331             L: END;
17 0816  07 0988  2332             END;
17 0816  07 0988  2333             R3 := TYPEINFO(R4) SHRL 8;  IF R3 = 3 THEN R3 := 2;
17 082A  07 0988  2334             IF R3 = 2 THEN
17 0832  07 0988  2335             BEGIN COMMENT RESULT - PLACE INFO INTO STACK;
17 0832  07 0988  2336                R0 := STACKP4(R8) OR SIGN;  STACKP4(R8) := R0;
17 083E  07 0988  2337             END;
17 083E  07 0988  2338             R1 := 0;  IC(R1,TYPE(R4));
17 0846  07 0988  2339             IF R1 = 18 THEN
17 084E  07 0988  2340             BEGIN COMMENT FPAR ARRAY - COPY DESCRIPTOR;
17 084E  07 0988  2341                R0 := #FF; R3 := CLN SHLL 12 OR DPD;  HOLD4 := R4;
17 0862  07 0988  2342                R4 := SAVE4;  FPARCODE;  SAVE4 := R4; R4 := HOLD4;
17 087A  07 0988  2343                R1 := TYPEINFO(R4) * 12;
17 0882  07 0988  2344                R1 := R1 + 3 SHLL 16 OR #D2000000;    R10 := IDLOC2(R4);
17 0892  07 0988  2345                R0 := CLN SHLL 12 OR R10 OR R1;
17 089E  07 0988  2346                R3 := #30000000;      EMIT;
17 08AE  07 0988  2347             END;
17 08AE  07 0988  2348             R1 := DPD + 8;    DPD := R1;
17 08BA  07 0988  2349          END;
17 08C6  07 0988  2350          R4 := SAVE4;    R1 := SAVER1;    RESET(TERMNODE);
17 08D2  07 0988  2351       END;

                             SEGMENT 17  NAME = SEG#17      LENGTH = 0988  BASE REG = 15

16 0088  07 0988  2352
16 0088  07 0988  2353    PROCEDURE PROCEXIT(R1);  COMMENT THIS PROCEDURE EMITS CODE FOR
16 0088  07 0988  2354       PROCEDURE EXIT;
16 0088  07 0988  2355       BEGIN INTEGER SAVER1;     SAVER1 := R1;
16 008C  07 098C  2356          COMMENT DETERMINE IF TYPED PROCEDURE. IF SO PLACE RESULT OR
16 008C  07 098C  2357             ADDRESS OF RESULT IN APPROPRIATE REGISTER DEPENDING ON
16 008C  07 098C  2358             WHETHER PROCEDURE HAS FPARS OR NOT;
16 008C  07 098C  2359          R1 := STACKP4(R8) AND MASK;
16 0094  07 098C  2360          IF R1 ¬= 0 THEN
16 009A  07 098C  2361          BEGIN COMMENT TYPED PROCEDURE;
16 009A  07 098C  2362             R6 := R1;  IF R7 >= 0 THEN TERMPROCEXIT ELSE
16 00AE  07 098C  2363                              IF R6 <= 5 OR R6 >= 8 THEN LOADREG;
```

```
16 00CE  07 098C  2364          R1 := STACKP8(R8);  R6 := R6 AND MASK;
16 00D6  07 098C  2365          IF R6 = 6 THEN R1 := 0 ELSE IF R6 = 7 THEN R1 := 1;
16 00F2  07 098C  2366          IF R1 ¬= 0 THEN
16 00F8  07 098C  2367          BEGIN COMMENT PROCEDURE HAS FPARS;
16 00F8  07 098C  2368             R3:=R6; IF R3>=8 THEN R3:=1; R10:=LSTACKM4(R5);
16 010A  07 098C  2369             CASE R3 OF
16 010A  07 098C  2370             BEGIN
16 010A  07 098C  2371                BEGIN R0 := R10 AND MASK5 SHRL 4 OR #18300000;
16 0120  07 098C  2372                      IF R0 ¬= #18330000 THEN EMIT;
16 0134  07 098C  2373                END;
16 0134  07 098C  2374                BEGIN R0 := R10 AND MASK5 SHRL 4 OR #38000000;
16 0146  07 098C  2375                   IF R0 ¬= #38000000 THEN EMIT;  R0 := 0;
16 015E  07 098C  2376                END;
16 015E  07 098C  2377                BEGIN R0 := R10 AND MASK5 SHRL 4 OR #28000000;
16 0170  07 098C  2378                   IF R0 ¬= #28000000 THEN EMIT;  R0 := 0;
16 0188  07 098C  2379                END;
16 0188  07 098C  2380                R0 := #38000000;
16 0190  07 098C  2381                R0 := #28000000;
16 0198  07 098C  2382                BEGIN COMMENT LOGICAL;
16 019C  07 098C  2383                END;
16 019C  07 098C  2384                BEGIN COMMENT STRING;
16 01A0  07 098C  2385                   IF ¬TERMNODE THEN
16 01A8  07 098C  2386                   BEGIN COMMENT NONTERMINAL;
16 01A8  07 098C  2387                      R0 := #41300000 OR LSTACKM4(R5);   EMIT;
16 01BC  07 098C  2388                   END;
16 01BC  07 098C  2389                   R10 := STACK(R8) AND MASK;
16 01C4  07 098C  2390                   R10 := TREE(R10) AND TYPEMASK;  COMMENT EXPR LENGTH;
16 01CC  07 098C  2391                   R3 := STACKP4(R8) AND TYPEMASK; COMMENT PROC LENGTH;
16 01D4  07 098C  2392                   IF R10 ¬= R3 THEN
16 01DA  07 098C  2393                   BEGIN COMMENT MOVE EXPR TO LOCAL STACK, THEN BLANK
16 01DA  07 098C  2394                      UP TO PROC LENGTH;
16 01DA  07 098C  2395                      R0 := R10 OR #D2000000 OR NEXTADDR;
16 01E4  07 098C  2396                      R3 := #30000000;       EMIT;
16 01F4  07 098C  2397                      COMMENT NOW BLANK ;
16 01F4  07 098C  2398                      R1 := STACKP4(R8) AND TYPEMASK SHRL 16;
16 0200  07 098C  2399                      R0:=R10 SHRL 16;R3:=R1-R0-1; COMMENT # TO BLANK-1;
16 020E  07 098C  2400                      IF R3 < 0 THEN GOTO L2;
16 0214  07 098C  2401                      R0 := R0 + NEXTADDR + 1 OR #92400000;   EMIT;
16 022C  07 098C  2402                      IF R3 = 0 THEN GOTO L2 ELSE IF R3 = 1 THEN
16 023A  07 098C  2403                      BEGIN COMMENT BLANK 1 MORE;  R0 := R0 + 1; EMIT;
16 024A  07 098C  2404                      END ELSE
16 024A  07 098C  2405                      BEGIN
16 024E  07 098C  2406                         R1 := R0 AND MASK + 1 OR #D2000000;
16 025C  07 098C  2407                         R0 := R3 - 1 SHLL 16 OR R1;
16 0268  07 098C  2408                         R3 := R0 - 1 SHLL 16;    EMIT;
16 027E  07 098C  2409                      END;
16 027E  07 098C  2410                   L2: R0 := #41300000 OR NEXTADDR;    EMIT;
16 0292  07 098C  2411                      R1:=STACKP4(R8) AND TYPEMASK SHRL 16+NEXTADDR+1;
16 02A6  07 098C  2412                      NEXTADDR := R1;
16 02AA  07 098C  2413                   END;
16 02AA  07 098C  2414                   IF TERMNODE THEN GOTO L1;
16 02B2  07 098C  2415                END;
16 02B2  07 098C  2416             END;
16 02D2  07 098C  2417             IF R0 = #38000000 OR R0 = #28000000 THEN FORCECOMPLEX;
16 02EE  07 098C  2418             R3 := LSTACKM4(R5);      RELEASE;      R5 := R5 - 4;
16 0302  07 098C  2419             RESETRECORD;
16 030E  07 098C  2420          L1: END ELSE
16 030E  07 098C  2421              BEGIN COMMENT PROCEDURE HAS NO FPARS;
```

```
16 0312  07 098C  2422              R3 := R6;  IF R3 >= 8 THEN R3 := 1;
16 0320  07 098C  2423              STORERESULT;    IF R6 = 6 THEN GOTO L;
16 0334  07 098C  2424              R3 := LSTACKM4(R5);    RELEASE;    R5 := R5 - 4;
16 0348  07 098C  2425              RESETRECORD;
16 0354  07 098C  2426           L: END;
16 0354  07 098C  2427           END ELSE
16 0354  07 098C  2428           BEGIN  R1 := R7 SHRL 24;  IF R1 = FUNCID THEN
16 0366  07 098C  2429              BEGIN COMMENT PROPER PROCEDURE WHICH CALLS A PROPER PROCEDURE;
16 0366  07 098C  2430                 DUMPALLGENREG;  DUMPALLFLREG;  R3:=R3-R3; PROCCALLCODE;
16 038C  07 098C  2431                 R7 := SIGN;
16 0390  07 098C  2432              END;
16 0390  07 098C  2433           END;
16 0390  07 098C  2434           COMMENT SEARCH FOR AND STORE ALL RESULT PARAMETERS;
16 0390  07 098C  2435           R1 := STACKP4(R8);
16 0394  07 098C  2436           IF R1 < 0 THEN
16 039A  07 098C  2437           BEGIN COMMENT THERE ARE RESULT FPARS;
16 039A  07 098C  2438              R1 := STACKP4(R8) AND MASK; IF R1 ¬= 0 THEN
16 03A8  07 098C  2439              BEGIN
16 03A8  07 098C  2440                 IF R1 = 1 OR R1 >= 6 THEN
16 03B8  07 098C  2441                 BEGIN
16 03B8  07 098C  2442                    R3 := NEXTADDR + 3 AND #FFFFFFFC; R1 := R3 + 4;
16 03CA  07 098C  2443                    NEXTADDR := R1; R0 := #00300000; R6 := 1; ASSEMBLE;
16 03E2  07 098C  2444                 END ELSE IF R1 <= 3 THEN
16 03EE  07 098C  2445                 BEGIN
16 03EE  07 098C  2446                    R3 := NEXTADDR + 7 AND #FFFFFFF8; R1 := R3 + 8;
16 04C0  07 098C  2447                    NEXTADDR := R1; R0 := 0; R6 := 3; ASSEMBLE;
16 0418  07 098C  2448                 END ELSE
16 0418  07 098C  2449                 BEGIN
16 041C  07 098C  2450                    R3 := NEXTADDR + 7 AND #FFFFFFF8; R1 := R3 + 16;
16 042E  07 098C  2451                    NEXTADDR := R1; R0 := #00020000; R6:=5; ASSEMBLE;
16 0446  07 098C  2452                 END;
16 0446  07 098C  2453           END;
16 0446  07 098C  2454           R3 := 12 - CLN SHLA 2 + DSP;    DPD := R3;
16 045A  07 098C  2455           R3 := STACKP8(R8) SHRL 16;    LIMT := R3;
16 0466  07 098C  2456           R3 := STACKP8(R8) AND MASK;    SAVE4 := R4;
16 0472  07 098C  2457           R4 := 6;    CNTP := R4;
16 047A  07 098C  2458           FOR R4 := R3 STEP 12 UNTIL LIMT DO
16 047C  07 098C  2459           BEGIN
16 0480  07 098C  2460              R3 := CNTP + 8;    CNTP := R3;
16 048C  07 098C  2461              R3 := TYPEINFO(R4) SHRL 8;    IF R3 = 3 THEN R3 := 2;
16 04A0  07 098C  2462              IF R3 = 2 THEN
16 04A8  07 098C  2463              BEGIN  R0 := R0-R0;  R3 := CLN SHLL 12 OR DPD;
16 04B6  07 098C  2464                 HOLD4 := R4;    R4 := SAVE4;    FPARCODE;
16 04CA  07 098C  2465                 SAVE4 := R4;    R4 := HOLD4;
16 04D2  07 098C  2466                 R3 := 0;    IC(R3,SIMPLETYPE(R4));
16 04DA  07 098C  2467                 IF R3 <= 5 THEN
16 04E2  07 098C  2468                 BEGIN COMMENT CALL CONVERT;  R10 := 0;  CALLCONV;  END
16 04EA  07 098C  2469                 ELSE BEGIN  R3 := R3 - 5;  R10 := 0;
16 04F6  07 098C  2470                 CASE R3 OF
16 04F6  07 098C  2471                 BEGIN
16 04F6  07 098C  2472                    R0 := #D2003000;
16 0502  07 098C  2473                    BEGIN COMMENT STRING;
16 0506  07 098C  2474                       R3 := TYPEINFO(R4) SHRL 8;  IF R3 = 3 THEN GOTO L;
16 0516  07 098C  2475                       R0 := #1B220000;    EMIT;
16 0526  07 098C  2476                       R0 := CLN SHLL 12 OR RETA OR #58100000;    EMIT;
16 0542  07 098C  2477                       R0 := #43201000 OR CNTP;  EMIT;
16 0556  07 098C  2478                       R1:=CPTBASE;R0:=CONSPTTAB(R1) AND MASK OR
16 0562  07 098C  2479                      #5B20E000;EMIT;R0:=INSCOUNTER+22 OR #4740E000;EMIT;
```

```
16 057E  07 098C  2480                        R0 := #92403000;     EMIT;
16 059A  07 098C  2481                        R0 := INSCOUNTER + 10 OR #47F0E000;     EMIT;
16 05B2  07 098C  2482                        R0 := #D2003001;  R3 := #30000000;     EMIT;
16 05C6  07 098C  2483                        R0 := INSCOUNTER - 6 OR #4420E000;     EMIT;
16 05DE  07 098C  2484                     L: R0 := SIMTYPEINFO(R4) SHLL 16 OR #D2003000;
16 05EA  07 098C  2485                     END;
16 05EA  07 098C  2486                     R0 := #D2033000;  BEGIN R0 := #D2033000; R10:=6; END;
16 05FE  07 098C  2487                  END;
16 0612  07 098C  2488                  R1 := IDLOC2(R4);
16 0616  07 098C  2489                  R3 := CLN SHLL 12 OR R1 SHLL 16;     EMIT;
16 0630  07 098C  2490                  IF R10 = 6 THEN REFSTOR;
16 0644  07 098C  2491                  END;
16 0644  07 098C  2492               END;
16 0644  07 098C  2493               R1 := DPD + 8;  DPD := R1;
16 0650  07 098C  2494            END;
16 065C  07 098C  2495            R4 := SAVE4;
16 0660  07 098C  2496            R1 := STACKP4(R8) AND MASK; IF R1 ¬= 0 THEN
16 066E  07 098C  2497            BEGIN
16 066E  07 098C  2498               IF R1 = 1 OR R1 >= 6 THEN
16 067E  07 098C  2499               BEGIN
16 067E  07 098C  2500                  R3 := NEXTADDR - 4; R0 := #08300000; R6 := 1; ASSEMBLE;
16 069A  07 098C  2501               END ELSE IF R1 <= 3 THEN
16 06A6  07 098C  2502               BEGIN
16 06A6  07 098C  2503                  R3 := NEXTADDR - 8; R0 := #08000000; R6 := 3; ASSEMBLE;
16 06C2  07 098C  2504               END ELSE
16 06C2  07 098C  2505               BEGIN
16 06C6  07 098C  2506                  R3 := NEXTADDR - 16; R0 := #08020000; R6 := 5; ASSEMBLE;
16 06D6  07 098C  2507               END;
16 06E2  07 098C  2508            END;
16 06E2  07 098C  2509         END;
16 06E2  07 098C  2510         COMMENT EMIT PROCEDURE CLOSE CODE;
16 06E2  07 098C  2511         R3 := 4;  PARAMETERS;
16 06F0  07 098C  2512         SAVE4 := R4;
16 06F4  07 098C  2513         R4 := INSCOUNTER + 3 AND _4;  IF R4 ¬= INSCOUNTER THEN
16 07C8  07 098C  2514         BEGIN R0 := 0; EMIT;
16 0718  07 098C  2515         END;
16 0718  07 098C  2516         R4 := #5840;  COMMENT USED IN SUBR FOR FORMAL PROCEDURE;
16 071C  07 098C  2517         FOR R3 := 4 STEP 4 UNTIL SEGTABIDX DO
16 0720  07 098C  2518         BEGIN  R10 := INSCOUNTER;   R0 := 0; WRITENUMBER;
16 0738  07 098C  2519            R1 := CHAIN(R3);  CHAIN(R3) := R10;
16 0740  07 098C  2520            WHILE R1 ¬= ENDCHAIN DO
16 0748  07 098C  2521            BEGIN  R0 := PROGRAM(R1+2);
16 074C  07 098C  2522               IF R4 ¬= PROGRAM(R1) THEN
16 0754  07 098C  2523                  R10 := R10 OR #E000 ELSE R10 := R10 - R1 ++ #3008;
16 0762  07 098C  2524               PROGRAM(R1+2) := R10;  R10 := CHAIN(R3);  R1 := R0;
16 076C  07 098C  2525            END;
16 0770  07 098C  2526         END;
16 077C  07 098C  2527         R10 := INSCOUNTER;
16 0780  07 098C  2528         IF R10 > 4096 THEN R4 := #98EF ELSE R4 := #58E0;
16 0794  07 098C  2529         R10 := BASELOC;  R1 := CHAIN(0);
16 079C  07 098C  2530         WHILE R1 ¬= ENDCHAIN DO
16 07A4  07 098C  2531         BEGIN  R0 := PROGRAM(R1+2);
16 07A8  07 098C  2532            PROGRAM(R1) := R4; PROGRAM(R1+2) := R10; R1 := R0;
16 07B2  07 098C  2533         END;
16 07B6  07 098C  2534         PROGRAM(0) := R4; R0 := 8; CHAIN(0) := R0;
16 07C2  07 098C  2535         IF R4 = #98EF THEN R4 := #90EF ELSE R4 := #50E0;
16 07D6  07 098C  2536         R1 := BASESTLOC; PROGRAM(R1) := R4;
16 07DE  07 098C  2537         R0 := INSCOUNTER; PROGRAM(18) := R0;
```

```
16 C7E6  07 098C  2538          R4 := SAVE4;  R1 := SAVER1;
16 07EE  07 098C  2539        END;
16 07F0  07 098C  2540
16 C7F0  07 098C  2541    PROCEDURE BLOCKCALLNENTRY(R1);
16 07F0  07 098C  2542       BEGIN INTEGER SAVER1;
16 C7F0  07 0990  2543
16 07F0  07 0990  2544          PROCEDURE LINKREF(R1); COMMENT SEARCH NAMETABLE FOR REFERENCES -
16 07F0  07 0990  2545             COUNT AND PLACE IN PROG SEG WITH STARTING ADDRESS;
16 07F0  C7 0990  2546             BEGIN INTEGER SAVERZ,BLENG;      SAVERZ := R1;
16 C7F8  07 0998  2547                R3 := R7 AND #FFF000 SHRL 10;     R3 := BLOCKLIST(R3);
16 0806  C7 0998  2548                R1 := R3 AND MASK;
16 C80C  07 0998  2549                R0 := R3 SHRL 16 + R1 - 12;      BLENG := R0;
16 081C  07 0998  2550                FOR R3 := R1 STEP 12 UNTIL BLENG DO
16 C81E  07 0998  2551                BEGIN    R1 := 0;
16 0826  07 0998  2552                   IC(R1,SIMPLETYPE(R3));
16 082A  07 0998  2553                   IF R1 = 9 THEN
16 C832  C7 0998  2554                   BEGIN COMMENT REFERENCE;
16 0832  07 0998  2555                      IC(R1,TYPE(R3));
16 0836  07 0998  2556                      IF R1 = 2 THEN
16 083E  C7 0998  2557                      BEGIN COMMENT REF ARRAY;
16 C83E  C7 0998  2558                         R10 := AREFCOUNT + 1;     AREFCOUNT := R10;
16 C84A  07 0998  2559                         IF R10 = 1 THEN
16 0852  07 0998  2560                         BEGIN COMMENT FIRST ONE - PUT ADDRESS IN PROG SEG;
16 0852  C7 0998  2561                            R1 := IDLOC2(R3);    AREFADD := R1;
16 C85A  07 0998  2562                         END;
16 C85A  07 0998  2563                      END ELSE IF R1 = 0 THEN
16 0864  07 0998  2564                      BEGIN COMMENT SIMPLE REF;
16 C864  07 0998  2565                         R0 := REFCOUNT + 1;    REFCOUNT := R0;
16 0870  07 0998  2566                         IF R0 = 1 THEN
16 0878  07 0998  2567                         BEGIN COMMENT FIRST ONE - PUT ADDRESS IN PROG SEG;
16 C878  07 0998  2568                            R1 := IDLOC2(R3);    REFADD := R1;
16 0880  07 0998  2569                         END;
16 0880  07 0998  2570                      END;
16 0880  07 0998  2571                   END;
16 0880  07 0998  2572                END;
16 088C  07 0998  2573                R0 := REFCOUNT;  R1 := AREFCOUNT;  STACKP4(R8) := R0;
16 C898  07 0998  2574                STACKP8(R8) := R1;
16 089C  C7 0998  2575                R1 := SAVERZ;
16 C8A0  07 0998  2576             END;
16 08A2  07 0998  2577
16 08A2  07 0998  2578          SAVER1 := R1;  R1 := 0;  REFCOUNT := R1;  AREFCOUNT := R1;
16 08B2  07 0998  2579          R1:=R7 AND MASK8; IF R1=0 THEN SET(RESFLAG )ELSE RESET(RESFLAG);
16 C8CA  C7 0998  2580          LINKREF;
16 08CE  07 0998  2581          R10 := R8 - SSIZE;
16 C8D4  07 0998  2582          TEST(RESFLAG);    IF = THEN
16 08DC  07 0998  2583          BEGIN COMMENT DATA SEGMENT FOR THIS BLOCK MERGED WITH DATA
16 08DC  07 0998  2584             SEGMENT FOR PROCEDURE HEADING;
16 C8DC  07 0998  2585             R1 := MASK1;       STACKP4(R10) := R1;
16 C8E4  07 0998  2586             R1 := REFPCOUNT;  R0 := REFCOUNT;  IF R1 = 0 AND R0 ¬= 0 THEN
16 C8F8  07 0998  2587             BEGIN COMMENT LOCAL REFS AND NO REF PARAMS - FIXUP LA 3     ;
16 C8F8  C7 0998  2588                R1 := LAADD + 2;  R0 := REFADD;    PROGRAM(R1) := R0;
16 0908  07 0998  2589             END;
16 09C8  07 0998  2590             R1 := AREFCOUNT;    IF R1 ¬= 0 THEN
16 0912  07 0998  2591             BEGIN COMMENT REF ARRAYS - FIXUP LA 4     ;
16 0912  07 0998  2592                R1 := LAADD + 6;    R0 := AREFADD;    PROGRAM(R1) := R0;
16 0922  07 0998  2593             END;
16 0922  07 0998  2594             GOTO MERGE;
16 0926  07 0998  2595          END;
```

```
16 0926  07 0998  2596          R1 := 0;  R10 := R8 - SSIZE;  STACKP4(R10) := R1;
16 0934  07 0998  2597          R1 := LOGPOINTER + 4;      LOGPOINTER := R1;
16 0940  07 0998  2598          R0 := NEXTADDR;       LOGSTACK(R1) := R0;
16 0948  07 0998  2599          R3 := CLN - 1;       CLN := R3;
16 0954  07 0998  2600          R1 := R7 AND MASK8;  R3 := R3 SHLL 12 OR R1 AND MASK;
16 0964  07 0998  2601          NEXTADDR := R3;
16 0968  07 0998  2602          R0 := CLN + 1 SHLL 16 OR #18200000;       EMIT;
16 0984  07 0998  2603          CALLALLOCATE;    R1 := REFCOUNT;
16 0994  07 0998  2604          IF R1 = 0 THEN R0 := #1B330000 ELSE R0 := #41300000 OR REFADD;
16 09AA  07 0998  2605          EMIT;       R1 := AREFCOUNT;
16 09BA  07 0998  2606          IF R1 = 0 THEN R0 := #1B440000 ELSE R0 := #41400000 OR AREFADD;
16 09D0  07 0998  2607          EMIT;       R0 := #90046000;    EMIT;
16 09EC  07 0998  2608          R0 := #50600000 OR MP;       EMIT;
16 0A00  07 0998  2609          R0 := CLN SHLL 20 OR #18060000;       EMIT;
16 0A18  07 0998  2610          R1 := CLN;
16 0A1C  07 0998  2611          IF R1 = 11 THEN
16 0A24  07 0998  2612          BEGIN R0 := #50B06000 OR DSP;    EMIT;    END ELSE
16 0A38  07 0998  2613          IF R1 < 11 THEN
16 0A44  07 0998  2614          BEGIN R0 := R1 SHLL 20 OR #900B6000 OR DSP;    EMIT;    END;
16 0A5E  07 0998  2615       MERGE:
16 0A5E  07 0998  2616          R10 := REFCOUNT;  IF R10 > 0 THEN
16 0A68  07 0998  2617          BEGIN  R0 := #58300000 OR NULLREF;  EMIT;
16 0A7C  07 0998  2618             R3 := CLN SHLL 12 OR REFADD;  R0 := #50300000 OR R3; EMIT;
16 0A9A  07 0998  2619             IF R10 > 1 THEN
16 0AA2  07 0998  2620             BEGIN  R0 := R10 SHLL 2 - 5 SHLL 16 OR R3 + 4 OR #D2000000;
16 0ABA  07 0998  2621                R3 := R3 SHLL 16;  EMIT;
16 0ACA  07 0998  2622             END;
16 0ACA  07 0998  2623             IF RESFLAG THEN R10 := R10 + REFPCOUNT;  R10 := R10 SHLL 16;
16 0ADA  07 0998  2624             R0 := CLN SHLL 12 OR REFVAR+1 OR R10 OR #92000000; EMIT;
16 0AFC  07 0998  2625          END;
16 0AFC  07 0998  2626          R1 := SAVER1;
16 0B00  07 0998  2627       END;
16 0B02  07 0998  2628
16 0B02  07 0998  2629    PROCEDURE BLOCKEXIT(R1);
16 0B02  07 0998  2630       BEGIN INTEGER SAVER1;       SAVER1 := R1;
16 0B06  07 099C  2631          R1 := STACKP4(R8);     IF R1 < 0 THEN GOTO MERGE;
16 0B10  07 099C  2632          R0 := CLN SHLL 12 OR #58100000 OR DL;     EMIT;
16 0B2C  07 099C  2633          R0 := #50100000 OR MP;     EMIT;
16 0B40  07 099C  2634          COMMENT RESET CURRENT LEVEL NUMBER AND FIXUP LA INSTRUCTION
16 0B40  07 099C  2635             IN BLOCK ENTRY;
16 0B40  07 099C  2636          R1 := CLN + 1;       CLN := R1;
16 0B4C  07 099C  2637          R1 := LOGPOINTER;       R3 := LOGSTACK(R1) + 2;      R1 := R1 - 4;
16 0B5C  07 099C  2638          R10 := NEXTADDR AND #FFF OR #6000 + 7 SHRL 3 SHLL 3;
16 0B74  07 099C  2639          PROGRAM(R3) := R10;
16 0B78  07 099C  2640          R3 := LOGSTACK(R1);        NEXTADDR := R3;
16 0B80  07 099C  2641          R1 := R1 - 4;       LOGPOINTER := R1;
16 0B88  07 099C  2642       MERGE:
16 0B88  07 099C  2643          R1 := SAVER1;
16 0B8C  07 099C  2644       END;
16 0B8E  07 099C  2645
16 0B8E  07 099C  2646    PROCEDURE LABELDEC(R1);  COMMENT BUILD BRANCH INSTRUCTION IN LABEL TAB;
16 0B8E  07 099C  2647       BEGIN INTEGER SAVER1;     SAVER1 := R1;
16 0B92  07 09A0  2648          R0 := #47F0E000 OR INSCOUNTER;
16 0B9A  07 09A0  2649          R3 := IDLOC2(R7);
16 0B9E  07 09A0  2650          PROGRAM(R3+2) := R0;     R0 := R0 SHRL 16;    PROGRAM(R3) := R0;
16 0BAA  07 09A0  2651          R1 := SAVER1;     R7 := SIGN;
16 0BB2  07 09A0  2652       END;
16 0BB4  07 09A0  2653
```

```
16 0BB4  07 09A0   2654     PROCEDURE GOOTOO(R1);  COMMENT EMIT CODE FOR GOTO STATEMENT;
16 0BB4  07 09A0   2655        BEGIN INTEGER SAVER1; SAVER1:=R1;  R10 := TYPEINFO(R7);
16 0BBC  07 09A4   2656           IF R10 ¬= CLN THEN
16 0BC4  07 09A4   2657           BEGIN COMMENT OUT OF BLOCK;
16 0BC4  07 09A4   2658              R0 := R10 SHLL 20 OR #50000000 OR MP;     EMIT;
16 0BDE  07 09A4   2659              R0 := IDLOC1(R7); IF R0 ¬= LOGSEG THEN
16 0BEA  07 09A4   2660              BEGIN  COMMENT OUT OF PROCEDURE;
16 0BEA  07 09A4   2661                 CHAINOUT; R0 := R0 OR #58F00000; EMIT;
16 0CC6  07 09A4   2662                 COMMENT ASSUME BOTH BASE REGISTERS REQUIRED;
16 0CC6  07 09A4   2663                 R0 := #18EF0000; EMIT;  R0 := #5AF00000 OR ADDRSTEP; EMIT;
16 0C2A  07 09A4   2664              END;
16 0C2A  07 09A4   2665           END;
16 0C2A  07 09A4   2666           R0 := IDLOC2(R7) OR #47F0E000;     EMIT;
16 0C3E  07 09A4   2667           R1 := SAVER1;     R7 := SIGN;
16 0C46  07 09A4   2668        END;
16 0C48  07 09A4   2669
16 0C48  07 09A4   2670           INTEGER SAVER1;     SAVER1 := R1;
16 0C4C  07 09A8   2671           CASE R3 OF
16 0C4C  07 09A8   2672           BEGIN
16 0C4C  07 09A8   2673              PROCEDUREENTRY;     PROCEXIT;
16 0C66  07 09A8   2674              BLOCKCALLNENTRY;     BLOCKEXIT;
16 0C76  07 09A8   2675              LABELDEC;       GOOTOO;
16 0CE6  07 09A8   2676           END;
16 0CA2  07 09A8   2677           R1 := SAVER1;
16 0CA6  07 09A8   2678        END;

                  SEGMENT 16  NAME = SEG#16       LENGTH = 0DE8  BASE REG = 15

11 052A  07 09A8   2679
11 052A  07 09A8   2680     PROCEDURE INDICATEBRANCH(R1);
11 052A  07 09A8   2681        BEGIN  R0 := R0 SHLL 16;  R3 := LOGPOINTER + 4;
11 0536  07 09A8   2682           LOGPOINTER := R3;  LOGSTACK(R3) := R0;
11 053E  07 09A8   2683        END;
11 0540  07 09A8   2684
11 0540  07 09A8   2685     PROCEDURE GETLENGTH(R1);
11 0540  07 09A8   2686        BEGIN  COMMENT  GET STRING LENGTH;
11 0540  07 09A8   2687           LOGICAL SAVER1;  SAVER1 := R1;
11 0544  07 09AC   2688           R3 := R7 AND MASK;  R1 := R7 SHRL 24;
11 0550  07 09AC   2689           IF R1 = STRYNG THEN
11 0558  07 09AC   2690           BEGIN R3 := R3 + CPTBASE; IC(R3,CONSPTTAB(R3)); R3:=R3 AND #FF;
11 0564  07 09AC   2691           END ELSE  R3 := NAMETABLE(R3+4) SHRL 16;
11 0570  07 09AC   2692           R1 := SAVER1;
11 0574  07 09AC   2693        END;
11 0576  07 09AC   2694
11 0576  07 09AC   2695     PROCEDURE LOGCOMPARE(R1);
11 0576  07 09AC   2696        COMMENT  EMIT CLI INSTRUCTION FOR LOGICAL EXPRESSIONS;
11 0576  07 09AC   2697        BEGIN  LOGICAL SAVER1;  SAVER1 := R1;
11 057A  07 09B0   2698           GETADDRESS;  R0 := CLII OR #00010000 OR R3;
11 059C  07 09B0   2699           EMIT;  R1 := SAVER1;
11 05A0  07 09B0   2700        END;
11 05A2  07 09B0   2701
11 05A2  07 09B0   2702     PROCEDURE INITIALIZELINK(R1);
11 05A2  07 09B0   2703        BEGIN
11 05A2  07 09B0   2704           R0 := INSCOUNTER - 4;  R3 := LOGPOINTER + 4;
11 05B2  07 09B0   2705           LOGPOINTER := R3;  LOGSTACK(R3) := R0;
11 05BA  07 09B0   2706        END;
11 05BC  07 09B0   2707
11 05BC  07 09B0   2708     PROCEDURE IFJARG2 (R1);
```

```
11 05BC  07 09B0  2709      BEGIN  COMMENT  IFJ;
11 05BC  07 09B0  2710         LOGICAL SAVER1;  SAVER1 := R1;
11 05C0  07 09B4  2711         IF R7 >= 0 THEN
11 05C6  07 09B4  2712         BEGIN  LOGCOMPARE;  R0 := 7;  LOGBRANCH;
11 05CA  07 09B4  2713            INITIALIZELINK;
11 05DE  07 09B4  2714         END ELSE
11 05DE  07 09B4  2715         BEGIN  R3 := R7 SHLL 8 SHRL 24;
11 05EC  07 09B4  2716            IF R3 = 1 THEN
11 05F4  07 09B4  2717            BEGIN  R0 := LSTACKM4(R5) OR CLII OR #00010000;
11 0600  07 09B4  2718               EMIT;  R3 := LSTACKM4(R5);  RELEASE;  R5 := R5 -4;
11 0620  07 09B4  2719               R0 := 7;  LOGBRANCH;  INITIALIZELINK;
11 0634  07 09B4  2720            END ELSE
11 0634  07 09B4  2721            BEGIN  R3 := LOGPOINTER;  IC(R10,LOGSTACK(R3));
11 0640  07 09B4  2722               R10 := R10 AND #FF;
11 0644  07 09B4  2723               CASE R10 OF
11 0644  07 09B4  2724               BEGIN
11 0644  07 09B4  2725                  BEGIN  R0 := LOGSTACK(R3) SHRL 16 AND #FF;
11 0658  07 09B4  2726                     R0 := R0 XOR #F;
11 065C  07 09B4  2727                     LOGBRANCH;  R3 := INSCOUNTER;  FIXUP;
11 0678  07 09B4  2728                     R3 := LOGPOINTER - 4;  LOGPOINTER := R3;
11 0684  07 09B4  2729                     LINKBRANCHES;
11 0690  07 09B4  2730                  END;
11 0690  07 09B4  2731                  BEGIN
11 0694  07 09B4  2732                     R0 := LOGSTACK(R3) SHRL 16 AND #FF;
11 06A0  07 09B4  2733                     LOGBRANCH;  R3 := LOGPOINTER - 4;
11 06B4  07 09B4  2734                     LOGPOINTER := R3;  LINKBRANCHES;
11 06C4  07 09B4  2735                     R3 := INSCOUNTER;  FIXUP;
11 06D4  07 09B4  2736                  END;
11 06D4  07 09B4  2737               END;
11 06E0  07 09B4  2738            END;
11 06EC  07 09B4  2739         END;
11 06EC  07 09B4  2740            R7 := #86000000;  R3 := LOGPOINTER + 4;
11 06EC  07 09B4  2741            LOGPOINTER := R3;  R0 := 0;  LOGSTACK(R3) := R0;
11 06F8  07 09B4  2742         R1 := SAVER1;
11 06FC  07 09B4  2743      END;
11 06FE  07 09B4  2744
11 06FE  07 09B4  2745    PROCEDURE MERGE(R1);
11 06FE  07 09B4  2746       COMMENT  JOIN LINKS OF THE TWO TOP EMEMENTS OF LOGSTACK;
11 06FE  07 09B4  2747       BEGIN  R3 := LOGPOINTER - 8;  R0 := LOGSTACK(R3);
11 070A  07 09B4  2748          IF R0 = 0 THEN
11 0710  07 09B4  2749          BEGIN  R0 := LOGSTACK(R3+4); LOGSTACK(R3) := R0; R3 := R3 + 4;
11 071C  07 09B4  2750             LOGPOINTER := R3;  R0 := LOGSTACK(R3+4);  LOGSTACK(R3) := R0;
11 0728  07 09B4  2751             GOTO XIT;
11 072C  07 09B4  2752          END;
11 072C  07 09B4  2753          R3 := R0;
11 072E  07 09B4  2754          WHILE R3 ¬= 0 DO
11 0734  07 09B4  2755          BEGIN  R10 := R3;  R3 := PROGRAM(R3+2) AND #1FFF;
11 073E  07 09B4  2756          END;
11 0742  07 09B4  2757          R3 := LOGPOINTER-4;  R0 := LOGSTACK(R3);  PROGRAM(R10+2) := R0;
11 0752  07 09B4  2758        R0 := LOGSTACK(R3+4);  LOGSTACK(R3) := R0;  LOGPOINTER := R3;
11 075E  07 09B4  2759    XIT:END;
11 0760  07 09B4  2760
11 0760  07 09B4  2761    PROCEDURE CORRECTREG(R1);
11 0760  07 09B4  2762       COMMENT  SETS PROPER RESULT REGISTER FOR CASE AND IF EXPR;
11 0760  07 09B4  2763       BEGIN  LOGICAL SAVER1, SAVER6, TEMP;
11 076C  07 09C0  2764          SAVER1 := R1;  SAVER6 := R6;
11 0768  07 09C0  2765          R0 := LSTACKM4(R5);  IF R3 > 5 THEN R3 := 1;  R6 := R3;
11 077A  07 09C0  2766          CASE R3 OF
```

```
11 077A   07 09C0   2767          BEGIN
11 C77A   07 09C0   2768             IF R0 ¬= #80200000 AND R0 ¬= #2000 THEN                          1
11 0792   07 09C0   2769             BEGIN  TEMP := R0;  R3 := R0;  RELEASE;  R5 := R5-4;
11 07A8   07 09C0   2770                GENREG;  COMMENT GENREG MUST OBTAIN R2 HERE;
11 C7B4   07 09C0   2771                R0 := TEMP;  IF R0 < 0 THEN
11 07BE   07 09C0   2772                R0 := R0 XOR SIGN SHRL 4 OR #18200000  ELSE
11 C7CA   07 09C0   2773                BEGIN  R1 := #2000;  LSTACKM4(R5) := R1;
11 C7D6   07 09C0   2774                   R0 := R0 AND #FFF;  IF ¬= THEN
11 07DE   07 09C0   2775                   R0 := #41200000 OR TEMP  ELSE
11 C7E6   07 09C0   2776                   R0 := TEMP SHLL 4 OR #18200000;
11 C7F6   07 09C0   2777                END;
11 07F6   07 09C0   2778                EMIT;
11 0802   C7 09C0   2779             END;
11 0802   07 09C0   2780             IF R0 ¬= #80000000 THEN                                          2
11 080E   07 09C0   2781             BEGIN  R0 := R0 XOR SIGN SHRL 4 OR #38000000;  EMIT;
11 0826   07 09C0   2782                R3 := LSTACKM4(R5); RELEASE; R5 := R5-4;
11 083A   C7 09C0   2783                R0 := #C0000000; FLREG;  COMMENT FLREG MUST OBTAIN F0;
11 084A   C7 09C0   2784             END;
11 C84A   07 09C0   2785             IF R0 ¬= #80000000 THEN                                          3
11 0856   07 09C0   2786             BEGIN  R0 := R0 XOR SIGN SHRL 4 OR #28000000;  EMIT;
11 086E   C7 09C0   2787                R3 := LSTACKM4(R5); RELEASE;  R5 := R5-4;
11 0882   07 09C0   2788                R0 := #80000000; FLREG;  COMMENT FLREG MUST OBTAIN F01;
11 0892   C7 09C0   2789             END;
11 0892   07 09C0   2790             BEGIN  R0 := #38000000;  FORCECOMPLEX;  R3 := LSTACKM4(R5);      4
11 08AA   C7 09C0   2791                R6 := 4; RELEASE; R5 := R5-4;  R0 := #40000000; PRFLREG;
11 08CE   07 09C0   2792             END;
11 C8CE   C7 09C0   2793             BEGIN  R0 := #28000000;  FORCECOMPLEX;  R3 := LSTACKM4(R5);      5
11 08E6   07 09C0   2794                R6 := 5; RELEASE; R5 := R5-4;  R0 := #00000000; PRFLREG;
11 090A   07 09C0   2795             END;
11 C90A   07 09C0   2796          END;
11 0922   07 09C0   2797          R6 := SAVER6;  R1 := SAVER1;
11 092A   C7 09C0   2798       END;
11 C92C   07 09C0   2799
11 092C   07 09C0   2800       PROCEDURE FILLCASETABLE(R1);
11 092C   C7 09C0   2801          COMMENT  FILL TABLE BACKWARDS.  START WITH ADDRESS IN TOP
11 C92C   07 09C0   2802                   LOGSTACK POSITION.  LAST ONE IN FIRSTCASE1(2).
11 092C   C7 09C0   2803                   LENGTH OF TABLE IN FIRSTCASE1;
11 092C   07 09C0   2804          BEGIN  LOGICAL SAVER1;  SAVER1 := R1;
11 0930   07 09C4   2805             R10 := FIRSTCASE1 SHLL 2;
11 C938   07 09C4   2806             R10 := R10 + INSCOUNTER;  INSCOUNTER := R10;
11 C940   07 09C4   2807             R10 := R10 - 4;  COMMENT RA HAS ADDRESS OF LAST CASETABLE ENTRY;
11 0944   C7 09C4   2808             R3 := LOGPOINTER;  R3 := LOGSTACK(R3);
11 094C   07 09C4   2809             FOR R1 := 2 STEP 1 UNTIL FIRSTCASE1 DO
11 0950   07 09C4   2810             BEGIN  R3 := PROGRAM(R3+2);  R0 := R3 ++ #47F0E004;
11 095E   07 09C4   2811                PROGRAM(R10+2) := R0;  R0 := R0 SHRL 16;  PROGRAM(R10) := R0;
11 096A   C7 09C4   2812                R3 := R3 AND #1FFF;  R10 := R10 - 4;
11 0972   07 09C4   2813             END;
11 C97E   C7 09C4   2814             R0 := FIRSTCASE1( 2)++ #47F0E004;
11 0986   07 09C4   2815             PROGRAM(R10+2) := R0;  R0 := R0 SHRL 16;  PROGRAM(R10) := R0;
11 0992   07 09C4   2816             R1 := SAVER1;
11 C996   07 09C4   2817          END;
11 C998   C7 09C4   2818
11 0998   C7 09C4   2819       PROCEDURE ANDORARG1(R1);
11 0998   07 09C4   2820          BEGIN  COMMENT  IF AND, ANDFLAG IS ON,  IF OR, ANDFLAG IS OFF;
11 0998   07 09C4   2821             LOGICAL  SAVER1;  SAVER1 := R1;
11 C99C   C7 09C8   2822             IF R7 >= 0 THEN
11 09A2   07 09C8   2823             BEGIN  LOGCOMPARE;  TEST(ANDFLAG);
11 09AA   C7 09C8   2824                IF  =  THEN R0 := 7 ELSE R0 := 8;
```

```
11 09BA  07 09C8  2825            R3 := LOGPOINTER+4;  LOGPOINTER := R3;  R10 := 0;
11 09CA  C7 09C8  2826            LOGSTACK(R3) := R10;
11 09CE  07 09C8  2827            LOGBRANCH;  INITIALIZELINK;
11 09DE  07 09C8  2828         END ELSE
11 09DE  C7 09C8  2829         BEGIN  R3 := R7 SHLL 8 SHRL 24;
11 09EC  07 09C8  2830            IF R3 = 1 THEN
11 09F4  07 09C8  2831            BEGIN  R0 := LSTACKM4(R5) OR CLII OR #00010000;  EMIT;
11 0A0C  07 09C8  2832               R3 := LSTACKM4(R5);  R5 := R5 - 4;  RELEASE; TEST(ANDFLAG);
11 0A20  07 09C8  2833               IF  =  THEN R0 := 7 ELSE R0 := 8;
11 0A34  07 09C8  2834               R3 := LOGPOINTER+4;  LOGPOINTER := R3;  R10 := 0;
11 0A44  07 09C8  2835               LOGSTACK(R3) := R10;
11 0A48  07 09C8  2836               LOGBRANCH;  INITIALIZELINK;
11 0A58  07 09C8  2837            END ELSE
11 0A58  C7 09C8  2838            BEGIN  R3 := LOGPOINTER;  R10 := R3 - 4;  LOGPOINTER := R10;
11 0A6A  07 09C8  2839               IC(R10,LOGSTACK(R3));  R10 := R10 AND #FF;  TEST(ANDFLAG);
11 0A76  07 09C8  2840               IF  =  THEN R10 :=R10 XOR #3;
11 0A7E  07 09C8  2841               CASE R10 OF
11 0A7E  C7 09C8  2842               BEGIN
11 0A7E  C7 09C8  2843                  BEGIN  R0 := 0;  IC(R0,LOGSTACK(R3+1));
11 0A8E  07 09C8  2844                     LOGBRANCH;  LINKBRANCHES;  R3 := INSCOUNTER;  FIXUP;
11 0AB6  07 09C8  2845                     R3 := LOGPOINTER+4;  LOGPOINTER := R3;
11 0AC2  07 09C8  2846                     R0 := LOGSTACK(R3-4);  LOGSTACK(R3) := R0;
11 0ACA  C7 09C8  2847                     R0 := 0;  LOGSTACK(R3-4) := R0;
11 0AD2  07 09C8  2848                  END;
11 0AD2  07 09C8  2849                  BEGIN  R0 := LOGSTACK(R3) SHRL 16 AND #FF XOR #F;
11 0AE6  07 09C8  2850                     LOGBRANCH;  R0 := LOGPOINTER+4;  LOGPOINTER := R0;
11 0AFE  07 09C8  2851                     R3 := INSCOUNTER;
11 0B02  07 09C8  2852                     FIXUP;  R3 := LOGPOINTER;  R0 := LOGSTACK(R3-4);
11 0B16  C7 09C8  2853                     LOGSTACK(R3) := R0;  R0 := 0;  LOGSTACK(R3-4) :=R0;
11 0B22  C7 09C8  2854                     LINKBRANCHES;
11 0B2E  07 09C8  2855                  END;
11 0B2E  07 09C8  2856               END;
11 0B3A  07 09C8  2857            END;
11 0B3A  C7 09C8  2858         END;
11 0B3A  07 09C8  2859         R6 := R6 OR #06000000;  R1 := SAVER1;
11 0B42  07 09C8  2860      END;
11 0B44  C7 09C8  2861
11 0B44  07 09C8  2862      PROCEDURE MERGE24(R1);
11 0B44  07 09C8  2863         BEGIN  LOGICAL SAVER1;  SAVER1 := R1;
11 0B48  C7 09CC  2864            R3 := LOGPOINTER - 12;  R0 := LOGSTACK(R3);
11 0B54  07 09CC  2865            R1 := LOGSTACK(R3+4);  LOGSTACK(R3) := R1;
11 0B5C  07 09CC  2866            LOGSTACK(R3+4) := R0;  MERGE;
11 0B64  07 09CC  2867            R0 := LOGPOINTER-4;  LOGPOINTER := R0;  MERGE;
11 0B74  07 09CC  2868            R3 := LOGPOINTER + 4;  LOGPOINTER := R3;  R0 := LOGSTACK(R3+4);
11 0B84  07 09CC  2869            LOGSTACK(R3) := R0;
11 0B88  07 09CC  2870            R1 := SAVER1;
11 0B8C  07 09CC  2871         END;
11 0B8E  07 09CC  2872      PROCEDURE MERGE34(R1);
11 0B8E  07 09CC  2873         BEGIN  LOGICAL SAVER1;  SAVER1 := R1;
11 0B92  07 09D0  2874            R3 := LOGPOINTER - 4;  LOGPOINTER := R3;  MERGE;
11 0BA2  07 09D0  2875            R3 := LOGPOINTER + 4;  LOGPOINTER := R3;
11 0BAE  07 09D0  2876            R0 := LOGSTACK(R3+4);  LOGSTACK(R3) := R0;
11 0BB6  07 09D0  2877            R3 := LOGPOINTER-12;  R0 := LOGSTACK(R3);  R1 := LOGSTACK(R3+4);
11 0BC6  C7 09D0  2878            LOGSTACK(R3) := R1;  LOGSTACK(R3+4) := R0;  MERGE;
11 0BD2  07 09D0  2879            R1 := SAVER1;
11 0BD6  07 09D0  2880         END;
11 0BD8  C7 09D0  2881
11 0BD8  C7 09D0  2882      SEGMENT PROCEDURE UNCONDJUMP(R1);
```

```
11 0BD8   07 09D0          2883        BEGIN  LOGICAL SAVER1,LNGTH;  SAVER1 := R1;
18 00C4   07 09D8          2884          IF R7 >= 0 THEN
18 00CA   07 09D8          2885          BEGIN  R3 := R7 AND OPCODEMASK;
18 00D0   07 09D8          2886             IF R3 = NULLST THEN GOTO STMNT ELSE
18 00D8   07 09D8          2887             IF R3 = #5B000000 THEN   COMMENT  RCCLID;
18 00E0   07 09D8          2888             BEGIN  RECORDALLOCATE;  R3 := 9;  GOTO SKIP;
18 00F4   07 09D8          2889             END ELSE GETTYPE;
18 0104   07 09D8          2890             R1 := R7 SHRL 24;
18 010A   07 09D8          2891             IF R1 = FUNCID AND R3 = 0 THEN
18 0118   07 09D8          2892             BEGIN CALLPROPROCWOPARAM; GOTO STMNT;
18 0128   07 09D8          2893             END;
18 0138   07 09D8          2894             CASE R3 OF
18 0138   07 09D8          2895             BEGIN
18 0138   07 09D8          2896                BEGIN  CONVERT;  LOADREG;  R3 := 1;  END;
18 014A   07 09D8          2897                BEGIN  CONVERT;  LOADREG;  R3 := 2;  END;
18 0168   07 09D8          2898                BEGIN  CONVERT;  LOADREG;  R3 := 3;  END;
18 0186   07 09D8          2899                BEGIN  CONVERT;  LOADREG;  R3 := 4;  END;
18 01A4   07 09D8          2900                BEGIN  CONVERT;  LOADREG;  R3 := 5;  END;
18 01C2   07 09D8          2901                R3 := 6;
18 01CA   07 09D8          2902                BEGIN  GETLENGTH;  R3 := R3 SHLL 16;  LNGTH := R3;
18 01E2   07 09D8          2903                   GETADDRESS;  R10 := R3;  GENREG;  R0:=R0 OR R10 OR LAA;
18 0202   07 09D8          2904                   EMIT;
18 021E   07 09D8          2905                   R3 := LSTACKM4(R5) SHLL 1 SHRL 9;  LSTACKM4(R5):=R3;
18 023E   07 09D8          2906                   R3 := 7;
18 0242   07 09D8          2907                END;
18 0242   07 09D8          2908                BEGIN  LOADREG;  R3 := 8;
18 0256   07 09D8          2909                END;
18 0256   07 09D8          2910                BEGIN  LOADREG;  R3 := 9;
18 026A   07 09D8          2911                END;
18 026A   07 09D8          2912                BEGIN  COMMENT  NULLST;
18 026E   07 09D8          2913                   R3 := 0;
18 0272   07 09D8          2914                END;
18 0272   07 09D8          2915             END;
18 027E   07 09D8          2916   SKIP:     IF R3 <= 5 THEN
18 0286   07 09D8          2917             BEGIN
18 0286   07 09D8          2918             IF R7 >= 0 THEN R7 := R3  SHLL 24 OR SIGN;
18 0296   07 09D8          2919             TEST(IFEXP);     IF ¬= THEN CONVERTRESULT;
18 02A8   07 09D8          2920             R3 := R7 SHLL 1 SHRL 25;
18 02B2   07 09D8          2921             END;
18 02B2   07 09D8          2922          END ELSE
18 02B2   07 09D8          2923          BEGIN  R3 := R7 AND #7F000000;
18 02BC   07 09D8          2924             IF R3 = 0 THEN GOTO STMNT ELSE R3 := R3 SHRL 24;
18 02C6   07 09D8          2925             IF R3 = 7 THEN
18 02CE   07 09D8          2926             BEGIN  R1 := R7 AND TYPEMASK;  LNGTH := R1;  END;
18 02D8   07 09D8          2927             IF R3 <= 5 OR R3 >= 8 THEN
18 02E8   07 09D8          2928             BEGIN  LOADREG;  IF ¬IFEXP THEN CONVERTRESULT;
18 0306   07 09D8          2929                R3 := R7 SHLL 1 SHRL 25;
18 0310   07 09D8          2930             END;
18 0310   07 09D8          2931          END;
18 0310   07 09D8          2932          IF R3 = 6 THEN
18 0318   07 09D8          2933             BEGIN  LOGICAL SAVE;
18 0318   07 09DC          2934                SET(LOGFLAG);  SET(ANDFLAG);  ANDORARG1;  R3 := LOGPOINTER;
18 032C   07 09DC          2935                TEST(UJIFEXP);  IF  =  THEN
18 0338   07 09DC          2936                BEGIN  R0 := LOGSTACK(R3-12);  R10 := LOGSTACK(R3-4);
18 0340   07 09DC          2937                   LOGSTACK(R3-12) := R10;  LOGSTACK(R3-4) := R0;
18 0348   07 09DC          2938                   R0 := LOGSTACK(R3-8);  R10 := LOGSTACK(R3);
18 0350   07 09DC          2939                   LOGSTACK(R3-8) := R10;  LOGSTACK(R3) := R0; GOTO STMNT;
18 035C   07 09DC          2940                END;
```

```
18 029C  07 09DC   2941              R0 := LOGSTACK(R3-8);  R10 := LOGSTACK(R3-4);
18 02A4  07 09DC   2942              LOGSTACK(R3-8):= R10;  R10 := LOGSTACK(R3);
18 02AC  07 09DC   2943              LOGSTACK(R3-4) := R10;  LOGSTACK(R3) :=R0;
18 02B4  07 09DC   2944              R7 := #86000000;  R6 := R6 OR R7;
18 02BA  07 09DC   2945              MERGE24;  TEST(IFEXP);  IF ¬= THEN GOTO STMNT;
18 02CE  07 09DC   2946              R3:=LOGPOINTER-4; LOGPOINTER:=R3; R0 :=LOGSTACK(R3+4);
18 02DE  07 09DC   2947              SAVE := R0;  R0 := #0200;  INDICATEBRANCH;  R0 := SAVE;
18 02F6  07 09DC   2948              R3 := LOGPOINTER+4;  LOGPOINTER := R3;  LOGSTACK(R3) := R0;
18 03C2  07 09DC   2949              GOTO XIT;
18 030A  07 09DC   2950           END ELSE
18 030A  07 09DC   2951           BEGIN  LOGICAL SAVER6;  SAVER6 := R6;  R6 := R3;
18 0314  07 09E0   2952              R7 := R3 SHLL 24 OR SIGN;  IF R3 = 7 THEN R7:=R7 OR LNGTH;
18 C32A  07 09E0   2953              IF R3 = 9 THEN
18 0332  07 09E0   2954              BEGIN  R5 := R5 - 4;  RESETRECORD;  R5 := R5 + 4;
18 0346  C7 09E0   2955              END;
18 0346  07 09E0   2956              CORRECTREG;
18 0352  07 09E0   2957              TEST(IFEXP); IF ¬= THEN
18 035A  07 09E0   2958              BEGIN  R0 := #F;  LOGBRANCH;  LINKBRANCHES;
18 0376  07 09E0   2959                 TEST(CLFLAG); IF ¬= THEN
18 037E  07 09E0   2960                 BEGIN  R3 := LSTACKM4(R5); RELEASE;  R5 := R5 - 4; END;
18 0392  07 09E0   2961              END;  R6 := SAVER6 OR R7;
18 C398  C7 09E0   2962           END;
18 C398  07 09E0   2963        GOTO XIT;
18 C39C  07 09E0   2964  STMNT: R0 := #F;  LOGBRANCH;  LINKBRANCHES;  R7 := SIGN;
18 03BC  C7 09E0   2965  XIT:   R1 := SAVER1;
18 03C0  C7 09E0   2966      END;

                  SEGMENT 18  NAME = SEG#18      LENGTH = 0408  BASE REG = 15

11 0BD8  07 09E0   2967
11 0BD8  C7 09E0   2968    SEGMENT PROCEDURE STFUNCTION (R1);
11 0BD8  C7 09E0   2969       BEGIN COMMENT  GENERATES CODE TO LOAD R3 OR F01/F23 WITH VALUE,
19 0000  07 09E0   2970                      RETURNS (LENGTH, TYPE) IN R3;
19 0000  07 09E0   2971          INTEGER SR1, SAVEADD, SAVETYPE;  SR1 := R1;
19 0004  07 09EC   2972          IF R7>= 0 THEN
19 000A  07 09EC   2973          BEGIN COMMENT ID;
19 000A  C7 09EC   2974             GETTYPE;  SAVETYPE := R3;  GETADDRESS;
19 0026  07 09EC   2975             DUMPALLGENREG;    DUMPALLFLREG;
19 003E  07 09EC   2976             SAVEADD := R3;    R3 := SAVETYPE;    IF R3 = 7 THEN
19 004E  C7 09EC   2977             BEGIN  COMMENT IT IS A STRING;
19 004E  C7 09EC   2978                IF R7 < 0 THEN R3 := R7 SHLL 8 SHRL 24 ELSE GETLENGTH;
19 006E  07 09EC   2979                R3 := R3 SHLL 16 OR 7;
19 0076  C7 09EC   2980             END;
19 0076  07 09EC   2981             R1 := SAVETYPE;  SAVETYPE := R3;  R3 := SAVEADD;
19 0082  07 09EC   2982             CASE R1 OF
19 0082  07 09EC   2983             BEGIN
19 0082  07 09EC   2984                R0 := #58300000;  COMMENT INTEGER;
19 008E  C7 09EC   2985                R0 := #78000000;  COMMENT REAL;
19 0096  07 09EC   2986                R0 := #68000000;  COMMENT LONG REAL;
19 009E  07 09EC   2987                BEGIN  COMMENT COMPLEX;
19 00A2  C7 09EC   2988                   R0 := #78000000 OR R3; EMIT; R3 := R3 + 4;
19 00B8  07 09EC   2989                   R0 := #78200000;
19 00BC  07 09EC   2990                END;
19 00BC  C7 09EC   2991                BEGIN  COMMENT LONG CCOMPLEX;
19 00C0  07 09EC   2992                   R0 := #68000000 OR R3; EMIT; R3 := R3 + 8;
19 00D6  C7 09EC   2993                   R0 := #68200000 ;
19 00DA  07 09EC   2994                END;
19 00DA  07 09EC   2995                R0 := #43300000; COMMENT LOGICAL;
```

```
19 00E2  07 09EC  2996                R0 := #41300000; COMMENT STRING;
19 00EA  07 09EC  2997                R0 := #58300000; COMMENT BITS;
19 00F2  07 09EC  2998                BEGIN  COMMENT  REFERENCE;
19 00F6  07 09EC  2999                   IF R3 > 0 THEN R0 := #58300000 ELSE
19 0100  07 09EC  3000                   BEGIN  R3 := R3 SHLL 1 SHRL 5;  R0 := #18300000;
19 0110  07 09EC  3001                      RESETRECORD;
19 011C  07 09EC  3002                   END;
19 011C  07 09EC  3003                END;
19 011C  07 09EC  3004             END;
19 0144  07 09EC  3005             R0 := R0 OR R3; EMIT;
19 0152  07 09EC  3006          END  ELSE
19 0152  07 09EC  3007          BEGIN  COMMENT  IT IS A  <T EXP>;
19 0156  07 09EC  3008             R6 := R7 SHLL 1 SHRL 25;
19 0160  07 09EC  3009             R10 := R7 SHLL 8 SHRL 24;
19 016A  07 09EC  3010             IF R6 = 4 OR R6 = 5 THEN LOADREG ELSE
19 0186  07 09EC  3011             IF R10 = 0 AND R6 = 6 THEN EVALUATELOG;
19 01A4  07 09EC  3012             R3 := LSTACKM4(R5); RELEASE; R5 := R5-4;
19 01B8  07 09EC  3013             DUMPALLGENREG;    DUMPALLFLREG;
19 01D0  07 09EC  3014             R1 := R6; IF R1 >= 8 THEN R1 := 1;
19 01DE  07 09EC  3015             CASE R1 OF
19 01DE  07 09EC  3016             BEGIN
19 01DE  07 09EC  3017                IF R3 ¬= #80300000 THEN
19 01EE  07 09EC  3018                BEGIN  COMMENT INTEGER, BITS, REFERENCE;
19 01EE  07 09EC  3019                   R0 := #08300000; ASSEMBLE;
19 01FE  07 09EC  3020                END;
19 01FE  07 09EC  3021                IF R3 ¬= #80000000 THEN
19 02CA  07 09EC  3022                BEGIN  COMMENT REAL;
19 020A  07 09EC  3023                   R0 := #08000000; ASSEMBLE;
19 021A  07 09EC  3024                END;
19 021A  07 09EC  3025                IF R3 ¬= #80000000 THEN
19 0226  07 09EC  3026                BEGIN COMMENT LONG REAL;
19 0226  07 09EC  3027                   R0 := #08000000; ASSEMBLE;
19 0236  07 09EC  3028                END;
19 0236  07 09EC  3029                IF R3 ¬= #80020000 THEN
19 0242  07 09EC  3030                BEGIN  COMMENT COMPLEX;
19 0242  07 09EC  3031                    R5 := R5 + 4; R0 := #38000000; FORCECOMPLEX; R5 := R5-4;
19 0256  07 09EC  3032                END;
19 025A  07 09EC  3033                IF R3 ¬= #80020000 THEN
19 0266  07 09EC  3034                BEGIN  COMMENT LONG COMPLEX;
19 0266  07 09EC  3035                    R5 := R5 + 4; R0 := #28000000; FORCECOMPLEX; R5 := R5-4;
19 027A  07 09EC  3036                END;
19 027E  07 09EC  3037                BEGIN  COMMENT LOGICAL;
19 0282  07 09EC  3038                   IF R3 > 0 THEN R0 := #43300000 OR R3 ELSE
19 028E  07 09EC  3039                   BEGIN R3 := R3 SHLL 1 SHRL 5; R0 := #18300000 OR R3;
19 02A0  07 09EC  3040                   END; EMIT;
19 02AC  07 09EC  3041                END;
19 02AC  07 09EC  3042                BEGIN  COMMENT STRING;
19 02B0  07 09EC  3043                   R0 := R3 AND MASK8;  IF R0 ¬= 0 THEN
19 02BC  07 09EC  3044                   BEGIN  R0 := LAA OR #300000 OR R3;  EMIT;
19 02D2  07 09EC  3045                   END ELSE
19 02D2  07 09EC  3046                   BEGIN
19 02D6  07 09EC  3047                      R3 := R3 SHLL 4 AND MASK4;
19 02DE  07 09EC  3048                      IF R3 ¬= #00030000 THEN
19 02E6  07 09EC  3049                      BEGIN  R0 := #18300000 OR R3;  EMIT;
19 02F8  07 09EC  3050                      END;
19 02F8  07 09EC  3051                   END;
19 02F8  07 09EC  3052                   R0 := R7 AND #FF0000 OR 7;  SAVETYPE := R0;
19 0306  07 09EC  3053                END;
```

```
19 0306   07 09EC          3054              END;
19 0326   07 09EC          3055              IF R6 ¬= 7 THEN SAVETYPE := R6;
19 0332   07 09EC          3056           END;
19 0332   07 09EC          3057           R3 := SAVETYPE;  RESET(R3FLAG);    R1 := SR1;
19 033E   07 09EC          3058        END;

                          SEGMENT 19  NAME = SEG#19      LENGTH = 0380  BASE REG = 15

11 0BD8   07 09EC          3059
11 0BD8   07 09EC          3060     SEGMENT PROCEDURE  STANDARDFUNCTION(R1);
11 0BD8   07 09EC          3061        BEGIN  LOGICAL SAVER1;
20 0000   07 09F0          3062           LOGICAL SAVEADD, TEMP;  BYTE RLF, LONGFLAG, TERMFLAG;
20 0000   07 09F8          3063           PROCEDURE IMAGLONGIMAG(R1);
20 0000   07 09F8          3064              BEGIN  ARRAY  2  LOGICAL TEMP;
20 0004   07 0A04          3065                 STM(R0,R1,TEMP);  CONVERT;  LOADREG;  R0 := TEMP;  FLREG;
20 002E   07 0A04          3066                 R6 := 3;  R3 := R0 OR SIGN;  R0 := R0 OR S;  ASSEMBLE;
20 0048   07 0A04          3067                 R1 := LSTACKM8(R5) AND MASK5 SHRL 4 OR LSTACKM4(R5);
20 0058   07 0A04          3068                 LSTACKM8(R5) := R1;  R5 := R5 - 4;  R4 := R4 - 4;
20 0064   07 0A04          3069                 R1 := FSTACKM4(R4) XOR SIGN;  FSTACKM4(R4) := R1;
20 0070   07 0A04          3070                 R1 := TEMP(4);
20 0074   07 0A04          3071              END;
20 0076   07 0A04          3072           PROCEDURE REALINT(R1);
20 0076   07 0A04          3073              BEGIN  LOGICAL SR0,SR1;
20 0076   07 0A0C          3074                 SR0 := R0;  SR1 := R1;  CONVERT;
20 0088   07 0A0C          3075                 IF R7 >= 0 THEN GETADDRESS ELSE
20 009A   07 0A0C          3076                 BEGIN  R3 := LSTACKM4(R5);  IF R3 < 0 THEN
20 00A8   07 0A0C          3077                    BEGIN  R6 := 2;  RELEASE;  R5 := R5 - 4;
20 00BC   07 0A0C          3078                       R0 := R3 XOR SIGN;  R3 := NEXTADDR + 3 AND SMASK;
20 00CE   07 0A0C          3079                       NEXTADDR := R3;    R10 := 4;
20 00D6   07 0A0C          3080                       INCRADDR;  R0 := R0 OR #70000000 OR R3;  EMIT;
20 00F4   07 0A0C          3081                    END ELSE
20 00F4   07 0A0C          3082                    BEGIN  R6 := 1;  RELEASE;  R5 := R5 - 4;  END;
20 010C   07 0A0C          3083                 END;
20 010C   07 0A0C          3084                 R0 := #41000000 OR R3;   EMIT;
20 011E   07 0A0C          3085                 R3 := 0; IF R3 ¬= R(6) THEN DUMPALLGENREG; R3:=#FFFFFFFF;
20 013A   07 0A0C          3086                 IF R3 = R(4) THEN GENREG ELSE
20 014E   07 0A0C          3087                 BEGIN R( 4):= R3;  GENREG;  R3 := 0;  R( 4):= R3;  END;
20 016A   07 0A0C          3088                 R0 := @REALINTEGER; R0 := NEG R0;
20 0170   07 0A0C          3089                 R3 := SR0;  EMITCALL;
20 0180   07 0A0C          3090                 R7 := #81000000;  R1 := SR1;
20 0188   07 0A0C          3091              END;
20 018A   07 0A0C          3092           PROCEDURE REALPART(R1);
20 018A   07 0A0C          3093              BEGIN  COMMENT R0 IS INPUT AS FOR FLREG;
20 018A   07 0A0C          3094                 LOGICAL SR1;  SR1 := R1;  IF R7 > 0 THEN
20 0194   07 0A10          3095                 BEGIN  GETADDRESS;  SAVEADD := R3;
20 01A4   07 0A10          3096                    TEST(RLF);  IF  =  THEN R3 := 0 ELSE
20 01B0   07 0A10          3097                    IF R7 < 0 THEN R3 := R7 XOR SIGN SHRL 24 ELSE GETTYPE;
20 01D4   07 0A10          3098                    IF R3 = 5 THEN R3 := 8;  R3 := R3 + SAVEADD;
20 01E4   07 0A10          3099                    SAVEADD := R3;  FLREG;
20 01F4   07 0A10          3100                    R0 := R0 OR #08000000;  R3 := SAVEADD;
20 01FC   07 0A10          3101                    ASSEMBLE;
20 0208   07 0A10          3102                 END ELSE
20 0208   07 0A10          3103                 BEGIN  R3 := LSTACKM4(R5);  IF R3 < 0 THEN
20 0216   07 0A10          3104                    BEGIN  R0 := FSTACK(R4-4) XOR SIGN;
20 021E   07 0A10          3105                       FSTACK(R4-4) := R0;  R4 := R4 + 4;
20 0226   07 0A10          3106                       R0 := R3 AND MASK5 OR SIGN;  LSTACKM4(R5) := R0;
20 0234   07 0A10          3107                       R3 := R3 AND MASK4 SHLL 4 OR SIGN;
20 0240   07 0A10          3108                       RELEASE;
```

```
20 024C  07 0A10  3109                END;
20 024C  07 0A10  3110             END;
20 024C  07 0A10  3111             R1 := SR1;
20 0250  07 0A10  3112          END;
20 0252  07 0A10  3113
20 0252  07 0A10  3114       SAVER1 := R1;
20 0256  07 0A10  3115       IF R3 <= 2 OR R3 = 6 THEN
20 0266  07 0A10  3116       BEGIN  COMMENT WRITE, WRITEON, SYSACT;
20 0266  07 0A10  3117          IF R3 ¬= 6 THEN R3 := R3-R3 ELSE R3 := #01000000; TEMP := R3;
20 027C  07 0A10  3118          CONVERT;  STFUNCTION;  R3 := R3 OR TEMP OR #B0000000;
20 0298  07 0A10  3119          R0 := @WRITEEDIT; R0 := NEG R0;  EMITCALL;   R7 := SIGN;
20 02AE  07 0A10  3120       END ELSE
20 02AE  07 0A10  3121       IF R3 <= 5 THEN
20 02EA  07 0A10  3122       BEGIN  COMMENT READ, READON, READCARD;
20 02BA  07 0A10  3123          IF R3 ¬= 5 THEN R3 := R3-R3 ELSE R3 := #01000000; TEMP := R3;
20 02D0  07 0A10  3124          IF R7 < 0 THEN
20 02D6  07 0A10  3125          BEGIN  R0 := LSTACKM4(R5); COMMENT IF R0 < 0 THEN;
20 02DA  07 0A10  3126             R6 := 1; ZRELEASE; R5 := R5 - 4;
20 02EE  07 0A10  3127             IF R0 ¬= #3000 THEN
20 02F6  07 0A10  3128             BEGIN  R0 := R0 OR LAA OR #300000; EMIT;
20 030A  07 0A10  3129             END;
20 030A  07 0A10  3130             R0 := R7 SHRL 24 AND #7F; R3 := R7 AND #FF0000 OR R0;
20 031C  07 0A10  3131          END ELSE
20 031C  07 0A10  3132          BEGIN  INTEGER SAVER3;
20 0320  07 0A14  3133             GETTYPE;  IF R3 = 7 THEN
20 0334  07 0A14  3134             BEGIN  GETLENGTH;  R3 := R3 SHLL 16 OR 7;
20 0348  07 0A14  3135             END;
20 0348  07 0A14  3136             SAVER3 := R3;  GETADDRESS;
20 0358  07 0A14  3137             R0 := R3 OR #41300000;  EMIT;   R3 := SAVER3;
20 036E  07 0A14  3138          END;
20 036E  07 0A14  3139          R3 := R3 OR TEMP OR #B0000000; R0 := @READIN; R0 := NEG R0;
20 037C  07 0A14  3140          EMITCALL;    R7 := SIGN;
20 038C  07 0A14  3141       END ELSE
20 038C  07 0A14  3142       IF R3 <= 22 THEN
20 0398  07 0A14  3143       BEGIN COMMENT STANDARD FUNCTIONS WITH INLINE CODE;
20 0398  07 0A14  3144          R3 := R3 - 6; CASE R3 OF BEGIN
20 039C  07 0A14  3145          BEGIN  COMMENT  ODD;
20 03A4  07 0A14  3146             LOADREG;    R1 := CPTBASE;  R0 := CONSPTTAB(R1)
20 03B4  07 0A14  3147             AND MASK OR LSTACKM4(R5) XOR SIGN
20 03C0  07 0A14  3148             OR #5400E000;  EMIT;  R3 := LSTACKM4(R5);  R6 := 1;
20 03DC  07 0A14  3149             RELEASE;  R0 := 0;  R3 := LOGPOINTER + 8;  R5 := R5 - 4;
20 03F8  07 0A14  3150             LOGPOINTER := R3;
20 03FC  07 0A14  3151             LOGSTACK(R3-4) := R0;  LOGSTACK(R3) := R0;
20 0404  07 0A14  3152             R0 := #0107;  INDICATEBRANCH;  R7 := #86000000;
20 0418  07 0A14  3153          END;
20 0418  07 0A14  3154          BEGIN  COMMENT  BITSTRING;
20 041C  07 0A14  3155             LOADREG;    R7 := #88000000;
20 042C  07 0A14  3156          END;
20 042C  07 0A14  3157          BEGIN  COMMENT  NUMBER;
20 0430  07 0A14  3158             LOADREG;  R7 := #81000000;
20 0440  07 0A14  3159          END;
20 0440  07 0A14  3160          BEGIN  COMMENT DECODE;
20 0444  07 0A14  3161             LOGICAL SAVER3;  RESET(TERMFLAG);  IF R7 >= 0 THEN
20 044E  07 0A18  3162             BEGIN  SET(TERMFLAG);  GETADDRESS;  END ELSE
20 045E  07 0A18  3163             BEGIN  R3 := LSTACKM4(R5);  R5 := R5 - 4;  END;
20 046A  07 0A18  3164             SAVER3 := R3;  GENREG;  R3 := R0 OR SIGN;  R0 := R0 OR S;
20 0484  07 0A18  3165             R6 := 1;    ASSEMBLE;
20 0494  07 0A18  3166             R0 := LSTACKM4(R5) XOR SIGN OR ICC OR SAVER3;
```

```
20 04A4  07 0A18  3167              EMIT;  IF ¬TERMFLAG THEN
20 04B8  07 0A18  3168              BEGIN  R3 := SAVER3;  R6 := 1;  RELEASE;  END;
20 04CC  07 0A18  3169              R7 := #81000000;
20 04D0  07 0A18  3170           END;
20 04D0  07 0A18  3171           BEGIN  COMMENT CODE;
20 04D4  07 0A18  3172              LOGICAL SAVER3;
20 04D4  07 0A1C  3173              LOADREG;  R3 := LSTACKM4(R5);  IF R3 >= 0 THEN
20 04EA  07 0A1C  3174              BEGIN  R6 := 1;  RELEASE;
20 04FA  07 0A1C  3175              END ELSE
20 04FA  07 0A1C  3176              BEGIN  R3 := R3 XOR SIGN;  R0 :=  R3 OR #50000000 OR
20 0508  07 0A1C  3177                 NEXTADDR + 3 AND SMASK;  R10 := R0 AND MASK;
20 051A  07 0A1C  3178                 NEXTADDR:=R10; EMIT; R3:=LSTACKM4(R5); R6:=1;
20 0532  07 0A1C  3179                 RELEASE;  R3 := NEXTADDR;  R10 := 4; INCRADDR;
20 0552  07 0A1C  3180                 R5 := R5 - 4;
20 0556  07 0A1C  3181              END;
20 0556  07 0A1C  3182              SAVER3 := R3;  GENREG;  R3 := R0 SHRL 8;
20 056C  07 0A1C  3183              LSTACKM4(R5) := R3;  R0 := R0 OR LAA OR SAVER3 ++ 3; EMIT;
20 0588  07 0A1C  3184     XIT:     R7 := #87000000;
20 058C  07 0A1C  3185           END;
20 058C  07 0A1C  3186           BEGIN  COMMENT  TRUNCATE;
20 0590  07 0A1C  3187              R0 := CASE1;    REALINT;
20 0598  07 0A1C  3188           END;
20 0598  07 0A1C  3189           BEGIN  COMMENT ROUND;
20 059C  07 0A1C  3190              R0 := CASE2;    REALINT;
20 05A4  07 0A1C  3191           END;
20 05A4  07 0A1C  3192           BEGIN  COMMENT ENTIER;
20 05A8  07 0A1C  3193              R0 := CASE3;    REALINT;
20 05B0  07 0A1C  3194           END;
20 05B0  07 0A1C  3195           BEGIN  COMMENT EXPONENT;
20 05B4  07 0A1C  3196              R0 := CASE4;    REALINT;
20 05BC  07 0A1C  3197           END;
20 05BC  07 0A1C  3198           BEGIN  COMMENT ROUNDED;
20 05C0  07 0A1C  3199              LOADREG; R7 := LSTACKM4(R5) AND MASK5;
20 05D4  07 0A1C  3200              R10 := NEXTADDR + 7 AND _8; R7 := R7 OR R10;
20 05E2  07 0A1C  3201              R0 := R7 OR #60000000; EMIT; R10 := R10 + 1;
20 05F8  07 0A1C  3202              R0 := #D7020000 OR R10; R3 := R10 SHLL 16; EMIT;
20 0610  07 0A1C  3203              R0 := R7 OR #6A000000; EMIT; R10 := R10 + 7;
20 0626  07 0A1C  3204              NEXTADDR := R10;  R7 := #82000000;
20 062E  07 0A1C  3205           END;
20 062E  07 0A1C  3206           BEGIN  COMMENT REALPART;
20 0632  07 0A1C  3207              SET(RLF);
20 0636  07 0A1C  3208              CONVERT;  R0 := #C0000000;  R6 := 2;  REALPART;
20 064C  07 0A1C  3209              R7 := #82000000;
20 0650  07 0A1C  3210           END;
20 0650  07 0A1C  3211           BEGIN  COMMENT IMAGPART;
20 0654  07 0A1C  3212              CONVERT;  IF R7 < 0 THEN
20 0664  07 0A1C  3213              BEGIN  R0 := LSTACKM4(R5);  IF R0 < 0 THEN
20 066E  07 0A1C  3214                 BEGIN  R3 := R0 AND MASK4 SHLL 4;
20 0678  07 0A1C  3215                    R0 := R0 AND MASK5 SHRL 4;  R0 := R0 OR R3 OR SIGN;
20 0686  07 0A1C  3216                 END ELSE R0 := R0 + 4;
20 068E  07 0A1C  3217                 LSTACKM4(R5) := R0;
20 0692  07 0A1C  3218              END;
20 0692  07 0A1C  3219              RESET(RLF);
20 0696  07 0A1C  3220              R0 := #C0000000;   R6 := 2;  REALPART;  R7 := #82000000;
20 06A6  07 0A1C  3221           END;
20 06A6  07 0A1C  3222           BEGIN  COMMENT LONGREALPART;
20 06AA  07 0A1C  3223              SET(RLF);
20 06AE  07 0A1C  3224              CONVERT;  R0 := #80000000;  R6 := 3;  REALPART;
```

Moore Business Forms, Inc. SW

```
20 06C4  07 0A1C  3225              R7 := #83000000;
20 06C8  07 0A1C  3226           END;
20 06C8  07 0A1C  3227           BEGIN  COMMENT LONGIMAGPART;
20 06CC  07 0A1C  3228              CONVERT;  IF R7 < 0 THEN
20 06DC  07 0A1C  3229              BEGIN  R0 := LSTACKM4(R5);  IF R0 < 0 THEN
20 06E6  07 0A1C  3230                 BEGIN  R3 := R0 AND MASK4 SHLL 4;
20 06F0  07 0A1C  3231                    R0 := R0 AND MASK5 SHRL 4;  R0 := R0 OR R3 OR SIGN;
20 06FE  07 0A1C  3232                 END ELSE  R0 := R0 + 8;
20 0706  07 0A1C  3233                 LSTACKM4(R5) := R0;
20 070A  07 0A1C  3234              END;
20 070A  07 0A1C  3235              RESET(RLF);
20 070E  07 0A1C  3236              R0 := #80000000;   R6 := 3;  REALPART;  R7 := #83000000;
20 071E  07 0A1C  3237           END;
20 071E  07 0A1C  3238           BEGIN COMMENT IMAG;
20 0722  07 0A1C  3239              R0 := #C0000000;    IMAGLONGIMAG;    R7 := #84000000;
20 072E  07 0A1C  3240           END;
20 072E  07 0A1C  3241           BEGIN COMMENT LONGIMAG;
20 0732  07 0A1C  3242              R0 := SIGN;    IMAGLONGIMAG;    R7 := #85000000;
20 073E  07 0A1C  3243           END; END;
20 0782  07 0A1C  3244        END ELSE
20 0782  07 0A1C  3245        IF R3 <= 36 THEN
20 078E  07 0A1C  3246        BEGIN  COMMENT SHORT/LONG ANALYSIS FUNCTIONS;
20 078E  07 0A1C  3247           IF R3 > 29 THEN SET(LONGFLAG) ELSE RESET(LONGFLAG);
20 07A2  07 0A1C  3248           CONVERT;   STFUNCTION;
20 07B6  07 0A1C  3249           R0 := R0-R0; R1 := STACKP8(R8) AND MASK / 12  -  22;
20 07C8  07 0A1C  3250           R0 := #41000000 OR R1;  EMIT;
20 07EA  07 0A1C  3251           R0 := @ANALFN; R0 := NEG R0; EMITCALL;
20 07EC  07 0A1C  3252           IF LONGFLAG THEN R0 := #80000000 ELSE R0 := #C0000000;
20 0800  07 0A1C  3253           FLREG;
20 080C  07 0A1C  3254           IF LONGFLAG THEN R7 := #83000000 ELSE R7 := #82000000;
20 0820  07 0A1C  3255        END ELSE
20 0820  07 0A1C  3256        IF R3 <= 38 THEN
20 082C  07 0A1C  3257        BEGIN  COMMENT COMPLEX SQRT FUNCTIONS;  INTEGER SAVERA;
20 082C  07 0A20  3258           IF R3 = 37 THEN R10 := 5 ELSE R10 := 6;  SAVERA := R10;
20 0844  07 0A20  3259           CONVERT;   STFUNCTION;   R3 := SAVERA SHLL 16;
20 0860  07 0A20  3260           R0 := @ARITHFN; R0 := NEG R0;  EMITCALL;  R10 := SAVERA;
20 0876  07 0A20  3261           IF R10 = 5 THEN R0 := #40000000 ELSE R0 := R0-R0; PRFLREG;
20 0894  07 0A20  3262           R7 := SAVERA;
20 0898  07 0A20  3263           IF R7 = 5 THEN R7 := #84000000 ELSE R7 := #85000000;
20 08AC  07 0A20  3264        END ELSE
20 08AC  07 0A20  3265        IF R3 <= 44 THEN
20 08B8  07 0A20  3266        BEGIN  COMMENT STRING CONVERSION FUNCTIONS;
20 08B8  07 0A20  3267           INTEGER LENGTH, FUNCNO, SAVER0;  R3 := R3-37;  FUNCNO := R3;
20 08C0  07 0A2C  3268            COMMENT  (FUNCNO DIV 2) = TYPE  ;
20 08C0  07 0A2C  3269           IF R3 = 6 OR R3 = 7 THEN R0 := 19 ELSE R0 := 11; LENGTH := R0;
20 08DC  07 0A2C  3270           CONVERT;   STFUNCTION;   R0 := FUNCNO; IF R0 <= 3 THEN
20 09C0  07 0A2C  3271           BEGIN  R0 := #18030000; EMIT;
20 0910  07 0A2C  3272           END;
20 0910  07 0A2C  3273           R0 := #FFFF; R(4) := R0; GENREG;
20 0924  07 0A2C  3274           R3 := R0 SHRL 8; LSTACKM4(R5) := R3;
20 092E  07 0A2C  3275           R0 := R0 OR LAA OR NEXTADDR; EMIT;
20 0942  07 0A2C  3276           R0 := R0-R0; R(4) := R0; R10 := LENGTH + 1; INCRADDR;
20 095C  07 0A2C  3277           R3 := FUNCNO SHLL 24 OR #B0000000;  R0 := @WRITEEDIT;
20 096C  07 0A2C  3278           R0 := NEG R0; EMITCALL; R7 := LENGTH SHLL 16 OR #87000000;
20 0986  07 0A2C  3279        END ELSE
20 0986  07 0A2C  3280        BEGIN  COMMENT TIME (R3 = 45);
20 098A  07 0A2C  3281           LOADREG;
20 0996  07 0A2C  3282           R0 := LSTACKM4(R5) AND MASK5 SHRL 4 OR #18000000;  EMIT;
```

```
20 09B2  07 0A2C   3283            R0 := @GTIME; R0 := NEG R0;  EMITCALL;
20 09C4  07 0A2C   3284            R0 := LSTACKM4(R5) AND MASK5 OR #18000000;    EMIT;
20 09DC  07 0A2C   3285            R7 := #81000000;
20 09E0  07 0A2C   3286          END;
20 09E0  07 0A2C   3287          R1 := SAVER1;
20 09E4  07 0A2C   3288        END;

                  SEGMENT 20  NAME = SEG#20      LENGTH = 0AC0  BASE REG = 15

11 0BD8  07 0A2C   3289
11 0BD8  07 0A2C   3290    SEGMENT PROCEDURE LEVEL2(R1);
11 0BD8  07 0A2C   3291     BEGIN
21 0000  07 0A2C   3292
21 0000  07 0A2C   3293     SEGMENT PROCEDURE  NUMERICALASSIGN(R1);
21 0000  07 0A2C   3294        COMMENT  PROCESSES NUMERICAL ASSIGNMENT (A:=, A:=2 - ARG2);
21 0000  07 0A2C   3295        BEGIN  LOGICAL SAVER1;  SAVER1 := R1;
22 0004  07 0A30   3296           R3 := TREE(R6); IF R3 < 0 THEN
22 000E  07 0A30   3297           BEGIN  COMMENT ON LEFT, EXPRESSION ON RIGHT PROCESSED;
22 000E  07 0A30   3298              IF R7 >= 0 THEN
22 0014  07 0A30   3299              BEGIN  COMMENT TERMINAL ON LEFT;
22 0014  07 0A30   3300                 GETTYPE;  R6 := R3;  R0 := R7 SHRL 24; IF R0 = CONID THEN
22 0030  07 0A30   3301                 BEGIN  COMMENT INITIAL FOR VALUE, FORCE TO R2;
22 0030  07 0A30   3302                    R3 := 1;  CORRECTREG;  R0 := LSTACKM4(R5);
22 0044  07 0A30   3303                    R3 := IDLOC1(R7) SHLL 12 + IDLOC2(R7);
22 0050  07 0A30   3304                    LSTACKM4(R5) := R3;  R5 := R5 + 4;  LSTACKM4(R5) := R0;
22 005C  07 0A30   3305                 END ELSE
22 005C  07 0A30   3306                 BEGIN  GETADDRESS;  R0 := LSTACKM4(R5); IF R0 > 0 THEN
22 0076  07 0A30   3307                    BEGIN  COMMENT EXPR DUMPED TO PROCESS LEFT SIDE;
22 0076  07 0A30   3308                       R7 := R6 SHLL 24 OR SIGN;  LOADREG;
22 008C  07 0A30   3309                       R0 := LSTACKM4(R5);
22 0090  07 0A30   3310                    END;
22 0090  07 0A30   3311                    ASSEMBLE;  IF R6 = 9 THEN BEGIN R10 := 4; REFSTOR; END;
22 00B4  07 0A30   3312                 END;
22 00B4  07 0A30   3313              END ELSE
22 00B4  07 0A30   3314              BEGIN  COMMENT NON-TERMINAL ON LEFT;
22 00B8  07 0A30   3315                 R6 := R7 SHRL 24 AND #7F; RESET(ARGFLAG);
22 00C6  07 0A30   3316                 R0 := LSTACKM4(R5-4); IF R0 > 0 THEN
22 00D0  07 0A30   3317                 BEGIN  COMMENT EXPR DUMPED TO PROCESS LEFT SIDE;
22 00D0  07 0A30   3318                    R5 := R5 - 4; R7 := R6 SHLL 24 OR SIGN; LOADREG;
22 00EA  07 0A30   3319                    R5 := R5 + 4; R0 := LSTACKM4(R5-4);  SET(ARGFLAG);
22 00F6  07 0A30   3320                 END;
22 00F6  07 0A30   3321                 R3 := LSTACKM4(R5);  ASSEMBLE;
22 0106  07 0A30   3322                 R3 := LSTACKM4(R5);  R6 := 1;  RELEASE; R5 := R5 - 4;
22 011E  07 0A30   3323                 R6 := R7 SHRL 24 AND #7F;
22 0128  07 0A30   3324                 IF ARGFLAG THEN IF R6 = 1 OR R6 >= 8 THEN ADJSTACKS;
22 014C  07 0A30   3325              END;
22 014C  07 0A30   3326              IF ¬CEQ2 THEN
22 0154  07 0A30   3327              BEGIN  COMMENT RELEASE EXPR REGISTER;
22 0154  07 0A30   3328                 R3 := LSTACKM4(R5);  RELEASE;  R5 := R5 - 4;
22 0168  07 0A30   3329              END;
22 0168  07 0A30   3330           END ELSE
22 0168  07 0A30   3331           BEGIN  COMMENT  ON RIGHT, LEFT SIDE PROCESSED;
22 016C  07 0A30   3332              CONVERT;  LOADREG;  R6 := R7 SHRL 24 AND #7F;
22 018C  07 0A30   3333              R3 := LSTACKM4(R5-4);  R1 := CLN SHLL 12; IF R3 > R1 THEN
22 019E  07 0A30   3334              BEGIN COMMENT HAD TO DUMP ADDRESS FROM LEFT;
22 019E  07 0A30   3335                 R5 := R5 - 4;  R7 := #81000000;  LOADREG;  R5 := R5 + 4;
22 01B6  07 0A30   3336                 R3 := LSTACKM8(R5) SHLL 1 SHRL 9;
22 01C2  07 0A30   3337              END ELSE
```

```
22 01C2  07 0A30   3338            BEGIN COMMENT DID NOT DUMP ADDRESS FROM LEFT;
22 01C6  07 0A30   3339               R3 := LSTACKM8(R5);
22 01CA  07 0A30   3340            END;
22 01CA  07 0A30   3341            R0 := LSTACKM4(R5);  ASSEMBLE;
22 01DA  07 0A30   3342            R3 := LSTACKM4(R5);  RELEASE;  R3 := LSTACKM8(R5);  R10 := R6;
22 01EE  07 0A30   3343            R6 := 1;   RELEASE;    R6 := R10;       R5 := R5 - 8;
22 0206  07 0A30   3344         END;
22 0206  07 0A30   3345         R7 := R6 SHLL 24 OR SIGN;   R1 := SAVER1;
22 0214  07 0A30   3346      END;

                  SEGMENT 22  NAME = SEG#22      LENGTH = 0248  BASE REG = 15

21 0000  07 0A30   3347
21 0000  07 0A30   3348   PROCEDURE LCOLEQARG2(R1);
21 0000  07 0A30   3349      BEGIN  LOGICAL SAVER1; BYTE ARGFLAG;  SAVER1 := R1; RESET(ARGFLAG);
21 000C  07 0A35   3350         IF R7 >= 0 THEN
21 0012  07 0A35   3351         BEGIN  R6 := R6 AND MASK;  R3 := TREE(R6);  IF R3 < 0 THEN
21 0020  07 0A35   3352            BEGIN GETADDRESS;  IF R7 < 0 THEN
21 0032  07 0A35   3353               BEGIN  R7 := #86000000; SET(R3FLAG); LOADREG; END;
21 0046  07 0A35   3354               R0 := LSTACKM4(R5) AND MASK5 OR STCC OR R3;  EMIT;
21 0060  07 0A35   3355               TEST(CEQ2);  IF ¬= THEN
21 0068  07 0A35   3356               BEGIN  R3 := LSTACKM4(R5);  R6 := 1;  RELEASE;  R5:=R5-4;
21 0080  07 0A35   3357               END;
21 008C  07 0A35   3358            END ELSE
21 0080  07 0A35   3359            BEGIN  LOGICAL SAVER3;  R3 := LSTACK(R5);  SAVER3 := R3;
21 008C  07 0A3C   3360               GETADDRESS;  R1 := SAVER3;  SAVER3 := R3;
21 00A0  07 0A3C   3361               IF R7 < 0 AND R1 ¬= LSTACKM4(R5) THEN
21 00AE  07 0A3C   3362               BEGIN  SET(R3FLAG);  LOADREG;
21 00BE  07 0A3C   3363                  R1 := LSTACKM4(R5) XOR SIGN SHRL 8;  LSTACKM4(R5) := R1;
21 00CA  07 0A3C   3364               END;
21 00CE  07 0A3C   3365               GENREG;
21 00DA  07 0A3C   3366               R3  := SAVER3;  R0 := R0 OR R3 OR ICC;  EMIT;
21 00F0  07 0A3C   3367               R0 := LSTACKM4(R5) SHLL 1 SHRL 1 OR LSTACKM8(R5) OR STCC;
21 0104  07 0A3C   3368               EMIT;  R6 := 1;  R3 := LSTACKM4(R5);  R0 := LSTACKM8(R5);
21 011C  07 0A3C   3369               RELEASE;  ZRELEASE;  R5 := R5 - 8;
21 0138  07 0A3C   3370            END;
21 0138  07 0A3C   3371         END ELSE
21 0138  07 0A3C   3372         BEGIN
21 013C  07 0A3C   3373            R3 := R7 SHLL 8 SHRL 24;  R6 := R6 AND MASK;  R0 := TREE(R6);
21 014E  07 0A3C   3374            IF R3 = 1 THEN
21 0156  07 0A3C   3375            BEGIN  IF R0 < 0 THEN
21 015C  07 0A3C   3376               BEGIN  COMMENT  ARRAY ON LEFT;
21 015C  07 0A3C   3377                  R1 := LSTACKM8(R5);  IF R1 > 0 THEN
21 0166  07 0A3C   3378                  BEGIN  COMMENT LOGICAL VALUE HAS BEEN DUMPED;
21 0166  07 0A3C   3379                     R5 := R5-4; R7 := #81000000; LOADREG; R5 := R5+4;
21 017E  07 0A3C   3380                     SET(ARGFLAG);
21 0182  07 0A3C   3381                  END;
21 0182  07 0A3C   3382                  R0 := LSTACKM8(R5) AND MASK5 OR LSTACKM4(R5) OR STCC;
21 0192  07 0A3C   3383                  EMIT; R3 := LSTACKM4(R5); R6 := 1; RELEASE; R5 := R5-4;
21 0186  07 0A3C   3384                  IF ARGFLAG THEN ADJSTACKS;
21 01CA  07 0A3C   3385                  IF ¬CEQ2 THEN
21 01D2  07 0A3C   3386                  BEGIN  R0 := LSTACKM4(R5); ZRELEASE; R5 := R5-4;
21 01E6  07 0A3C   3387                  END;
21 01E6  07 0A3C   3388               END ELSE
21 01E6  07 0A3C   3389               BEGIN  COMMENT  ARRAY OR PROCEDURE ON RIGHT;
21 01EA  07 0A3C   3390                  R3 := LSTACKM4(R5) SHRL 8 AND MASK5;
21 01F6  07 0A3C   3391                  R0 := LSTACKM4(R5) OR R3 OR ICC;  EMIT;
21 020C  07 0A3C   3392                  R0 := LSTACKM8(R5) OR R3 OR STCC;  EMIT;
```

```
21 0222  07 0A3C  3393                R3 := LSTACKM4(R5);  RELEASE;  R3 := LSTACKM8(R5);
21 0236  07 0A3C  3394                R6 := 1;  RELEASE;  R5 := R5 - 8;
21 024A  07 0A3C  3395             END;
21 024A  07 0A3C  3396          END ELSE
21 024A  07 0A3C  3397          BEGIN IF R0 < 0 THEN
21 0254  07 0A3C  3398             BEGIN  COMMENT  ON  LEFT;
21 0254  07 0A3C  3399                R3 := LOGPOINTER;  R0 := 0;  IC(R0,LOGSTACK(R3+1));
21 0260  07 0A3C  3400                LOGBRANCH;  LINKBRANCHES;  R3 := LOGPOINTER;
21 027C  07 0A3C  3401                IC(R3,LOGSTACK(R3));  R3 := R3 AND MASK9;
21 0284  07 0A3C  3402                CASE R3 OF
21 0284  07 0A3C  3403                BEGIN
21 0284  07 0A3C  3404                   BEGIN  R0 := LSTACKM4(R5) OR MVII;  EMIT;
21 02A0  07 0A3C  3405                      R3 := INSCOUNTER;  FIXUP;  R0 := LSTACKM4(R5)
21 02B0  07 0A3C  3406                      OR MVII OR TRUEE; EMIT;  R3 := INSCOUNTER; FIXUP;
21 02D8  07 0A3C  3407                   END;
21 02D8  07 0A3C  3408                   BEGIN  R0 := LSTACKM4(R5) OR MVII OR TRUEE; EMIT;
21 02F4  07 0A3C  3409                      R3 := INSCOUNTER;  FIXUP;  R0 := LSTACKM4(R5)
21 0304  07 0A3C  3410                      OR MVII;  EMIT;  R3 := INSCOUNTER;  FIXUP;
21 0328  07 0A3C  3411                   END;
21 0328  07 0A3C  3412                END;
21 0334  07 0A3C  3413                R3 := LSTACKM4(R5);  RELEASE;  R5 := R5 - 4;
21 0348  07 0A3C  3414             END ELSE
21 0348  07 0A3C  3415             BEGIN  COMMENT  ON RIGHT;
21 034C  07 0A3C  3416                IF R3 ¬= 2 THEN EVALUATELOG;
21 0360  07 0A3C  3417                R0 := LSTACKM8(R5);  R3 := R0 AND #F000 SHRL 12;
21 036E  07 0A3C  3418                IF R3 >= CLN THEN
21 0376  07 0A3C  3419                BEGIN  R5 := R5 - 4;  R7 := #81000000;  LOADREG;
21 038A  07 0A3C  3420                   R5 := R5 + 4;  R0 := LSTACKM8(R5) XOR SIGN SHRL 8;
21 039A  07 0A3C  3421                END;
21 039A  07 0A3C  3422                R0 := R0 OR LSTACKM4(R5) XOR SIGN OR STCC;  EMIT;
21 03B2  07 0A3C  3423                R3 := LSTACKM4(R5);  RELEASE;  R3 := LSTACKM8(R5);
21 03C6  07 0A3C  3424                RELEASE;  R5 := R5 - 8;
21 03D6  07 0A3C  3425             END;
21 03D6  07 0A3C  3426          END;
21 03D6  07 0A3C  3427       END;
21 03D6  07 0A3C  3428       R1 := SAVER1;
21 03EA  07 0A3C  3429    END;
21 03EC  07 0A3C  3430
21 03EC  07 0A3C  3431 SEGMENT PROCEDURE  SCOLEQARG2(R1);
21 03EC  07 0A3C  3432    BEGIN  LOGICAL SAVER1,SAVEADD,SR0,TREEN;  SAVER1 := R1;
23 0004  07 0A4C  3433       R6 := R6 AND MASK2; TREEN := R6;
23 000C  07 0A4C  3434       IF R7 >= 0 THEN
23 0012  07 0A4C  3435       BEGIN COMMENT TERMINALNODE;
23 0012  07 0A4C  3436          R0 := TREE(R6);  IF R0 < 0 THEN
23 001C  07 0A4C  3437          BEGIN COMMENT ON LEFT;
23 001C  07 0A4C  3438             R0 := R0 AND TYPEMASK;  GETADDRESS;  SAVEADD := R3;
23 0030  07 0A4C  3439             R1 := STACKP4(R8);  IF R7 < 0 AND R1 ¬= LSTACKM4(R5) THEN
23 0042  07 0A4C  3440             BEGIN  SR0:=R0; R7 := #87000000; SET(R3FLAG); LOADREG;
23 005A  07 0A4C  3441                R0 := LSTACKM4(R5) XOR SIGN SHRL 8; LSTACKM4(R5) := R0;
23 006A  07 0A4C  3442                R0 := SR0;
23 006E  07 0A4C  3443             END;
23 006E  07 0A4C  3444             R0 := R0 OR R3 OR #D2000000;  R3 := LSTACKM4(R5) SHLL 16;
23 007C  07 0A4C  3445             EMIT;
23 0088  07 0A4C  3446          END ELSE
23 0088  07 0A4C  3447          BEGIN  R0 := R0 AND TYPEMASK;  GETADDRESS;  R3 := R3 SHLL 16;
23 00A0  07 0A4C  3448             R1 := STACKP4(R8);  IF R7 < 0 AND R1 ¬= LSTACKM4(R5) THEN
23 00B2  07 0A4C  3449             BEGIN  SR0 := R0;  R7 := #87000000; SET(R3FLAG);  LOADREG;
23 00CA  07 0A4C  3450                R0 := LSTACKM4(R5) XOR SIGN SHRL 8;  LSTACKM4(R5) := R0;
```

```
23 00D6  07 0A4C  3451                  R0 := SR0;
23 00DE  07 0A4C  3452               END;
23 00DE  07 0A4C  3453               R0 := R0 OR LSTACKM4(R5) OR #D2000000;  EMIT;
23 00F2  07 0A4C  3454            END;
23 00F2  07 0A4C  3455            SR0 := R0;
23 00F6  07 0A4C  3456            R6 := 1;  R3 := LSTACKM4(R5);  RELEASE;  R5 := R5 - 4;
23 010E  07 0A4C  3457            TEST(CEQ2);  IF  =  THEN
23 0116  07 0A4C  3458            BEGIN  R3 := SAVEADD;  IF R3 = #3000 THEN
23 0122  07 0A4C  3459               BEGIN  GENREG;  R0 := R0 OR LAA OR #3000;  EMIT;
23 0142  07 0A4C  3460               END ELSE
23 0142  07 0A4C  3461               BEGIN  LSTACK(R5) := R3;  R5 := R5 + 4;
23 014E  07 0A4C  3462               END;
23 014E  07 0A4C  3463            END;
23 014E  07 0A4C  3464         END ELSE
23 014E  07 0A4C  3465         BEGIN COMMENT NON-TERMINALNODE;
23 0152  07 0A4C  3466            TEST(RECORDFLAG);  IF ¬= THEN
23 015A  07 0A4C  3467            BEGIN
23 015A  07 0A4C  3468               R0 := LSTACKM8(R5);  IF R0 ¬= STACKP4(R8) THEN
23 0166  07 0A4C  3469               BEGIN LOGICAL SAVE;  R5 := R5 - 8;  SAVE := R0;  GENREG;
23 017A  07 0A50  3470                  R0 := R0 OR #58000000 OR SAVE;  EMIT;  R5 := R5 + 4;
23 0192  07 0A50  3471                  R0 := LSTACKM8(R5) SHLL 1 SHRL 9;  LSTACKM8(R5) := R0;
23 01A2  07 0A50  3472               END;
23 01A2  07 0A50  3473            END;
23 01A2  07 0A50  3474            R0 := TREE(R6);  IF R0 < 0 THEN
23 01AC  07 0A50  3475            BEGIN COMMENT ON LEFT;
23 01AC  07 0A50  3476               R3 := LSTACKM4(R5);  SAVEADD := R3;
23 01B4  07 0A50  3477               R0 := R0 AND TYPEMASK OR LSTACKM4(R5) OR #D2000000;
23 01C0  07 0A50  3478               R3 := LSTACKM8(R5) SHLL 16;  EMIT;
23 01D4  07 0A50  3479            END ELSE
23 01D4  07 0A50  3480            BEGIN COMMENT ON RIGHT;
23 01D8  07 0A50  3481               R3 := LSTACKM8(R5);  SAVEADD := R3;
23 01E0  07 0A50  3482               R0 := R0 AND TYPEMASK OR LSTACKM8(R5) OR #D2000000;
23 01EC  07 0A50  3483               R3 := LSTACKM4(R5) SHLL 16;  EMIT;
23 0200  07 0A50  3484            END;
23 0200  07 0A50  3485            SR0 := R0;
23 0204  07 0A50  3486            R6 := 1;  TEST(CEQ2);  IF ¬=  THEN
23 0210  07 0A50  3487            BEGIN  R3 := LSTACKM4(R5);  RELEASE;
23 0220  07 0A50  3488            END;
23 0220  07 0A50  3489            R5 := R5 - 4;  R3 := LSTACKM4(R5);  RELEASE;  R5 := R5 - 4;
23 0238  07 0A50  3490            TEST(CEQ2);  IF  =  THEN
23 0240  07 0A50  3491            BEGIN  R3 := SAVEADD;  LSTACK(R5) := R3;  R10 := R3 SHRL 12;
23 024E  07 0A50  3492               IF R3 > 0 AND R10 < CLN THEN RSTACKM4(R2) := R5;
23 0260  07 0A50  3493               R5 := R5 + 4;
23 0264  07 0A50  3494            END;
23 0264  07 0A50  3495         END;
23 0264  07 0A50  3496         R0 := SR0; R3 := TREEN;
23 026C  07 0A50  3497         R3 := TREE(R3) AND MASK;  R1 := R0 AND TYPEMASK;
23 027A  07 0A50  3498         R3 := TREE(R3) AND TYPEMASK - R1;
23 0284  07 0A50  3499         COMMENT  R3 HAS ACTUAL NUMBER OF BLANKS NEEDED;
23 0284  07 0A50  3500         IF R3 > 0 THEN
23 028A  07 0A50  3501         BEGIN  R1 := R1 SHRL 16 + 1;  R0 := R0 AND MASK + R1;
23 0298  07 0A50  3502            R0 := R0 OR #92400000;  EMIT;  R1 := R3 SHRL 16 - 1;
23 02B2  07 0A50  3503            IF R1 = 0 THEN NULL ELSE
23 02B8  07 0A50  3504            IF R1 = 1 THEN
23 02C4  07 0A50  3505            BEGIN  R0 := R0 AND MASK + 1 OR #92400000;  EMIT;
23 02DC  07 0A50  3506            END ELSE
23 02DC  07 0A50  3507            BEGIN  R0 := R0 AND MASK;  R3 := R0 SHLL 16;  R0 := R0 + 1;
23 02EE  07 0A50  3508               R1 := R1 - 1 SHLL 16;
```

```
23 02F6  07 0A50   3509              R0 := R0 OR R1 OR #D2000000;  EMIT;
23 0308  07 0A50   3510           END;
23 0308  07 0A50   3511        END;
23 0308  07 0A50   3512        R1 := SAVER1;
23 030C  07 0A50   3513     END;

                  SEGMENT 23  NAME = SEG#23      LENGTH = 0340  BASE REG = 15

21 03DC  07 0A50   3514
21 03DC  07 0A50   3515     PROCEDURE RCOLEQARG2(R1);
21 03DC  07 0A50   3516        BEGIN  LOGICAL SAVER1;  SAVER1 := R1;
21 03E0  07 0A54   3517           R3 := TREE(R6);  IF R3 < 0  THEN NUMERICALASSIGN ELSE
21 03F4  07 0A54   3518           BEGIN  IF R7 >= 0 THEN
21 03FE  07 0A54   3519              BEGIN R3 := R7 SHRL 24;  IF R3 = RCCLID  THEN
21 040C  07 0A54   3520                 BEGIN  RECORDALLOCATE;  R7 := #89000000;
21 041C  07 0A54   3521                 END;
21 041C  07 0A54   3522              END;
21 041C  07 0A54   3523              NUMERICALASSIGN;
21 0426  07 0A54   3524           END;
21 0426  07 0A54   3525           RESETRECORD;  R5 := R5 + 4;  RESETRECORD;  R5 := R5 - 4;
21 0446  07 0A54   3526           TEST(CEQ2);  IF ¬= THEN R7 := SIGN ELSE R7 := #89000000;
21 045A  07 0A54   3527           R1 := SAVER1;
21 045E  07 0A54   3528        END;
21 0460  07 0A54   3529
21 0460  07 0A54   3530     SEGMENT PROCEDURE  RECDESIGASSIGN(R1);
21 0460  07 0A54   3531        BEGIN
24 0000  07 0A54   3532           LOGICAL SAVER0,SAVER1,SAVER3,SAVER6,SAVERA;
24 0000  07 0A68   3533           SAVER1 := R1;  SAVER6 := R6;  SAVERA := R10;
24 000C  07 0A68   3534           IF R7 < 0 AND R7 ¬= #86000000 THEN
24 001A  07 0A68   3535           BEGIN  R3 := LSTACKM8(R5);  IF R3 >= 0 THEN
24 0024  07 0A68   3536              BEGIN  SAVER3 := R3;  R5 := R5 - 8;  GENREG;
24 0038  07 0A68   3537                 R0 := R0 OR SAVER3 OR #58000000;  EMIT;
24 004C  07 0A68   3538                 R3 := R7 SHRL 24 AND #F; R0 := LSTACK(R5);
24 005A  07 0A68   3539                 IF R3 = 1 OR R3 > 5 OR R0 > 0 THEN
24 0070  07 0A68   3540                 BEGIN  R3 := RSTACK(R2-8);  R0 := RSTACK(R2-4);
24 0078  07 0A68   3541                    RSTACK(R2-8) := R0;  RSTACK(R2-4) := R3;
24 0080  07 0A68   3542                 END;
24 0080  07 0A68   3543              END ELSE R5 := R5 - 4;
24 0088  07 0A68   3544           END ELSE
24 0088  07 0A68   3545           BEGIN  R3 :=  LSTACKM4(R5);   IF R3 >= 0 THEN
24 0096  07 0A68   3546              BEGIN  SAVER3 := R3;  R5 := R5 - 4;  GENREG;
24 00AA  07 0A68   3547                 R0 := R0 OR SAVER3 OR #58000000;  EMIT;
24 00BE  07 0A68   3548              END;
24 00BE  07 0A68   3549           END;
24 00BE  07 0A68   3550           R3 := LSTACK(R5);  LSTACK(R5+4) := R3;
24 00C6  07 0A68   3551           R10 := R7;  R7 := ID SHLL 24 OR STACKP4(R8);
24 00D4  07 0A68   3552           GETADDRESS;  R7 := R10;  R3 := R3 AND #FFF;    SAVER3 := R3;
24 00EA  07 0A68   3553           GENREG;  R3 := LSTACKM4(R5) SHLL 1 SHRL 9;  LSTACKM4(R5) := R3;
24 0106  07 0A68   3554           R3 := LSTACKM8(R5) XOR SIGN SHRL 8;
24 0112  07 0A68   3555           R0 := R0 OR #41000000 OR SAVER3 OR R3;  EMIT;  SET(FIELDOP);
24 012C  07 0A68   3556           R10 := STACKP4(R8);  R3 := 0;  IC(R3,NAMETABLE(R10+9));
24 0138  07 0A68   3557           IF R3 = 7 THEN
24 0140  07 0A68   3558           BEGIN  R1 := SAVEL  AND MASK;  R1 := TREE(R1) AND MASK;
24 0150  07 0A68   3559              R0 := NAMETABLE(R10+4) SHRL 16;
24 0158  07 0A68   3560              STC(R0,TREE(R1+1));
24 015C  07 0A68   3561           END;
24 015C  07 0A68   3562           R7 := R7 AND #FFFFFFFE;  IF R7 >0 THEN
24 0166  07 0A68   3563           BEGIN  R10 := R6;  R6 := R3 SHLL 24 OR SIGN OR R10;
```

```
24 0174  07 0A68   3564          END ELSE
24 0174  07 0A68   3565          BEGIN  R6 := R6 OR R7;  IF R7 ¬= #86000000 THEN R5 := R5 + 4;
24 0186  07 0A68   3566          END;
24 0186  07 0A68   3567          SET(RECORDFLAG);
24 018A  07 0A68   3568          CASE R3 OF
24 018A  07 0A68   3569          BEGIN
24 018A  07 0A68   3570             NUMERICALASSIGN;  NUMERICALASSIGN;  NUMERICALASSIGN;
24 0188  07 0A68   3571             NUMERICALASSIGN;  NUMERICALASSIGN; LCOLEQARG2;
24 01E4  07 0A68   3572             SCOLEQARG2;   NUMERICALASSIGN;
24 0200  07 0A68   3573             BEGIN  RESET(FIELDOP);  RCOLEQARG2;
24 0214  07 0A68   3574             END;
24 0214  07 0A68   3575          END;
24 023C  07 0A68   3576          R3 := SAVER3;  IF R3 = 0 THEN CHECKMARK;
24 0252  07 0A68   3577          R3 := 0;  R7 := #89000000;
24 025A  07 0A68   3578          RESET(RECORDFLAG);  R1 := SAVER1;
24 0262  07 0A68   3579          R6 := SAVER6 OR R7;
24 0268  07 0A68   3580       END;

                  SEGMENT 24  NAME = SEG#24      LENGTH = 02B8  BASE REG = 15

21 0460  07 0A68   3581
21 0460  07 0A68   3582    PROCEDURE CEQUAL(R1);
21 0460  07 0A68   3583       BEGIN  LOGICAL SAVER1,SAVER3,RELATE; SAVER1 := R1;  SAVER3 := R3;
21 0468  07 0A74   3584          RELATE := R0;  LOADREG;
21 0478  07 0A74   3585          BEGIN  R0 := LSTACKM4(R5);
21 047C  07 0A74   3586             IF R0 < 0 THEN
21 0482  07 0A74   3587             BEGIN  R0 := R0 AND MASK5;  R3 := LSTACKM8(R5);
21 048A  07 0A74   3588                IF R3 < 0 THEN R3 := R3 AND MASK5 OR SIGN;
21 0498  07 0A74   3589                R0 := R0 OR #09000000;
21 049C  07 0A74   3590                R6 := SAVER3 - 2;  ASSEMBLE;  R0 := RELATE;
21 04B4  07 0A74   3591                IF R0 = 7 THEN
21 04BC  07 0A74   3592                BEGIN  R1 := 0;  R3 := LOGPOINTER + 4;  LOGPOINTER := R3;
21 04CC  07 0A74   3593                   LOGSTACK(R3) := R1;
21 04D0  07 0A74   3594                END;
21 04D0  07 0A74   3595                R0 := 7;  LOGBRANCH;  INITIALIZELINK;
21 04EC  07 0A74   3596                R0 := LSTACKM4(R5) SHLL 4 AND MASK5;  R3 := LSTACKM8(R5);
21 04FC  07 0A74   3597                IF R3 < 0 THEN R3 := R3 SHLL 4 AND MASK5 OR SIGN ELSE
21 050E  07 0A74   3598                IF R6 = 2 THEN R3 := R3 + 4 ELSE R3 := R3 + 8;
21 0526  07 0A74   3599                R0 := R0 OR #09000000;  ASSEMBLE;
21 0536  07 0A74   3600                R0 := RELATE;
21 053A  07 0A74   3601                IF R0 = 8 THEN
21 0542  07 0A74   3602                BEGIN  R3 := LOGPOINTER + 4;  R1 := 0;  LOGPOINTER := R3;
21 0552  07 0A74   3603                   LOGSTACK(R3) := R1;
21 0556  07 0A74   3604                END;
21 0556  07 0A74   3605                R0 := R0 OR #100 SHLL 16;  R3 := LOGPOINTER + 4;
21 0566  07 0A74   3606                LOGPOINTER := R3;  LOGSTACK(R3) := R0;
21 056E  07 0A74   3607                R10 := R6;  R6 := 4;  R3 := LSTACKM4(R5);  RELEASE;
21 0584  07 0A74   3608                   R3 := LSTACKM8(R5);  RELEASE;  R6 := R10;  R5:=R5-8;
21 059A  07 0A74   3609             END;
21 059A  07 0A74   3610          END;
21 059A  07 0A74   3611          R7 := #86000000;  R1 := SAVER1;
21 05A2  07 0A74   3612       END;
21 05A4  07 0A74   3613
21 05A4  07 0A74   3614    PROCEDURE ARG2RELATION(R1);
21 05A4  07 0A74   3615       BEGIN  COMMENT  R0 HAS LEFT TO RIGHT COMPARE BRANCH;
21 05A4  07 0A74   3616          LOGICAL SAVER1, RELATE;
21 05A4  07 0A7C   3617          SAVER1 := R1;
21 05A8  07 0A7C   3618          RELATE := R0;  CONVERT;  R6 := R6 SHLL 1 SHRL 25;
```

```
21 05BE  07 0A7C  3619          IF R7 > 0 THEN
21 05C4  07 0A7C  3620          BEGIN  COMMENT TERMINALNODE;
21 05C4  07 0A7C  3621             R7 := R7 SHLL 1 SHRL 1;  GETADDRESS;  IF R7 < 0 THEN LOADREG;
21 05EA  07 0A7C  3622             R5 := R5 - 4;  RESETRECORD;  R5 := R5 + 4;
21 05FE  07 0A7C  3623             R0 := LSTACKM4(R5) OR #09000000;  ASSEMBLE;
21 0612  07 0A7C  3624          END ELSE
21 0612  07 0A7C  3625          BEGIN  COMMENT  NONTERMINAL;
21 0616  07 0A7C  3626             LOADREG;  R0 := LSTACKM4(R5) OR #09000000;
21 062A  07 0A7C  3627             R5 := R5 - 4;  RESETRECORD;  R5 := R5 - 4;  RESETRECORD;
21 064A  07 0A7C  3628             R5 := R5 + 8;
21 064E  07 0A7C  3629             R3 := LSTACKM8(R5);  ASSEMBLE;
21 065E  07 0A7C  3630             R3 := LSTACKM8(R5);  R5 := R5 - 4;  RELEASE;
21 0672  07 0A7C  3631             R3 := LSTACK(R5);  LSTACKM4(R5) := R3;
21 067A  07 0A7C  3632             R0 := RELATE;  IF R0 ¬= 8 THEN IF R0 ¬= 7 THEN
21 068E  07 0A7C  3633             BEGIN R0 := R0 XOR #7;  RELATE := R0;  END;
21 0696  07 0A7C  3634          END;
21 0696  07 0A7C  3635          R3 := LSTACKM4(R5);  R5 := R5 - 4;  RELEASE;
21 06AA  07 0A7C  3636          R3 := LOGPOINTER + 12;  LOGPOINTER := R3;
21 06B6  07 0A7C  3637          R1 := 0;
21 06BA  07 0A7C  3638          R0 := RELATE OR #100 SHLL 16;
21 06C6  07 0A7C  3639          LOGSTACK(R3) := R0;  LOGSTACK(R3-4) := R1;
21 06CE  07 0A7C  3640          LOGSTACK(R3-8) := R1;
21 06D2  07 0A7C  3641          R7 := #86000000;  R1 := SAVER1;
21 06DA  07 0A7C  3642       END;
21 06DC  07 0A7C  3643
21 06DC  07 0A7C  3644    PROCEDURE LEQUAL(R1);
21 06DC  07 0A7C  3645    BEGIN  LOGICAL SAVER1,SAVER6;  SAVER1 := R1;  SAVER6 := R6;
21 06E4  07 0A84  3646       IF R7 < 0 THEN
21 06EA  07 0A84  3647       BEGIN  R3 := R7 SHLL 8 SHRL 24;
21 06F4  07 0A84  3648          IF R3 = 1 THEN
21 06FC  07 0A84  3649          BEGIN R0 := LSTACKM4(R5);  R3 := R0 AND #F000 SHLL 8;  R10 := R3;
21 070A  07 0A84  3650             R0 := R0 OR R3;  R3 := R3 OR SIGN;
21 0712  07 0A84  3651             LSTACKM4(R5) := R3;  R0 := R0 OR #43000000;  EMIT;
21 0726  07 0A84  3652             R1 := CPTBASE; R3 := CONSPTTAB(R1) AND MASK OR #E000;
21 0736  07 0A84  3653             R0 := R10 OR #04000000;
21 073C  07 0A84  3654             R6 := 1;  ASSEMBLE ;
21 074C  07 0A84  3655          END ELSE EVALUATELOG;
21 075C  07 0A84  3656       END ELSE
21 075C  07 0A84  3657       BEGIN  LOGICAL SAVEADD;  GETADDRESS;  SAVEADD := R3;
21 0770  07 0A88  3658          GENREG;  R0 := R0 OR SAVEADD OR ICC;  EMIT;
21 0790  07 0A88  3659          R0 := LSTACKM4(R5) XOR SIGN OR #04000000;
21 079C  07 0A88  3660          R1:=CPTBASE; R3:=CONSPTTAB(R1) AND MASK OR #E000; R6:=1;ASSEMBLE;
21 07B0  07 0A88  3661       END;
21 07BC  07 0A88  3662       R1 := SAVER1;  R6 := SAVER6;
21 07C4  07 0A88  3663    END;
21 07C6  07 0A88  3664
21 07C6  07 0A88  3665    SEGMENT PROCEDURE  ARG2EQUALITIES(R1);
21 07C6  07 0A88  3666       BEGIN  LOGICAL SAVER1,RELATE,SAVEADD;
25 0000  07 0A94  3667          LOGICAL SR6;
25 0000  07 0A98  3668          SAVER1 := R1;  SR6 := R6;
25 0008  07 0A98  3669          IF R0 ¬= 8 AND R0 ¬= 7 THEN
25 0018  07 0A98  3670          BEGIN  R3 := TREE(R6);  IF R3 <= 0 THEN R0 := R0 XOR #7;
25 0026  07 0A98  3671          END;
25 0026  07 0A98  3672          RELATE := R0;
25 002A  07 0A98  3673          R3 := R7 SHLL 1 SHRL 25; IF R7>0 THEN IF R3¬=RCCLID THEN
25 0042  07 0A98  3674          BEGIN  GETTYPE;  IF R3 < 6 THEN
25 0056  07 0A98  3675             BEGIN CONVERT;
25 0060  07 0A98  3676                IF R7 < 0 THEN R3 := R7 SHLL 1 SHRL 25 ELSE GETTYPE;
```

```
25 0080  07 0A98  3677          END;
25 0080  07 0A98  3678        END ELSE R3 := 9;
25 0088  07 0A98  3679        R0 := RELATE;
25 008C  07 0A98  3680        IF R3 < 4 THEN ARG2RELATION ELSE
25 00A0  07 0A98  3681        IF R3 < 6 THEN CEQUAL ELSE
25 00B8  07 0A98  3682        BEGIN  R3 := R3 - 5;
25 00C0  07 0A98  3683        CASE  R3  OF
25 00C0  07 0A98  3684        BEGIN
25 00C0  07 0A98  3685           BEGIN  COMMENT  = LOGICAL ARG2;
25 00C8  07 0A98  3686              LEQUAL;  R0 := LSTACKM4(R5) OR #09000000;
25 00DC  07 0A98  3687              R3 := LSTACKM8(R5);  R6 := 1;  ASSEMBLE;  R6 := 1;
25 00F4  07 0A98  3688              R3 := LSTACKM4(R5);  RELEASE;  R3 := LSTACKM8(R5);
25 01C8  07 0A98  3689              RELEASE;  R5 := R5 - 8;  R7 := #86000000;
25 011C  07 0A98  3690              R3 := LOGPOINTER + 12;  LOGPOINTER := R3;
25 0128  07 0A98  3691              R1 := 0;  R0 := RELATE OR #100 SHLL 16;
25 0138  07 0A98  3692              LOGSTACK(R3) := R0;  LOGSTACK(R3-4) := R1;
25 0140  07 0A98  3693              LOGSTACK(R3-8) := R1;
25 0144  07 0A98  3694           END;
25 0144  07 0A98  3695           BEGIN  COMMENT  = STRING ARG2;
25 0148  07 0A98  3696                 R6 := R7;
25 014A  07 0A98  3697              IF R7 < 0 THEN
25 015C  07 0A98  3698              BEGIN  R0 := LSTACKM8(R5);  IF R0 ¬= STACKP4(R8) THEN
25 015C  07 0A98  3699              BEGIN  LOGICAL SAVE;  R5 := R5 - 8;  SAVE := R0;  GENREG;
25 0170  07 0A9C  3700                 R0 := R0 OR #58000000 OR SAVE;  EMIT;
25 0184  07 0A9C  3701                 R5 := R5+4;  R0 := LSTACKM8(R5);
25 018C  07 0A9C  3702                 R0 := R0 XOR SIGN SHRL 8;  LSTACKM8(R5) := R0;
25 0198  07 0A9C  3703              END;
25 0198  07 0A9C  3704              R3 := LSTACKM4(R5);  RELEASE;  R5 := R5-4;
25 01AC  07 0A9C  3705              R3 := R3 SHLL 16;
25 01B0  07 0A9C  3706              END ELSE
25 01B0  07 0A9C  3707              BEGIN  R1 := LSTACKM4(R5);
25 01B8  07 0A9C  3708                 SAVEADD := R1;  GETADDRESS;  R1 := SAVEADD;
25 01CC  07 0A9C  3709                 IF R7 < 0 AND R1 ¬= LSTACKM4(R5) THEN
25 01CA  07 0A9C  3710                 BEGIN  SET(R3FLAG);  LOADREG;
25 01EA  07 0A9C  3711                 R1 := LSTACKM4(R5) XOR SIGN SHRL 8;  LSTACKM4(R5) := R1;
25 01F6  07 0A9C  3712                    R0 := SAVEADD;
25 01FE  07 0A9C  3713                 END;  R3 := R3 SHLL 16;
25 0202  07 0A9C  3714              END;
25 0202  07 0A9C  3715              R6 := SR6;  R0 := TREE(R6) AND TYPEMASK OR LSTACKM4(R5);
25 0212  07 0A9C  3716              R0 := R0 OR #D5000000;  EMIT;
25 0222  07 0A9C  3717                 R3 := LSTACKM4(R5);  R6 := 1;  RELEASE;  R5 := R5 - 4;
25 023A  07 0A9C  3718              R7 := #86000000;
25 023E  07 0A9C  3719              R3 := LOGPOINTER + 12;  LOGPOINTER := R3;
25 024A  07 0A9C  3720              R1 := 0;  R0 := RELATE OR #100 SHLL 16;
25 025A  07 0A9C  3721              LOGSTACK(R3) := R0;  LOGSTACK(R3-4) := R1;
25 0262  07 0A9C  3722              LOGSTACK(R3-8) := R1;
25 0266  07 0A9C  3723           END;
25 0266  07 0A9C  3724           BEGIN  ARG2RELATION;
25 0276  07 0A9C  3725           END;
25 0276  07 0A9C  3726           BEGIN  LOGICAL SAVER6;  SAVER6 := R6;  IF R7 >= 0 THEN
25 0284  07 0AA0  3727              BEGIN  R3 := R7 SHLL 1 SHRL 25;  IF R3 = RCCLID THEN
25 0296  07 0AA0  3728                 BEGIN  RECORDALLOCATE;  R7 := #89000000;
25 02A6  07 0AA0  3729                 END;
25 02A6  07 0AA0  3730              END;
25 02A6  07 0AA0  3731              R0 := RELATE;  R6 := SAVER6;  ARG2RELATION;
25 02BA  07 0AA0  3732           END;
25 02BA  07 0AA0  3733        END;
25 02CE  07 0AA0  3734        END;
```

```
25 02CE  07 0AA0   3735          R1 := SAVER1;
25 02D2  07 0AA0   3736       END;

                  SEGMENT 25  NAME = SEG#25      LENGTH = 0320  BASE REG = 15

21 07C6  07 0AA0   3737
21 07C6  07 0AA0   3738    PROCEDURE IFSTANDEXP(R1);
21 07C6  07 0AAC   3739       BEGIN  LOGICAL SAVER1;   SAVER1 := R1;
21 07CA  07 0AA4   3740          R3 := LOGPOINTER;  R0 := LOGSTACK(R3);
21 07D2  07 0AA4   3741          IF R0 ¬= 0 THEN
21 07D8  07 0AA4   3742          BEGIN  R3 := INSCOUNTER;  FIXUP;
21 07E8  07 0AA4   3743             R3 := LOGPOINTER;  R0 := LOGSTACK(R3+4);
21 07F0  07 0AA4   3744             LOGSTACK(R3) := R0;
21 07F4  07 0AA4   3745          END ELSE
21 07F4  07 0AA4   3746          BEGIN  R3 := R3 - 4; LOGPOINTER := R3;
21 0800  07 0AA4   3747          END;
21 0800  07 0AA4   3748          R1:= SAVER1;
21 0804  07 0AA4   3749       END;
21 0806  07 0AA4   3750
21 0806  07 0AA4   3751    PROCEDURE ANDORARG2(R1);
21 0806  07 0AA4   3752       BEGIN  COMMENT ANDFLAG IS ON IF AND, OFF IF OR;
21 0806  07 0AA4   3753          LOGICAL SAVER1;  SAVER1 := R1;
21 080A  07 0AA8   3754          IF R7 >= 0 THEN
21 0810  07 0AA8   3755          BEGIN  LOGCOMPARE;  TEST(ANDFLAG);
21 0820  07 0AA8   3756             IF = THEN R0 := #0207 ELSE R0 := #0108;
21 0830  07 0AA8   3757             INDICATEBRANCH;
21 083C  07 0AA8   3758          END ELSE
21 083C  07 0AA8   3759          BEGIN  R3 := R7 SHLL 8 SHRL 24;
21 084A  07 0AA8   3760             IF R3 = 1 THEN
21 0852  07 0AA8   3761             BEGIN  R0 := LSTACKM4(R5) OR CLII OR #00010000;  EMIT;
21 086A  07 0AA8   3762                R3 := LSTACKM4(R5);  RELEASE;  R5 := R5 - 4;
21 087E  07 0AA8   3763                TEST(ANDFLAG);
21 0882  07 0AA8   3764                IF  =  THEN R0 := #0207 ELSE R0 := #0108;
21 0892  07 0AA8   3765                INDICATEBRANCH;
21 089E  07 0AA8   3766             END ELSE
21 089E  07 0AA8   3767             BEGIN  R3 := LOGPOINTER;  IC(R10,LOGSTACK(R3));
21 08AA  07 0AA8   3768                R10 := R10 AND #FF;  TEST(ANDFLAG);
21 08B2  07 0AA8   3769                IF  =  THEN R10 := R10 XOR #3;
21 08BA  07 0AA8   3770                CASE  R10 OF
21 08BA  07 0AA8   3771                BEGIN
21 08BA  07 0AA8   3772                   MERGE24;    MERGE34;
21 08DE  07 0AA8   3773                END;
21 08EA  07 0AA8   3774             END;
21 08EA  07 0AA8   3775          END;
21 08EA  07 0AA8   3776          R7 := #86000000;  R1 := SAVER1;
21 08F2  07 0AA8   3777       END;
21 08F4  07 0AA8   3778
21 08F4  07 0AA8   3779    PROCEDURE ARG2FORLIST(R1);  COMMENT FORCE REGISTER, EMIT LINK;
21 08F4  07 0AA8   3780       BEGIN  INTEGER SAVER1;  SAVER1 := R1;
21 08F8  07 0AAC   3781          LOADREG;  R3 := 1; CORRECTREG;  R3 := LSTACKM4(R5); R6 := 1;
21 091C  07 0AAC   3782          RELEASE;  R5 := R5 - 4;    R3 := R8 - SSIZE;
21 0932  07 0AAC   3783          R0 := STACKP4(R8) + 1; STACKP4(R3) := R0;
21 093E  07 0AAC   3784          R0 := INSCOUNTER;  STACKP8(R3) := R0;
21 0946  07 0AAC   3785          R0 := #45100000 OR STACKP8(R8);  EMIT;  R1 := SAVER1;
21 095E  07 0AAC   3786       END;
21 0960  07 0AAC   3787
21 0960  07 0AAC   3788    SEGMENT PROCEDURE  ARG21(R1);
21 0960  07 0AAC   3789       BEGIN  LOGICAL SAVER1;
```

```
26 0000   07 0AB0        3790
26 0000   07 0AB0        3791          BYTE DIVFLAG;  COMMENT SET TO CONTROL MULT/DIV CODE IN PCLL;
26 0000   07 0AB1        3792          PROCEDURE PCLL(R1);  COMMENT GENERATE CALL FOR COMPLEX MULT/DIV;
26 0000   07 0AB1        3793             BEGIN  LOGICAL SAVER1;  SAVER1 := R1;
26 0008   07 0AB8        3794                R0 := @CARITHFN; R0 := NEG R0;
26 000E   07 0AB8        3795                R3 := R6-3; IF DIVFLAG THEN R3 := R3+2;
26 0020   07 0AB8        3796                 COMMENT 1=CMULT, 2=LCMULT, 3=CDIV, 4=LCDIV ;
26 0020   07 0AB8        3797                R3 := R3 SHLL 16;  EMITCALL;
26 0030   07 0AB8        3798                IF DIVFLAG AND R6 = 4 THEN R0 := #40000000 ELSE R0 := 0;
26 004C   07 0AB8        3799                IF ¬DIVFLAG THEN R6 := 5;  PRFLREG;   R1 := SAVER1;
26 0068   07 0AB8        3800             END;
26 006A   07 0AB8        3801
26 006A   07 0AB8        3802          PROCEDURE  SUBDIV(R1); COMMENT DOES SETUP FOR ARG2 SUBTRA
26 006A   07 0AB8        3803             DIVIDE;
26 006A   07 0AB8        3804             BEGIN LOGICAL SAVER1;
26 006A   07 0ABC        3805             PROCEDURE SUBSUBDIV(R1); COMMENT HANDLES COMMON CASE;
26 006A   07 0ABC        3806                BEGIN LOGICAL SAVER1;
26 006E   07 0AC0        3807                   SAVER1 := R1;
26 0072   07 0AC0        3808                   LOADREG; R0 := LSTACKM4(R5); R3 := LSTACKM8(R5);
26 0086   07 0AC0        3809                   IF R3 < 0 THEN RELEASE ELSE ADJSTACKS;
26 00A8   07 0AC0        3810                   LSTACKM8(R5) := R0; R5 := R5-4;
26 00B0   07 0AC0        3811                   R1 := SAVER1;
26 00B4   07 0AC0        3812                END;
26 00B6   07 0AC0        3813                SAVER1 := R1;
26 00BA   07 0AC0        3814                IF R7 > 0 THEN
26 00C0   07 0AC0        3815                BEGIN
26 00C0   07 0AC0        3816                   IF R10 < 0 THEN SUBSUBDIV ELSE
26 00CA   07 0AC0        3817                   BEGIN GETADDRESS; IF R7 < 0 THEN LOADREG;
26 00EC   07 0AC0        3818                      R0 := LSTACKM4(R5);
26 00F0   07 0AC0        3819                   END;
26 00F0   07 0AC0        3820                END ELSE
26 00F0   07 0AC0        3821                BEGIN
26 00F4   07 0AC0        3822                   IF R10 < 0 THEN SUBSUBDIV ELSE
26 00FE   07 0AC0        3823                   BEGIN R5 := R5-4; R0 := LSTACKM4(R5); R3 := LSTACK(R5);
26 010E   07 0AC0        3824                      IF R0 < 0 THEN RELEASE ELSE
26 0120   07 0AC0        3825                      BEGIN LOADREG; RELEASE;
26 013C   07 0AC0        3826                         IF R3 < 0 OR R6 = 1 THEN ADJSTACKS;
26 0156   07 0AC0        3827                         R0 := LSTACKM4(R5);
26 015A   07 0AC0        3828                      END;
26 015A   07 0AC0        3829                   END;
26 015A   07 0AC0        3830                END;
26 015A   07 0AC0        3831                R1 := SAVER1;
26 015E   07 0AC0        3832          END;
26 0160   07 0AC0        3833          SET(ARGFLAG);        SAVER1 := R1;
26 0168   07 0AC0        3834          CASE R3 OF
26 0168   07 0AC0        3835          BEGIN
26 0168   07 0AC0        3836             BEGIN  COMMENT   +  ;                                          1
26 0170   07 0AC0        3837                CONVERT; R6 := R6 SHLL 1 SHRL 25;
26 0182   07 0AC0        3838                IF R7 >= 0 THEN
26 0188   07 0AC0        3839                BEGIN   GETADDRESS;
26 0194   07 0AC0        3840                IF R7 < 0 THEN LOADREG;
26 01A6   07 0AC0        3841                   R0 := LSTACKM4(R5);
26 01AA   07 0AC0        3842                END ELSE
26 01AA   07 0AC0        3843                BEGIN
26 01AE   07 0AC0        3844                   R0 := LSTACKM8(R5); R3 := LSTACKM4(R5);
26 01B6   07 0AC0        3845                   IF R0 >= 0 THEN
26 01BC   07 0AC0        3846                   BEGIN IF R3 > 0 THEN LOADREG;
26 01CE   07 0AC0        3847                      R0 := LSTACKM4(R5); R3 := LSTACKM8(R5);
```

```
26 01D6  07 0AC0  3848                    LSTACKM8(R5) := R0; ADJSTACKS;
26 01E6  07 0AC0  3849                 END ELSE RELEASE; R5 := R5-4;
26 01FA  07 0AC0  3850              END; R0 := R0 OR A; ASSEMBLE;
26 020A  07 0AC0  3851              R7 := R6 SHLL 24 OR SIGN;
26 0214  07 0AC0  3852              CONVERTRESULT;
26 021E  07 0AC0  3853           END;
26 021E  07 0AC0  3854           BEGIN  COMMENT   -  ;                                  2
26 0222  07 0AC0  3855              CONVERT; R10 := R6; R10 := TREE(R10);
26 0232  07 0AC0  3856              R6 := R6 SHLL 1 SHRL 25;
26 023A  07 0AC0  3857              SUBDIV;
26 023E  07 0AC0  3858              R0 := R0 OR S; ASSEMBLE;
26 024E  07 0AC0  3859              R7 := R6 SHLL 24 OR SIGN;
26 0258  07 0AC0  3860              CONVERTRESULT;
26 0262  07 0AC0  3861           END;
26 0262  07 0AC0  3862           BEGIN  COMMENT   *  ;                                  3
26 0266  07 0AC0  3863              CONVERT; R6 := R6 SHLL 1 SHRL 25;
26 0278  07 0AC0  3864              IF R6 <= 3 THEN
26 0280  07 0AC0  3865              BEGIN SET(PAIRFLAG);
26 0284  07 0AC0  3866                 IF R7 >= 0 THEN
26 028A  07 0AC0  3867                 BEGIN GETADDRESS; IF R7 < 0 THEN LOADREG;
26 02A8  07 0AC0  3868                    R0 := LSTACKM4(R5);
26 02AC  07 0AC0  3869                    IF R6 = 1 THEN FORCEPAIR;
26 02C0  07 0AC0  3870                 END ELSE
26 02C0  07 0AC0  3871                 BEGIN R0 := LSTACKM8(R5); R3 := LSTACKM4(R5);
26 02CC  07 0AC0  3872                    IF R0 >= 0 THEN
26 02D2  07 0AC0  3873                    BEGIN IF R3 >= 0 THEN LOADREG; R0 := LSTACKM4(R5);
26 02E4  07 0AC0  3874                       IF R6 = 1 THEN FORCEPAIR;
26 02FC  07 0AC0  3875                       R3 := LSTACKM8(R5); R0 := LSTACKM4(R5);
26 0304  07 0AC0  3876                       LSTACKM8(R5) := R0; ADJSTACKS;
26 0314  07 0AC0  3877                    END ELSE
26 0314  07 0AC0  3878                    BEGIN
26 0318  07 0AC0  3879                       IF R6 = 1 THEN
26 0320  07 0AC0  3880                       BEGIN IF R3 < 0 THEN R3 := LSTACKM4(R5);
26 032A  07 0AC0  3881                          R5 := R5-4; FORCEPAIR; R5 := R5+4;
26 033E  07 0AC0  3882                       END;
26 033E  07 0AC0  3883                       RELEASE;
26 034A  07 0AC0  3884                    END;
26 034A  07 0AC0  3885                    R5 := R5-4;
26 034E  07 0AC0  3886                 END;
26 034E  07 0AC0  3887                 RESET(PAIRFLAG); R0:=R0 AND #FFF0FFFF OR M; ASSEMBLE;
26 035A  07 0AC0  3888              END ELSE
26 0366  07 0AC0  3889              BEGIN  RESET(DIVFLAG);  CONVERT;  IF R7 > 0 THEN
26 037E  07 0AC0  3890                 BEGIN  GETADDRESS;  R0 := #41000000 OR R3; EMIT;
26 039C  07 0AC0  3891                    R7 := R6 SHLL 24 OR SIGN;  STFUNCTION;  PCLL;
26 03B4  07 0AC0  3892                 END ELSE
26 03B4  07 0AC0  3893                 BEGIN R0 := LSTACKM4(R5);  IF R0 > 0 THEN
26 03C2  07 0AC0  3894                    BEGIN  R3 := R0;  R0 := R0 OR #41000000; EMIT;
26 03D4  07 0AC0  3895                       RELEASE; R5 := R5-4;  R7 := R6 SHLL 24 OR SIGN;
26 03EA  07 0AC0  3896                       STFUNCTION;   PCLL;
26 03FC  07 0AC0  3897                    END ELSE
26 03FC  07 0AC0  3898                    BEGIN  R3 := LSTACKM8(R5);
26 0404  07 0AC0  3899                       IF R3 < 0 THEN
26 040A  07 0AC0  3900                       BEGIN  R0 := FSTACK(0);  DUMPPRFLREG;  END;
26 041A  07 0AC0  3901                       R0 := LSTACKM4(R5);  R7 := R6 SHLL 24 OR SIGN;
26 0428  07 0AC0  3902                       IF R0 < 0 THEN R0 := LSTACKM8(R5);
26 0432  07 0AC0  3903                       R0 := R0 OR #41000000; EMIT;  STFUNCTION;
26 044C  07 0AC0  3904                       R5 := R5-4;  PCLL;
26 0454  07 0AC0  3905                    END;
```

```
26 0454  07 0AC0  3906                    END;
26 0454  07 0AC0  3907                 END;
26 0454  07 0AC0  3908                 R7 := R6 SHLL 24 OR SIGN;
26 045E  07 0AC0  3909                 IF R6 = 1 THEN
26 0466  07 0AC0  3910                 BEGIN COMMENT EVEN-ODD PAIR IN LSTACKM4.  UNFLAG ODD IN
26 0466  07 0AC0  3911                    R AND ERASE FROM LSTACKM4.  EMIT  SLDA  ODDREG,32;
26 0466  07 0AC0  3912                    R3 := LSTACKM4(R5) AND MASK4 SHRL 15;  R0:=R0-R0;
26 0474  07 0AC0  3913                    R(R3) := R0;  R0 := LSTACKM4(R5) AND MASK5 OR
26 0480  07 0AC0  3914                    #8F000020;   EMIT;  R0 := R0 AND MASK5 OR SIGN;
26 0498  07 0AC0  3915                    LSTACKM4(R5) := R0;
26 049C  07 0AC0  3916                 END ELSE
26 049C  07 0AC0  3917                 IF R6 = 2 THEN
26 04A8  07 0AC0  3918                 BEGIN
26 04A8  07 0AC0  3919                    R7 := #83000000;
26 04AC  07 0AC0  3920                    R1 := FSTACKM4(R4) AND #BFFFFFFF; FSTACKM4(R4) := R1;
26 04B4  07 0AC0  3921                 END ELSE
26 04B8  07 0AC0  3922                 IF R6 = 4 THEN
26 04C4  07 0AC0  3923                 BEGIN
26 04C4  07 0AC0  3924                    R7 := #85000000;
26 04C8  07 0AC0  3925                    R1 := FSTACKM4(R4) AND #3FFFFFFF; FSTACKM4(R4) := R1;
26 04D0  07 0AC0  3926                 END;
26 04D4  07 0AC0  3927                 CONVERTRESULT;
26 04DE  07 0AC0  3928              END;
26 04DE  07 0AC0  3929              BEGIN  COMMENT   /  ;                                          4
26 04E2  07 0AC0  3930                 CONVERT; R10 := R6; R10 := TREE(R10);
26 04F2  07 0AC0  3931                 R6 := R6 SHLL 1 SHRL 25;
26 04FA  07 0AC0  3932                 IF R6 <= 3 THEN
26 0502  07 0AC0  3933                 BEGIN  SUBDIV;  R0 := R0 OR D;  ASSEMBLE;
26 0516  07 0AC0  3934                 END ELSE
26 0516  07 0AC0  3935                 BEGIN  SET(DIVFLAG);  IF R10 < 0 THEN
26 0524  07 0AC0  3936                    BEGIN  COMMENT ON LEFT SIDE;
26 0524  07 0AC0  3937                       STFUNCTION;  R0 := LSTACKM4(R5) OR #41000000;
26 0536  07 0AC0  3938                       EMIT;    R5 := R5-4;
26 0546  07 0AC0  3939                    END ELSE
26 0546  07 0AC0  3940                    BEGIN  COMMENT ON RIGHT SIDE;
26 054A  07 0AC0  3941                       IF R7 > 0 THEN
26 055C  07 0AC0  3942                       BEGIN  GETADDRESS; R0 := R3 OR #41000000; EMIT;
26 056E  07 0AC0  3943                          R7 := R6 SHLL 24 OR SIGN;  STFUNCTION;
26 0582  07 0AC0  3944                       END ELSE
26 0582  07 0AC0  3945                       BEGIN  DUMPALLFLREG;
26 0592  07 0AC0  3946                          R0 := LSTACKM4(R5) OR #41000000; EMIT;
26 05A6  07 0AC0  3947                          R5 := R5-4;  R7 := R6 SHLL 24 OR SIGN;
26 05B4  07 0AC0  3948                          STFUNCTION;
26 05BE  07 0AC0  3949                       END;
26 05BE  07 0AC0  3950                    END;
26 05BE  07 0AC0  3951                    PCLL;
26 05C2  07 0AC0  3952                 END;
26 05C2  07 0AC0  3953                 R7 := R6 SHLL 24 OR SIGN;
26 05CC  07 0AC0  3954                 CONVERTRESULT;
26 05D6  07 0AC0  3955              END;
26 05D6  07 0AC0  3956              BEGIN  COMMENT   **  ;                                          5
26 05DA  07 0AC0  3957                 INTEGER TYPE;   COMMENT ON EXPONENT SIDE;
26 05DA  07 0AC4  3958                PROCEDURE PCALL(R1);
26 05DA  07 0AC4  3959                 BEGIN  LOGICAL SAVER1;  SAVER1 := R1;
26 05E2  07 0AC8  3960                    R3 := R3 SHLL 16; R0 := @ARITHFN; R0 := NEG R0;
26 05EC  07 0AC8  3961                    EMITCALL;  R1 := SAVER1;
26 05FC  07 0AC8  3962                 END;
26 05FE  07 0AC8  3963                 R6 := STACKP4(R8) SHLL 1 SHRL 25; TYPE := R6; R0 := 0;
```

```
26 0612  07 0AC8  3964          R1 := R7 SHRL 24; IF R1 = NUMBER THEN
26 0620  07 0AC8  3965          BEGIN  R1 := R7 AND #FFFF + CPTBASE;
26 062A  07 0AC8  3966             R1 := CONSPTTAB(R1) AND #FFFF; R0 := PROGRAMM(R1);
26 0636  07 0AC8  3967          END;
26 0636  07 0AC8  3968          IF R0 = 2 AND R6 <= 3 THEN
26 0646  07 0AC8  3969          BEGIN  COMMENT SQUARE OF INTEGER, REAL, LONG REAL;
26 0646  07 0AC8  3970             R0 := LSTACKM4(R5);  IF R0 > 0 THEN
26 0650  07 0AC8  3971             BEGIN  LOADREG;  R0 := LSTACKM4(R5);
26 0660  07 0AC8  3972             END;
26 0660  07 0AC8  3973             R3 := R0; R0 := R0 OR M;  ASSEMBLE;
26 0672  07 0AC8  3974          END ELSE
26 0672  07 0AC8  3975          BEGIN  LOADREG; R0 := R0-R0; IF R0 < LSTACKM8(R5) THEN
26 068C  07 0AC8  3976             BEGIN  R7 := STACKP4(R8);
26 0690  07 0AC8  3977                R5 := R5-4;  LOADREG;  R5 := R5+4;
26 06A4  07 0AC8  3978             END;
26 06A4  07 0AC8  3979             R6 := 1; R3 := LSTACKM4(R5);  RELEASE;
26 06B8  07 0AC8  3980             R6 := TYPE;  R10 := R6;
26 06BE  07 0AC8  3981             R0 := LSTACKM4(R5) SHLL 8 SHRL 28;
26 06CA  07 0AC8  3982             R3 := R6 SHLL 8; R0 := R0 OR R3 OR #41000000;
26 06D6  07 0AC8  3983             IF R6 <= 3 THEN
26 06DE  07 0AC8  3984             BEGIN  R3 := LSTACKM8(R5) AND MASK5 SHRL 16;
26 06EA  07 0AC8  3985                R0 := R0 OR R3;  EMIT;
26 06F8  07 0AC8  3986                R5 := R5-4;  R3 := 1;  PCALL;
26 0704  07 0AC8  3987             END ELSE
26 0704  07 0AC8  3988             BEGIN  EMIT;  R5 := R5-4;  R7 := STACKP4(R8) OR SIGN;
26 071C  07 0AC8  3989                STFUNCTION;  R3 := 2; PCALL;
26 0732  07 0AC8  3990                IF R6 = 4 THEN R0 := #40000000 ELSE R0 := 0;
26 0746  07 0AC8  3991                PRFLREG;
26 0752  07 0AC8  3992             END;
26 0752  07 0AC8  3993          END;
26 0752  07 0AC8  3994          R7 := R6 SHLL 24 OR SIGN;  CONVERTRESULT;
26 0766  07 0AC8  3995       END;
26 0766  07 0AC8  3996       BEGIN  COMMENT   L:=  ;                                    6
26 076A  07 0AC8  3997          LCOLEQARG2;   R7 := SIGN;
26 077A  07 0AC8  3998       END;
26 077A  07 0AC8  3999       BEGIN  COMMENT   A:=  ;                                    7
26 077E  07 0AC8  4000          NUMERICALASSIGN;     R7 := SIGN;
26 078C  07 0AC8  4001       END;
26 078C  07 0AC8  4002       BEGIN  COMMENT   S:=  ;                                    8
26 0790  07 0AC8  4003          SCOLEQARG2;    R7 := SIGN;
26 079E  07 0AC8  4004       END;
26 079E  07 0AC8  4005       BEGIN  COMMENT   R:=  ;                                    9
26 07A2  07 0AC8  4006          RCOLEQARG2;
26 07AE  07 0AC8  4007       END;
26 07AE  07 0AC8  4008       NULL;                                                      10
26 07B2  07 0AC8  4009       NULL;                                                      11
26 07B6  07 0AC8  4010       BEGIN  COMMENT   STEPUNTIL  ;                              12
26 07BA  07 0AC8  4011          LOADREG;  R0 := STACKP4(R8);
26 07CA  07 0AC8  4012          IF R0 ¬= 1 THEN R10 := LSTACKM4(R5-4) + 4 ELSE
26 07DA  07 0AC8  4013          BEGIN  R3 := NEXTADDR + 3 AND #FFFC;  NEXTADDR := R3;
26 07EE  07 0AC8  4014             R10 := 4;  INCRADDR;  R10 := R3;
26 0800  07 0AC8  4015          END;
26 0800  07 0AC8  4016          R0 := LSTACKM4(R5) AND #00F00000 OR R10 OR #50000000;
26 080E  07 0AC8  4017          EMIT;  R3 := LSTACKM4(R5);  LSTACKM4(R5) := R10;
26 0822  07 0AC8  4018          R6 := 1;  RELEASE;
26 0832  07 0AC8  4019       END;
26 0832  07 0AC8  4020       BEGIN  COMMENT   DIV  ;                                    13
26 0836  07 0AC8  4021          DIVREM;  CONVERTRESULT;
```

```
26 C84C  07 0AC8  4022    END;
26 C84C  07 0AC8  4023    BEGIN  COMMENT   REM  ;                                    14
26 C850  07 0AC8  4024       SET(REMAINDER); DIVREM; RESET(REMAINDER);
26 C864  07 0AC8  4025       CONVERTRESULT;
26 C86E  07 0AC8  4026    END;
26 C86E  07 0AC8  4027    BEGIN  COMMENT   <  ;                                    15
26 0872  07 0AC8  4028       R0 := #4;  ARG2EQUALITIES;
26 0880  07 0AC8  4029    END;
26 C880  07 0AC8  4030    BEGIN  COMMENT   <=  ;                                   16
26 0884  07 0AC8  4031       R0 := #C;  ARG2EQUALITIES;
26 C892  07 0AC8  4032    END;
26 C892  07 0AC8  4033    BEGIN  COMMENT   >  ;                                    17
26 C896  07 0AC8  4034       R0 := #2;  ARG2EQUALITIES;
26 C8A4  07 0AC8  4035    END;
26 C8A4  07 0AC8  4036    BEGIN  COMMENT   >=  ;                                   18
26 08A8  07 0AC8  4037       R0 := #A;  ARG2EQUALITIES;
26 C8D6  07 0AC8  4038    END;
26 08B6  07 0AC8  4039    BEGIN  COMMENT   =  ;                                    19
26 08BA  07 0AC8  4040       R0 := #8;  ARG2EQUALITIES;
26 C8C8  07 0AC8  4041    END;
26 08C8  07 0AC8  4042    BEGIN  COMMENT   ¬=  ;                                   20
26 08CC  07 0AC8  4043       R0 := #7;  ARG2EQUALITIES;
26 08DA  07 0AC8  4044    END;
26 08DA  07 0AC8  4045    NULL;                                                    21
26 C8DE  07 0AC8  4046    BEGIN  COMMENT   L:=2  ;                                 22
26 08E2  07 0AC8  4047       SET(CEQ2);  LCOLEQARG2;  R7 := #86020000;  RESET(CEQ2);
26 08FA  07 0AC8  4048    END;
26 C8FA  07 0AC8  4049    BEGIN  COMMENT   A:=2  ;                                 23
26 08FE  07 0AC8  4050       SET(CEQ2); NUMERICALASSIGN; CONVERTRESULT; RESET(CEQ2);
26 091A  07 0AC8  4051    END;
26 C91A  07 0AC8  4052    BEGIN  COMMENT   S:=2  ;                                 24
26 091E  07 0AC8  4053       SET(CEQ2);  SCOLEQARG2;  R7 := #87000000;  RESET(CEQ2);
26 0934  07 0AC8  4054    END;
26 C934  07 0AC8  4055    BEGIN  COMMENT   R:=2  ;                                 25
26 0938  07 0AC8  4056       SET(CEQ2);  RCOLEQARG2;  RESET(CEQ2);
26 C94C  07 0AC8  4057    END;
26 C94C  07 0AC8  4058    NULL;                                                    26
26 0950  07 0AC8  4059    NULL;                                                    27
26 0954  07 0AC8  4060    NULL;                                                    28
26 C958  07 0AC8  4061    BEGIN  COMMENT   AP)  ;                                  29
26 095C  07 0AC8  4062       R1 := STACKP8(R8);
26 0960  07 0AC8  4063       IF R1 < 0 THEN
26 0966  07 0AC8  4064       BEGIN  R0 := 0;  R1 := R1 AND MASK / 12;
26 0972  07 0AC8  4065          R3 := R1;  STANDARDFUNCTION;   CONVERTRESULT;
26 0988  07 0AC8  4066       END ELSE
26 0988  07 0AC8  4067       BEGIN  R3 := 3;  PARAMETERS;  R7 := R7 OR SIGN;
26 099E  07 0AC8  4068       END;
26 C99E  07 0AC8  4069    END;
26 099E  07 0AC8  4070    BEGIN  COMMENT   INDX  ;                                 30
26 09A2  07 0AC8  4071       R3 := @INDEXOP;    ARRAYS;
26 09B0  07 0AC8  4072    END;
26 09B0  07 0AC8  4073    BEGIN  COMMENT   REFX  ;                                 31
26 09B4  07 0AC8  4074       R3 := R7 AND TYPEMASK;  IF ¬= AND CHECKFLAG THEN
26 09C6  07 0AC8  4075       BEGIN  R0 := LSTACKM4(R5) XOR SIGN OR REFTEMP OR
26 09D2  07 0AC8  4076          #50000000; EMIT;  R0 := #95000000 OR R3 OR REFTEMP;
26 09EC  07 0AC8  4077          EMIT;  R0 := #45100000 OR REFBINDERR; EMIT;
26 0A0C  07 0AC8  4078       END;
26 0A0C  07 0AC8  4079       R3 := R7 AND #FFFF;  R3 := IDLOC2(R3);
```

```
26 0A16  07 0AC8  4080                  R5 := R5 - 4;  RESETRECORD;  R5 := R5 + 4;
26 0A2A  07 0AC8  4081                  R0 := LSTACKM4(R5) XOR SIGN;  R10 := R0 SHRL 8;
26 0A38  07 0AC8  4082                  R10 := R10 ++ R3;
26 0A3A  07 0AC8  4083                  LSTACKM4(R5) := R10;  GETTYPE;
26 0A4A  07 0AC8  4084                  IF R3 = 6 THEN R10 := #10000 ELSE
26 0A56  07 0AC8  4085                  IF R3 = 7 THEN
26 0A62  07 0AC8  4086                  BEGIN  GETLENGTH;  R10 := R3 SHLL 16;  R3 := 7;
26 0A78  07 0AC8  4087                  END ELSE R10 := 0;
26 0A80  07 0AC8  4088                  R7 := R3 SHLL 24 OR SIGN OR R10;
26 0A8C  07 0AC8  4089                  CONVERTRESULT;
26 0A96  07 0AC8  4090               END;
26 0A96  07 0AC8  4091               BEGIN  COMMENT   IFEXP  ;                          32
26 0A9A  07 0AC8  4092                  SET(IFEXP);  UNCONDJUMP;  RESET(IFEXP);
26 0AAC  07 0AC8  4093                  R3 := LOGPOINTER + 4;  LOGPOINTER := R3;
26 0AB8  07 0AC8  4094                  R3 := INSCOUNTER;  FIXUP;
26 0AC8  07 0AC8  4095                  R3 := LOGPOINTER - 4;  LOGPOINTER := R3;
26 0AD4  07 0AC8  4096                  CONVERTRESULT;
26 0ADE  07 0AC8  4097               END;
26 0ADE  07 0AC8  4098               NULL;                                              33
26 0AE2  07 0AC8  4099               NULL;                                              34
26 0AE6  07 0AC8  4100       END;  R1 := SAVER1;  END;

                 SEGMENT 26  NAME = SEG#26      LENGTH = 0C18  BASE REG = 15

21 0960  07 0AC8  4101    SEGMENT PROCEDURE ARG22(R1);
21 0960  07 0AC8  4102       BEGIN  LOGICAL SAVER1;  SAVER1 := R1;
27 0004  07 0ACC  4103          CASE R3 OF
27 0004  07 0ACC  4104          BEGIN
27 0004  07 0ACC  4105               BEGIN  COMMENT   SHL  ;                           35
27 000C  07 0ACC  4106                  BITSSHIFTARG2;
27 0018  07 0ACC  4107                  R0 := R0 OR #89000000; EMIT;
27 0028  07 0ACC  4108                  R7 := R6 SHLL 24 OR SIGN;
27 0032  07 0ACC  4109               END;
27 0032  07 0ACC  4110               BEGIN  COMMENT   SHR  ;                           36
27 0036  07 0ACC  4111                  BITSSHIFTARG2;
27 0042  07 0ACC  4112                  R0 := R0 OR #88000000; EMIT;
27 0052  07 0ACC  4113                  R7 := R6 SHLL 24 OR SIGN;
27 005C  07 0ACC  4114               END;
27 005C  07 0ACC  4115               BEGIN  COMMENT   BB  ;                            37
27 0060  07 0ACC  4116                  R1 := STACKP8(R8); IF R1 ¬= 0 THEN
27 006A  07 0ACC  4117                  BEGIN  COMMENT AT LEAST ONE REFERENCE ARRAY;
27 006A  07 0ACC  4118                     R1 := R1 SHLL 16;  R0 := CLN SHLL 12 OR REFARY+1 OR
27 007E  07 0ACC  4119                      R1 OR #92000000;  EMIT;
27 0090  07 0ACC  4120                     R0 := @NULLREFARY; R0 := NEG R0;  EMITCALL;
27 00A2  07 0ACC  4121                  END;              R7 := SIGN;
27 00A6  07 0ACC  4122               END;
27 00A6  07 0ACC  4123               BEGIN  COMMENT   END  ;                           38
27 00AA  07 0ACC  4124                  R0 := STACKP8(R8);
27 00AE  07 0ACC  4125                  IF R0 < 0 THEN
27 00B4  07 0ACC  4126                  BEGIN COMMENT BLOCKEXIT;
27 00B4  07 0ACC  4127                     R3 := 4;      ENTRYEXIT;
27 00C2  07 0ACC  4128                  END;
27 00C2  07 0ACC  4129                  R0 := R7 AND OPCODEMASK;
27 00C8  07 0ACC  4130                  IF R0 = NULLST THEN R7 := SIGN ELSE R7 := R7 OR SIGN;
27 00DC  07 0ACC  4131               END;
27 00DC  07 0ACC  4132               BEGIN  COMMENT   PCL  ;                           39
27 00E0  07 0ACC  4133                  R3 := 2;        ENTRYEXIT;
27 00EE  07 0ACC  4134               END;
```

```
27 00EE  07 0ACC   4135                BEGIN  COMMENT   SUBSTRIN  ;                                  40
27 00F2  07 0ACC   4136                   BYTE LOADFLAG;  RESET(LOADFLAG);
27 00F6  07 0ACD   4137                   IF R7 < 0 THEN
27 00FC  07 0ACD   4138                   BEGIN  R0 := LSTACKM8(R5) AND #FFF;
27 0104  07 0ACD   4139                      IF R0 > 0 THEN
27 010A  07 0ACD   4140                      BEGIN R5 := R5 - 4;  LOADREG;  R5 := R5 + 4;
27 011E  07 0ACD   4141                         SET(LOADFLAG);
27 0122  07 0ACD   4142                      END;
27 0122  07 0ACD   4143                      R0 := STACKP4(R8);  IF R0 < 0 THEN
27 012C  07 0ACD   4144                      BEGIN  R0 := R0 XOR SIGN + STACKP8(R8);
27 0134  07 0ACD   4145                         R3 := R7 SHLL 8 SHRL 24;
27 013E  07 0ACD   4146                         IF R3 >= R0 THEN GOTO OPT1;
27 0144  07 0ACD   4147                      END;
27 0144  07 0ACD   4148                      IF CHECKFLAG THEN
27 014C  07 0ACD   4149                      BEGIN  R0 := R7 SHLL 8 SHRL 24 -STACKP8(R8) OR LAA;
27 015E  07 0ACD   4150                         EMIT; R3 := LSTACKM8(R5) AND MASK5;
27 0172  07 0ACD   4151                         R0 := R3 OR #15000000; EMIT;
27 0184  07 0ACD   4152                         R0 := #45100000 OR STRINGERR; EMIT;
27 0198  07 0ACD   4153                      END;
27 0198  07 0ACD   4154  OPT1:               R0 := LSTACKM8(R5) AND MASK5 SHRL 4
27 01A0  07 0ACD   4155                         OR LSTACKM8(R5) AND #FFF000 OR LSTACKM4(R5)
27 01AC  07 0ACD   4156                         OR #41000000;
27 01B4  07 0ACD   4157                      EMIT;
27 01C0  07 0ACD   4158                      R0 := LSTACKM8(R5) XOR SIGN SHRL 8; LSTACKM8(R5):=R0;
27 01CC  07 0ACD   4159                      R3 := LSTACKM4(R5);  RELEASE;  R5 := R5 - 4;
27 01E4  07 0ACD   4160                      IF LOADFLAG THEN
27 01EC  07 0ACD   4161                      BEGIN  R6 := 1; ADJSTACKS; R6 := 7;
27 0200  07 0ACD   4162                      END;
27 0200  07 0ACD   4163                      R7 := STACKP8(R8) SHLL 16 OR #7000000;
27 02CC  07 0ACD   4164                   END ELSE
27 02CC  07 0ACD   4165                   BEGIN
27 0210  07 0ACD   4166                      LOGICAL SAVEADD;
27 0210  07 0AD4   4167                      R0 := STACKP4(R8);  IF R0 < 0 THEN
27 021A  07 0AD4   4168                      BEGIN  R0 := R0 XOR SIGN + STACKP8(R8);
27 0222  07 0AD4   4169                         GETLENGTH;  IF R3 >= R0 THEN GOTO OPT2;
27 0234  07 0AD4   4170                      END;
27 0234  07 0AD4   4171                      IF CHECKFLAG THEN
27 023C  07 0AD4   4172                      BEGIN  GETLENGTH; R0 := R3 - STACKP8(R8) OR LAA;
27 0252  07 0AD4   4173                         EMIT; R3 := LSTACKM4(R5) AND MASK5;
27 0266  07 0AD4   4174                         R0 := R3 OR #15000000; EMIT;
27 0278  07 0AD4   4175                         R0 := #45100000 OR STRINGERR; EMIT;
27 028C  07 0AD4   4176                      END;
27 028C  07 0AD4   4177  OPT2:               GETADDRESS;
27 0298  07 0AD4   4178                      R0 := LSTACKM4(R5);  SAVEADD := R3;  IF R0 < 0 THEN
27 02A6  07 0AD4   4179                      BEGIN  R0 := R0 AND MASK5 SHRL 4 OR LSTACKM4(R5)
27 02AE  07 0AD4   4180                         XOR SIGN OR R3 OR LAA;  EMIT;
27 02C8  07 0AD4   4181                      END ELSE
27 02C8  07 0AD4   4182                      BEGIN  R10 := R0;  R5 := R5 - 4;  GENREG;
27 02DE  07 0AD4   4183                         R3 := R0 SHRL 8;  IF R3 = SAVEADD THEN
27 02EC  07 0AD4   4184                         BEGIN  R0 := LSTACKM4(R5) XOR SIGN OR A;
27 02F8  07 0AD4   4185                            R3 := R10;  R6 := 1;  ASSEMBLE;
27 030A  07 0AD4   4186                         END ELSE
27 030A  07 0AD4   4187                         BEGIN  R0 := R0 OR #58000000 OR R10;  EMIT;
27 0320  07 0AD4   4188                            R3 := LSTACKM4(R5) XOR SIGN;
27 0328  07 0AD4   4189                            R0 := R3 SHRL 4 OR R3 OR SAVEADD OR LAA;
27 0338  07 0AD4   4190                            EMIT;
27 0344  07 0AD4   4191                         END;
27 0344  07 0AD4   4192                      END;
```

```
27 0344  07 0AD4  4193                    R0 := LSTACKM4(R5) XOR SIGN SHRL 8;
27 0350  07 0AD4  4194                    LSTACKM4(R5) := R0;
27 0354  07 0AD4  4195                    R7 := STACKP8(R8) SHLL 16 OR #7000000;
27 0360  07 0AD4  4196                 END;
27 0360  07 0AD4  4197                 R7 := R7 OR SIGN;
27 0364  07 0AD4  4198              END;
27 0364  07 0AD4  4199              BEGIN  COMMENT   |  ;                                  41
27 0368  07 0AD4  4200                 R3 := R7 AND MASK;  STACKP8(R8+12) := R3;
27 0372  07 0AD4  4201                 R3 := STACKP4(R8);  STACKP4(R8+12) := R3;
27 037A  07 0AD4  4202              END;
27 037A  07 0AD4  4203              BEGIN  COMMENT   AP,  ;                                42
27 037E  07 0AD4  4204                 R1 := STACKP8(R8);
27 0382  07 0AD4  4205                 IF R1 < 0 THEN
27 0388  07 0AD4  4206                 BEGIN  R0 := 0;  STACKP8(R8+12) := R1;
27 0390  07 0AD4  4207                    R1 := R1 AND MASK / 12; R3 := R1; STANDARDFUNCTION;
27 03A4  07 0AD4  4208                 END ELSE
27 03A4  07 0AD4  4209                 BEGIN  R3 := 2;  PARAMETERS;  R3 := R8 - SSIZE;
27 03BC  07 0AD4  4210                    R1 := @STACKP4(R8);  R10 := @STACKP4(R3);
27 03C4  07 0AD4  4211                    LM(R0,R1,B1);  STM(R0,R1,B10);  R7 := SIGN;
27 03D0  07 0AD4  4212                 END;
27 03D0  07 0AD4  4213              END;
27 03D0  07 0AD4  4214              BEGIN  COMMENT   R,  ;                                 43
27 03D4  07 0AD4  4215                 RECDESIGASSIGN;
27 03DE  07 0AD4  4216              END;
27 03DE  07 0AD4  4217              BEGIN  COMMENT   AR,  ;                                44
27 03E2  07 0AD4  4218                 R3 := @FILLDESCRIP;    ARRAYS;
27 03F0  07 0AD4  4219              END;
27 03F0  07 0AD4  4220              BEGIN  COMMENT   AR)  ;                                45
27 03F4  07 0AD4  4221                 R3 := @CLOSEDESCRIP;      ARRAYS;
27 04C2  07 0AD4  4222              END;
27 04C2  07 0AD4  4223              BEGIN  COMMENT   R)  ;                                 46
27 04C6  07 0AD4  4224                 RECDESIGASSIGN;
27 0410  07 0AD4  4225              END;
27 0410  07 0AD4  4226              BEGIN  COMMENT   LOGOR  ;                              47
27 0414  07 0AD4  4227                 RESET(ANDFLAG);  ANDORARG2;
27 0424  07 0AD4  4228              END;
27 0424  07 0AD4  4229              BEGIN  COMMENT   BITOR  ;                              48
27 0428  07 0AD4  4230                 BITSANDORARG2;
27 0434  07 0AD4  4231                 R0 := R0 OR #06000000; ASSEMBLE;
27 0444  07 0AD4  4232                 R7 := R6 SHLL 24 OR SIGN;
27 044E  07 0AD4  4233              END;
27 044E  07 0AD4  4234              BEGIN  COMMENT   LOGAND  ;                             49
27 0452  07 0AD4  4235                 SET(ANDFLAG);  ANDORARG2;
27 0462  07 0AD4  4236              END;
27 0462  07 0AD4  4237              BEGIN  COMMENT   BITAND  ;                             50
27 0466  07 0AD4  4238                 BITSANDORARG2;
27 0472  07 0AD4  4239                 R0 := R0 OR #04000000; ASSEMBLE;
27 0482  07 0AD4  4240                 R7 := R6 SHLL 24 OR SIGN;
27 048C  07 0AD4  4241              END;
27 048C  07 0AD4  4242              BEGIN  COMMENT   ITERST  ;                             51
27 0490  07 0AD4  4243                 IF R7 >= 0 AND R7 ¬= NULLST THEN CALLPROPROCWOPARAM;
27 04AA  07 0AD4  4244                 R0 := #58200000 OR LSTACKM4(R5);  EMIT;
27 04BE  07 0AD4  4245                 R10 := STACKP8(R8);  R1 := STACKP4(R8);
27 04C6  07 0AD4  4246                 CASE R1 OF
27 04C6  07 0AD4  4247                 BEGIN
27 04C6  07 0AD4  4248                    BEGIN  R0 := #47F0E004 ++ R10;  EMIT;
27 04E0  07 0AD4  4249                       R0 := INSCOUNTER OR #E000; PROGRAM(R10+14) := R0;
27 04EC  07 0AD4  4250                    END;
```

```
27 04EC  07 0AD4  4251                BEGIN  R0 := #98450000 OR LSTACKM4(R5-8);  EMIT;
27 0504  07 0AD4  4252                   R0 := #8724E008 ++ R10;  EMIT;
27 0516  07 0AD4  4253                   R0 := INSCOUNTER OR #E000;  PROGRAM(R10+6) := R0;
27 0522  07 0AD4  4254                END;
27 0522  07 0AD4  4255                BEGIN  R0 := #47F0E004 ++ R10;  EMIT;
27 0538  07 0AD4  4256                   R0 := INSCOUNTER OR #E000; PROGRAM(R10+18) := R0;
27 0544  07 0AD4  4257                   R0 := R0 ++ #47000004;  EMIT;
27 0554  07 0AD4  4258                END;
27 0554  07 0AD4  4259             END;
27 0564  07 0AD4  4260             R5 := R5 - 12;  R7 := SIGN;
27 056C  07 0AD4  4261          END;
27 056C  07 0AD4  4262          BEGIN  COMMENT   ITERST2  ;                                  52
27 0570  07 0AD4  4263             IF R7 >= 0 AND R7 ¬= NULLST THEN CALLPROPROCWOPARAM;
27 058A  07 0AD4  4264             R10 := STACKP4(R8);  IF R10 ¬= 0 THEN
27 0594  07 0AD4  4265             BEGIN  R0 := LSTACKM4(R5) OR #58100000;  EMIT;
27 05A8  07 0AD4  4266                R0 := #07F10000;  EMIT;   R5 := R5 - 4;
27 05BC  07 0AD4  4267                R0 := INSCOUNTER OR #E000; PROGRAM(R10+2) := R0;
27 05C8  07 0AD4  4268             END;
27 05C8  07 0AD4  4269             R7 := SIGN;
27 05CC  07 0AD4  4270          END;
27 05CC  07 0AD4  4271          BEGIN  COMMENT   FORLIST  ;                                  53
27 05D0  07 0AD4  4272             ARG2FORLIST;   R7 := #81000000;
27 05E0  07 0AD4  4273          END;
27 05E0  07 0AD4  4274          BEGIN  COMMENT   FORCL  ;                                    54
27 05E4  07 0AD4  4275             R10 := STACKP4(R8);  R0 := INSCOUNTER;
27 05EC  07 0AD4  4276             R3 := R8 - SSIZE; STACKP4(R3) := R10; STACKP8(R3) := R0;
27 05F6  07 0AD4  4277             IF R10 ¬= 2 THEN
27 0602  07 0AD4  4278             BEGIN  R0 := #47F0E008 ++ INSCOUNTER;  EMIT;
27 0616  07 0AD4  4279                R0 := #5A200000 OR LSTACKM4(R5-8);  EMIT;
27 062A  07 0AD4  4280             END;
27 062A  07 0AD4  4281             R0 := #59200000 OR LSTACKM4(R5-4);  EMIT;
27 063E  07 0AD4  4282             IF R10 = 1 THEN R0 := #47400000  ELSE
27 064A  07 0AD4  4283             IF R10 = 2 THEN R0 := #47200000  ELSE
27 065A  07 0AD4  4284             BEGIN  R0 := #58300008 ++ LSTACKM4(R5-8);  EMIT;
27 0672  07 0AD4  4285                R0 := #44300000;
27 0676  07 0AD4  4286             END;
27 0676  07 0AD4  4287             EMIT;  R0 := #50200000 OR LSTACKM4(R5);  EMIT;
27 0696  07 0AD4  4288          END;
27 0696  07 0AD4  4289          BEGIN  COMMENT   ENDFORLI  ;                                 55
27 069A  07 0AD4  4290             R0 := STACKP4(R8);  IF R0 = 0 THEN
27 06A4  07 0AD4  4291             BEGIN  LOADREG;  R10 := 0;  COMMENT FLAG THIS CASE;
27 06B4  07 0AD4  4292                R0 := LSTACKM4(R5) AND #00F00000 OR LSTACKM4(R5-4)
27 06BC  07 0AD4  4293                 OR #50000000;  EMIT;
27 06D0  07 0AD4  4294                R3 := LSTACKM4(R5); R6 := 1; RELEASE;  R5 := R5 - 8;
27 06E8  07 0AD4  4295             END ELSE
27 06E8  07 0AD4  4296             BEGIN  ARG2FORLIST;   R0 := #47F00000; EMIT;
27 0708  07 0AD4  4297                R0 := INSCOUNTER OR #E000;  R1 := INSCOUNTER - 8;
27 0718  07 0AD4  4298                WHILE R1 ¬= 0 DO
27 071E  07 0AD4  4299                BEGIN  R3 := PROGRAM(R1+2);
27 0722  07 0AD4  4300                   PROGRAM(R1+2) := R0;  R1 := R3;
27 0728  07 0AD4  4301                END;
27 072C  07 0AD4  4302                R3 := NEXTADDR + 3 AND #FFFC;  NEXTADDR := R3;
27 073C  07 0AD4  4303                R0 := R3 OR #50100000; EMIT; R10 := 4; INCRADDR;
27 075E  07 0AD4  4304                R0 := #50200000 OR LSTACKM4(R5);  EMIT;
27 0772  07 0AD4  4305                LSTACKM4(R5) := R3;  R10 := INSCOUNTER - 12;
27 077E  07 0AD4  4306             END;
27 077E  07 0AD4  4307             R3 := R8 - SSIZE;  STACKP4(R3) := R10;
27 0788  07 0AD4  4308          END;
```

```
27 C788  07 0AD4  4309          BEGIN  COMMENT   UJIFEXP  ;                                  56
27 C78C  07 0AD4  4310             SET(UJIFEXP);  UNCONDJUMP;  RESET(UJIFEXP);
27 C79E  07 0AD4  4311          END;
27 C79E  07 0AD4  4312          BEGIN  COMMENT   UJ  ;                                      57
27 07A2  07 0AD4  4313             IF R7 ¬= SIGN THEN
27 07AA  07 0AD4  4314             BEGIN R3 := STACKP8(R8);  R1 := SAVEL;
27 07B2  07 0AD4  4315                STC(R3,TREE(R1+1));
27 07B6  07 0AD4  4316                STACKP8(R8+12) := R3;
27 07BA  07 0AD4  4317                IF R7 < 0 THEN CONVERTRESULT;
27 07CA  07 0AD4  4318             END;
27 07CA  07 0AD4  4319             UNCONDJUMP;
27 07D4  07 0AD4  4320          END;
27 07D4  07 0AD4  4321          BEGIN  COMMENT   CL  ;                                      58
27 07D8  07 0AD4  4322             LOGICAL SAVETREE;  R3 := SAVEL AND MASK;
27 07E0  07 0AD8  4323             R1 := TREE(R3);  SAVETREE := R1;
27 07E8  07 0AD8  4324             R0 := STACKP8(R8) SHLL 16;  R1 := R1 AND MASK2 OR R0;
27 07F6  07 0AD8  4325             TREE(R3) := R1;
27 07FA  07 0AD8  4326             RESET(LOGFLAG);
27 07FE  07 0AD8  4327             SET(CLFLAG);  UNCONDJUMP;  RESET(CLFLAG);
27 0810  07 0AD8  4328             R3 := SAVEL;  R1 := SAVETREE;  TREE(R3) := R1;
27 081C  07 0AD8  4329             TEST(LOGFLAG);  IF  =  THEN
27 0824  07 0AD8  4330             BEGIN  R3 := LOGPOINTER;  R0 := LOGSTACK(R3-12);
27 082C  07 0AD8  4331                R10 := LOGSTACK(R3-8);  LOGSTACK(R3-12) := R10;
27 0834  07 0AD8  4332                R10 := LOGSTACK(R3-4);  LOGSTACK(R3-8) := R10;
27 083C  07 0AD8  4333                LOGSTACK(R3-4) := R0;  R7 := #86000000;
27 0844  07 0AD8  4334             END;
27 0844  07 0AD8  4335             R3 := LOGPOINTER - 4;  R10 := @LOGSTACK(R3);
27 0850  07 0AD8  4336             R3 := LOGSTACK(R3);  MVI(0,B10(1));
27 0858  07 0AD8  4337             FIRSTCASE := R3;  R3 := INSCOUNTER - 4;
27 0864  07 0AD8  4338             FIXUP;  R3 := LOGPOINTER + 4;  LOGPOINTER := R3;
27 087C  07 0AD8  4339             FILLCASETABLE;
27 0888  07 0AD8  4340             R3 := LOGPOINTER + 4;  LOGPOINTER := R3;
27 0894  07 0AD8  4341             R3 := INSCOUNTER;  FIXUP;
27 08A4  07 0AD8  4342             R3 := LOGPOINTER - 8;  LOGPOINTER := R3;
27 08B0  07 0AD8  4343             TEST(LOGFLAG);  IF  =  THEN
27 08B8  07 0AD8  4344             BEGIN  R0 := #0200;  INDICATEBRANCH;
27 08C8  07 0AD8  4345             END;
27 08C8  07 0AD8  4346             R7 := R7 OR SIGN;
27 08CC  07 0AD8  4347             RESET(LOGFLAG);
27 08D0  07 0AD8  4348             IF R7 ¬= SIGN THEN CONVERTRESULT;
27 08E2  07 0AD8  4349          END;
27 08E2  07 0AD8  4350          BEGIN  COMMENT   IFST  ;                                    59
27 08E6  07 0AD8  4351             IF R7 > 0 THEN
27 08EC  07 0AD8  4352             BEGIN
27 08EC  07 0AD8  4353                R1 := R7 SHRL 24;
27 08F2  07 0AD8  4354                IF R1 = FUNCID THEN
27 08FA  07 0AD8  4355                BEGIN DUMPALLGENREG; R3:=R3-R3; PROCCALLCODE;
27 0914  07 0AD8  4356                END;
27 0914  07 0AD8  4357             END;
27 0914  07 0AD8  4358             R7 := SIGN;
27 0918  07 0AD8  4359             R3 := LOGPOINTER + 4;  LOGPOINTER := R3;
27 0924  07 0AD8  4360             R3 := INSCOUNTER;  FIXUP;
27 0934  07 0AD8  4361             R3 := LOGPOINTER - 4;  LOGPOINTER := R3;
27 0940  07 0AD8  4362          END;
27 0940  07 0AD8  4363          BEGIN  COMMENT   ::  ;                                      60
27 0944  07 0AD8  4364             LOADREG;        R7 := R7 OR SIGN;
27 0954  07 0AD8  4365          END;
27 0954  07 0AD8  4366       END;  R1 := SAVER1;  END;
```

SEGMENT 27 NAME = SEG#27 LENGTH = 0AA0 BASE REG = 15

```
21 0960  07 0AD8  4367    SEGMENT PROCEDURE ARG23(R1);
21 0960  07 0AD8  4368       BEGIN  LOGICAL SAVER1;  SAVER1 := R1;
28 0004  07 0ADC  4369          CASE R3 OF
28 0004  07 0ADC  4370          BEGIN
28 0004  07 0ADC  4371                BEGIN  COMMENT   IS  ;                                          61
28 000C  07 0ADC  4372                   R0 := TYPEINFO(R7) SHLL 24 SHRL 8 OR CLII  OR
28 001C  07 0ADC  4373                              LSTACKM4(R5);    EMIT;
28 002C  07 0ADC  4374                   R6 := 1;  R3 := LSTACKM4(R5);  RELEASE;
28 0040  07 0ADC  4375                   R5 := R5 - 4;  RESETRECORD;
28 0050  07 0ADC  4376                   R3 := LOGPOINTER + 8;  LOGPOINTER := R3;
28 005C  07 0ADC  4377                   R0 := 0;  LOGSTACK(R3) := R0;  LOGSTACK(R3-4) := R0;
28 0068  07 0ADC  4378                   R0 := #0108;  INDICATEBRANCH; R7 := #86000000;
28 007C  07 0ADC  4379                END;
28 007C  07 0ADC  4380                NULL;                                                           62
28 0080  07 0ADC  4381                BEGIN  COMMENT   ,  ;                                           63
28 0084  07 0ADC  4382                   R1 := R7 SHRL 24;
28 008A  07 0ADC  4383                   IF R1  = FUNCID THEN
28 0092  07 0ADC  4384                   BEGIN COMMENT PROCEDURE CALL WITH NO PARAMETERS.  CHECK
28 0092  07 0ADC  4385                      IF TYPED;
28 0092  07 0ADC  4386                      GETTYPE;  IF R3 = 0 THEN
28 00A4  07 0ADC  4387                      BEGIN COMMENT PROPER PROCEDURE;
28 00A4  07 0ADC  4388                         DUMPALLGENREG;DUMPALLFLREG;R3:=R3-R3;PROCCALLCODE;
28 00BE  07 0ADC  4389                         R7 := SIGN;
28 00CE  07 0ADC  4390                      END ELSE
28 00CE  07 0ADC  4391                      BEGIN COMMENT TYPED - END OF TYPED PROCEDURE;
28 00D2  07 0ADC  4392                         TERMPROCEXIT;    R7 := R6 SHLL 24 OR SIGN;
28 00E8  07 0ADC  4393                      END;
28 00E8  07 0ADC  4394                   END ELSE
28 00E8  07 0ADC  4395                   IF R7 >= 0 AND R1 ¬= SEG AND R1 ¬= RCCLDC THEN
28 0102  07 0ADC  4396                      TERMPROCEXIT;
28 010E  07 0ADC  4397                   CONVERTRESULT;
28 0118  07 0ADC  4398                END;
28 0118  07 0ADC  4399                BEGIN  COMMENT   WHILEOP  ;                                     64
28 011C  07 0ADC  4400                   IFJARG2;
28 0128  07 0ADC  4401                END;
28 0128  07 0ADC  4402                BEGIN  COMMENT   WHILEST  ;                                     65
28 012C  07 0ADC  4403                   IF R7 >= 0 AND R7 ¬= NULLST THEN CALLPROPROCWOPARAM;
28 0146  07 0ADC  4404                   R0 := #47F0E000 OR STACKP4(R8);  EMIT;
28 015A  07 0ADC  4405                   R3 := INSCOUNTER;  FIXUP;  R3 := LOGPOINTER - 4;
28 0172  07 0ADC  4406                   LOGPOINTER := R3;      R7 := SIGN;
28 017A  07 0ADC  4407                END;
28 017A  07 0ADC  4408                BEGIN  COMMENT   IFJ  ;                                         66
28 017E  07 0ADC  4409                   IFJARG2;
28 018A  07 0ADC  4410                END;
28 018A  07 0ADC  4411                BEGIN  COMMENT   UMINUS  ;                                      67
28 018E  07 0ADC  4412                   CONVERT; LOADREG;
28 01A4  07 0ADC  4413                   R6 := R7 SHLL 1 SHRL 25;
28 01AE  07 0ADC  4414                   R0 := LSTACKM4(R5); R3 := R0; R0 := R0 OR #03000000;
28 01B8  07 0ADC  4415                   ASSEMBLE;
28 01C4  07 0ADC  4416                   CONVERTRESULT;  R7 := R7 AND OPCODEMASK;
28 01D2  07 0ADC  4417                END;
28 01D2  07 0ADC  4418                BEGIN  COMMENT   ABS  ;                                         68
28 01D6  07 0ADC  4419                   CONVERT;  IF R7 < 0 THEN R3 := R7 SHLL 1 SHRL 25 ELSE
28 01F0  07 0ADC  4420                   GETTYPE;  IF R3 < 4 THEN
28 0208  07 0ADC  4421                   BEGIN  LOADREG;
28 0214  07 0ADC  4422                   R6 := R7 SHLL 1 SHRL 25;
```

```
28 021E  07 0ADC  4423                 R0 := LSTACKM4(R5); R3 := R0; ASSEMBLE;
28 0230  C7 0ADC  4424                 END ELSE
28 0230  07 0ADC  4425                 BEGIN  SHORT INTEGER SAVETYPE;  SAVETYPE := R3;
28 0238  07 0ADE  4426                    STFUNCTION;  R0 := @ARITHFN; R0 := NEG R0;
28 0248  07 0ADE  4427                    R3 := SAVETYPE - 1; COMMENT 3=CABS, 4=LCABS;
28 0250  07 0ADE  4428                    R3 := R3 SHLL 16;  EMITCALL;  R3 := SAVETYPE;
28 0264  07 0ADE  4429                    IF R3 = 4 THEN
28 026C  C7 0ADE  4430                    BEGIN  R0 := #C0000000;  R7 := #82000000;  END ELSE
28 0274  07 0ADE  4431                    BEGIN R0:=#80000000; R7:=#83000000; END; FLREG;
28 028C  C7 0ADE  4432                 END;
28 028C  07 0ADE  4433                 CONVERTRESULT;  R7 := R7 AND OPCODEMASK;
28 029A  07 0ADE  4434              END;
28 029A  07 0ADE  4435              NULL;                                                    69
28 029E  07 0ADE  4436              NULL;                                                    70
28 02A2  07 0ADE  4437              BEGIN  COMMENT   LOG¬  ;                                 71
28 02A6  07 0ADE  4438                 IF R7 >= 0 THEN
28 02AC  C7 0ADE  4439                 BEGIN  LOGCOMPARE;  R3 := LOGPOINTER + 8;
28 02C0  07 0ADE  4440                    LOGPOINTER := R3;  R0 := 0;  LOGSTACK(R3) := R0;
28 02CC  C7 0ADE  4441                    LOGSTACK(R3-4) := R0;
28 02D0  07 0ADE  4442                    R0 := #0107;  INDICATEBRANCH;
28 02E0  07 0ADE  4443                 END ELSE
28 02E0  07 0ADE  4444                 BEGIN  R3 := R7 SHLL 8 SHRL 24;
28 02EE  07 0ADE  4445                    IF R3 = 1 THEN
28 02F6  07 0ADE  4446                    BEGIN  R0 := LSTACKM4(R5) OR CLII OR #00010000;
28 0302  07 0ADE  4447                    EMIT;  R3 := LOGPOINTER + 8;  LOGPOINTER := R3;
28 031A  07 0ADE  4448                    R0 := 0;  LOGSTACK(R3) := R0;  LOGSTACK(R3-4) :=R0;
28 0326  07 0ADE  4449                       R3 := LSTACKM4(R5);  R5 := R5 - 4;  RELEASE;
28 033A  07 0ADE  4450                       R0 := #0107;  INDICATEBRANCH;
28 034A  07 0ADE  4451                    END ELSE
28 034A  07 0ADE  4452                    BEGIN  R3 := LOGPOINTER;  R0 := LOGSTACK1(R3) XOR
28 0356  07 0ADE  4453                       #0300;  LOGSTACK1(R3) := R0;
28 035E  07 0ADE  4454                    END;
28 035E  07 0ADE  4455                 END;
28 035E  07 0ADE  4456                 R7 :=#86000000;
28 0362  07 0ADE  4457              END;
28 0362  07 0ADE  4458              BEGIN  COMMENT   BIT¬  ;                                 72
28 0366  07 0ADE  4459                 LOADREG;
28 0372  07 0ADE  4460                 R6 := R7 SHLL 1 SHRL 25;
28 037C  07 0ADE  4461                 R0 := LSTACKM4(R5) AND MASK5 OR #57000000;
28 0388  07 0ADE  4462                 R0 := R0 OR ALLONES;    EMIT;
28 0398  07 0ADE  4463              END;
28 0398  07 0ADE  4464              NULL;                                                    73
23 039C  07 0ADE  4465              NULL;                                                    74
28 03A0  07 0ADE  4466              BEGIN  COMMENT   GOTO  ;                                 75
28 03A4  07 0ADE  4467                 R3 := 6;    ENTRYEXIT;
28 03B2  07 0ADE  4468              END;
28 03B2  07 0ADE  4469              BEGIN  COMMENT   :  ;                                    76
28 03B6  07 0ADE  4470                 R3 := 5;    ENTRYEXIT;
28 03C4  07 0ADE  4471              END;
28 03C4  07 0ADE  4472              BEGIN  COMMENT   STACKADDR  ;                            77
28 03C8  07 0ADE  4473              END;
28 03C8  07 0ADE  4474              BEGIN                                                    78
28 03CC  07 0ADE  4475                 IFJARG2;
28 03D8  07 0ADE  4476              END;
28 03D8  07 0ADE  4477              BEGIN  COMMENT   CARD  ;                                 79
28 03DC  07 0ADE  4478                 R3 := SAVEL;  R3 := TREE(R3) AND MASK;  CARDNUM := R3;
28 03EC  07 0ADE  4479                 R10:=RUNERRPT; R0:=R7 SHRL 24; IF R0¬=PROCDC THEN
28 03FE  07 0ADE  4480                 BEGIN R1 := R10 - 4;  R0 := B1 AND MASK;
```

```
28 040C  07 0ADE  4431                IF R0 = CARDNUM THEN GOTO H;    R0 := B1 SHRL 16;
28 041C  07 0ADE  4432                IF R0 = INSCOUNTER THEN
28 0424  07 0ADE  4433                BEGIN
28 0424  07 0ADE  4434                   R0 := B1 AND MASK1 OR R3;  B1 := R0;  GOTO H;
28 0436  07 0ADE  4435                END;
28 0436  07 0ADE  4436             END;
28 0436  07 0ADE  4437             COMMENT CHECK IF CARD TABLE WILL OVERWRITE TREE;
28 0436  07 0ADE  4438             IF R10 >= TREEBASE THEN BEGIN R3 := 6;  EROR;  END;
28 044E  07 0ADE  4439             R1 := INSCOUNTER SHLL 16 OR R3;
28 0458  07 0ADE  4440             B10 := R1;    R10 := R10 + 4;    RUNERRPT := R10;
28 0464  07 0ADE  4441          H: R1 := R7 SHRL 24;  R8 := R8 - SSIZE;
28 046E  07 0ADE  4442             IF R1 = PROCDC THEN
28 0476  07 0ADE  4443             BEGIN R10:=R10-4; R3:=B10 AND MASK; B10:=R3;
28 0486  07 0ADE  4444                R3 := 1;  ENTRYEXIT;
28 0494  07 0ADE  4445             END ELSE    IF R1 = BEGINN THEN
28 04A0  07 0ADE  4446             BEGIN COMMENT CHECK FOR BLOCK OR COMPOUND STATEMENT;
28 04A0  07 0ADE  4447                R3:=STACK(R8) AND MASK; R3:=TREE(R3) SHLL 1 SHRL 25;
28 04B4  07 0ADE  4448                IF R3 = BBB THEN
28 04BC  07 0ADE  4449                BEGIN COMMENT BLOCK ENTRY;
28 04BC  07 0ADE  4450                   R3 := 3;  ENTRYEXIT;
28 04CA  07 0ADE  4451                   R1 := SIGN; R3 := R8 - SSIZE;  STACKP8(R3) := R1;
28 04D8  07 0ADE  4452                END ELSE
28 04D8  07 0ADE  4453                BEGIN COMMENT COMPOUND STATEMENT;
28 04DC  07 0ADE  4454                   R1 := 0;  STACKP8(R8) := R1;
28 04E4  07 0ADE  4455                END;
28 04E4  07 0ADE  4456             END ELSE
28 04E4  07 0ADE  4457             IF R1 = ARRAYDC THEN
28 04F0  07 0ADE  4458             BEGIN   R3 := @INITADCL;  ARRAYS;    R7 := SIGN;
28 0502  07 0ADE  4459             END;
28 0502  07 0ADE  4460             R8 := R8 + SSIZE;
28 0506  07 0ADE  4461          END;
28 0506  07 0ADE  4462          BEGIN  COMMENT   CASE  ;                                    80
28 050A  07 0ADE  4463             R3 := 0;  IC(R3,TREE(R6+1));  IF R3 > 5 THEN R3 := 0;
28 051E  07 0ADE  4464             STACKP8(R8+12) := R3;
28 0522  07 0ADE  4465             R10 := R6;  LOADREG;
28 0530  07 0ADE  4466             R0 := TREE(R10); TEMP := R0;
28 0538  07 0ADE  4467             R0 := R0 AND MASK OR #41000000;
28 0540  07 0ADE  4468             IF CHECKFLAG THEN EMIT;
28 0554  07 0ADE  4469             R3 := LSTACKM4(R5) XOR SIGN; IF CHECKFLAG THEN
28 0564  07 0ADE  4470             BEGIN  R0 := R3 OR #15000000; EMIT;
28 0576  07 0ADE  4471                R0 := #45100000 OR CASEERR; EMIT;
28 058A  07 0ADE  4472             END;
28 058A  07 0ADE  4473             R0 := R3 OR #8B000002; EMIT;
28 059C  07 0ADE  4474             R3 := LSTACKM4(R5);  R6 := 1;  RELEASE;
28 05B0  07 0ADE  4475             MARKRECORDS;  DUMPALLGENREG;  DUMPALLFLREG;
28 05D4  07 0ADE  4476             R0 := LSTACKM4(R5) XOR SIGN SHRL 4 OR #47F00000; EMIT;
28 05F0  07 0ADE  4477             R5 := R5 - 4;
28 05F4  07 0ADE  4478             R0 := INSCOUNTER - 4;
28 05FC  07 0ADE  4479             R3 := LOGPOINTER + 4;
28 06C4  07 0ADE  4480             R10 := TEMP SHLL 16;  R0 := R0 OR R10;
28 06CE  07 0ADE  4481             LOGSTACK(R3) := R0;  R3 := R3 + 4;  CLI(6,TEMP(1));
28 061A  07 0ADE  4482             IF  =  THEN
28 061E  07 0ADE  4483             BEGIN  R10 := 0;  LOGSTACK(R3+4) := R10;
28 0626  07 0ADE  4484                LOGSTACK(R3) := R10;  R3 := R3 + 8;
28 062E  07 0ADE  4485             END;
28 062E  07 0ADE  4486             LOGPOINTER := R3;
28 0632  07 0ADE  4487             R0 := 0;  LOGSTACK(R3) := R0;
28 063A  07 0ADE  4488          END;
```

```
28 063A  07 0ADE  4539                NULL;                                  81
28 063E  07 0ADE  4540                NULL;                                  82
28 0642  07 0ADE  4541             END;
28 069E  07 0ADE  4542          R1 := SAVER1;
28 06A2  07 0ADE  4543       END;

                  SEGMENT 28  NAME = SEG#28      LENGTH = 0718  BASE REG = 15

21 0960  07 0ADE  4544    SEGMENT PROCEDURE  ARG1(R1);
21 0960  07 0ADE  4545       BEGIN  LOGICAL SAVER1;
29 0000  07 0AE4  4546
29 0000  07 0AE4  4547     PROCEDURE ARG1RELATION(R1);
29 0000  07 0AE4  4548        BEGIN  LOGICAL SAVER1;
29 0004  07 0AE8  4549           SAVER1 := R1;
29 0008  07 0AE8  4550           R3 := R7 SHRL 24;
29 000E  07 0AE8  4551           CONVERT;  LOADREG;  R6 := R6 OR R7;
29 0026  07 0AE8  4552           R1 := SAVER1;
29 002A  07 0AE8  4553        END;
29 002C  07 0AE8  4554
29 002C  07 0AE8  4555     PROCEDURE ARG1EQUALITIES(R1);
29 002C  07 0AE8  4556        BEGIN  LOGICAL SAVER1;  SAVER1 := R1;
29 0030  07 0AEC  4557           R3 := R7 SHLL 1 SHRL 25;  IF R7 > 0 THEN IF R3 ¬= RCCLID THEN
29 0048  07 0AEC  4558              GETTYPE ELSE R3 := 9;
29 005C  07 0AEC  4559           IF  R3 < 6 THEN
29 0064  07 0AEC  4560           BEGIN  COMMENT  TERMINAL ARITHMETIC QUANTITY;
29 0064  07 0AEC  4561              CONVERT;  LOADREG;  R6 := R6 OR R7;
29 007C  07 0AEC  4562           END ELSE
29 007C  07 0AEC  4563           BEGIN  R3 := R3 - 5;
29 0084  07 0AEC  4564              CASE R3 OF
29 0084  07 0AEC  4565              BEGIN
29 0084  07 0AEC  4566              LEQUAL;
29 0098  07 0AEC  4567              BEGIN  COMMENT  = STRING REG1;
29 009C  07 0AEC  4568                 IF  R7 >= 0 THEN
29 00A2  07 0AEC  4569                 BEGIN  GETADDRESS;  LSTACK(R5) := R3;
29 00B2  07 0AEC  4570                    IF R3 = #3000 THEN
29 00BA  07 0AEC  4571                    BEGIN R3 := FLAG;  R( 6):= R3;  RSTACK(R2) := R5;
29 00C6  07 0AEC  4572                       R2 := R2 + 4;
29 00CA  07 0AEC  4573                    END;  R5 := R5 + 4;
29 00CE  07 0AEC  4574                 END;
29 00CE  07 0AEC  4575                 R3 := LSTACKM4(R5);  STACKP4(R8) := R3;
29 00D6  07 0AEC  4576              END;
29 00D6  07 0AEC  4577              BEGIN  ARG1RELATION;
29 00DE  07 0AEC  4578              END;
29 00DE  07 0AEC  4579              BEGIN  IF R7 >= 0 THEN
29 00E8  07 0AEC  4580                 BEGIN  R3 := R7 SHLL 1 SHRL 25;  IF R3 = RCCLID THEN
29 00FA  07 0AEC  4581                    BEGIN  RECORDALLOCATE;  R7 := #89000000;
29 010A  07 0AEC  4582                    END;
29 010A  07 0AEC  4583                 END;
29 010A  07 0AEC  4584                 ARG1RELATION;
29 010E  07 0AEC  4585              END;
29 010E  07 0AEC  4586              END;
29 0122  07 0AEC  4587           END;
29 0122  07 0AEC  4588           R1 := SAVER1;
29 0126  07 0AEC  4589        END;
29 0128  07 0AEC  4590
29 0128  07 0AEC  4591     PROCEDURE LCOLEQARG1 (R1);
29 0128  07 0AEC  4592        BEGIN  LOGICAL SAVER1,SAVEADD;  SAVER1 := R1;
29 012C  07 0AF4  4593           IF R7 >= 0 THEN
```

```
29 0132  07 0AF4   4594          BEGIN  R6 := R6 AND MASK;  R3 := TREE(R6);  IF R3< 0 THEN
29 0140  07 0AF4   4595             BEGIN  GETADDRESS;  SAVEADD := R3;  GENREG;
29 015C  07 0AF4   4596                R0 := R0 OR SAVEADD OR ICC;  EMIT;
29 0170  07 0AF4   4597             END ELSE
29 0170  07 0AF4   4598             BEGIN  GETADDRESS;  LSTACK(R5) := R3; R5 := R5 + 4;
29 0188  07 0AF4   4599             END;
29 0188  07 0AF4   4600          END ELSE
29 0188  07 0AF4   4601          BEGIN  R3 := R6 AND MASK; R3 := TREE(R6);  IF  R3 < 0  THEN
29 019C  07 0AF4   4602             BEGIN  R3 := R7 SHLL 8 SHRL 24;  IF R3 = 1 THEN
29 01AE  07 0AF4   4603                BEGIN  R0 := LSTACKM4(R5);  R7 := R7 AND MASK7;
29 01B6  07 0AF4   4604                   R10 := R0 AND BASEMASK SHLL 8;  R0 := R0 OR R10 OR ICC;
29 01C6  07 0AF4   4605                   R10 := R10 OR SIGN;  LSTACKM4(R5) := R10;  EMIT;
29 01DA  07 0AF4   4606                END ELSE IF R3 ¬= 2 THEN EVALUATELOG;
29 01F2  07 0AF4   4607             END;
29 01F2  07 0AF4   4608          END;
29 01F2  07 0AF4   4609          R1 := SAVER1;
29 01F6  07 0AF4   4610       END;
29 01F8  07 0AF4   4611
29 01F8  07 0AF4   4612    PROCEDURE SCOLEQARG1 (R1);
29 01F8  07 0AF4   4613       BEGIN  LOGICAL SAVER1;  SAVER1 := R1;
29 01FC  07 0AF8   4614          IF R7 >= 0 THEN
29 02C2  07 0AF8   4615          BEGIN  COMMENT TERMINALNODE;
29 0202  07 0AF8   4616             GETADDRESS;  LSTACK(R5) := R3;
29 0212  07 0AF8   4617             IF R3 = #3000 THEN
29 021A  07 0AF8   4618             BEGIN  R3 := FLAG;  R( 6):= R3;  RSTACK(R2) := R5;  R2:=R2+4;
29 022A  07 0AF8   4619             END;
29 C22A  07 0AF8   4620             R5 := R5 + 4;
29 022E  07 0AF8   4621          END;
29 022E  07 0AF8   4622          R3 := LSTACKM4(R5);  STACKP4(R8) := R3;
29 0236  07 0AF8   4623          R1 := SAVER1;
29 023A  07 0AF8   4624       END;
29 023C  07 0AF8   4625
29 023C  07 0AF8   4626    PROCEDURE RCOLEQARG1(R1);
29 023C  07 0AF8   4627       BEGIN  LOGICAL SAVER1;  SAVER1 := R1;
29 0240  07 0AFC   4628          R3 := TREE(R6);  IF R3 < 0 THEN
29 024A  C7 0AFC   4629          BEGIN  IF R7 >= 0 THEN
29 0250  07 0AFC   4630             BEGIN  R3 := R7 SHRL 24;
29 0256  07 0AFC   4631                IF R3 = RCCLID THEN RECORDALLOCATE ELSE LOADREG;
29 027A  07 0AFC   4632             END ELSE  LOADREG;
29 028A  07 0AFC   4633          END ELSE
29 028A  07 0AFC   4634          BEGIN  IF R7 >= 0 THEN
29 0294  07 0AFC   4635             BEGIN  GENREG;  GETADDRESS;  R0 := R0 OR R3 OR LAA;  EMIT;
29 02BE  07 0AFC   4636                R0 := LSTACKM4(R5) SHLL 1 SHRL 9;  LSTACKM4(R5) := R0;
29 02CE  07 0AFC   4637             END;
29 02CE  07 0AFC   4638          END;
29 02CE  07 0AFC   4639          R6 := R6 OR #89000000;  R1 := SAVER1;
29 02D6  07 0AFC   4640       END;
29 02D8  07 0AFC   4641
29 02D8  07 0AFC   4642    PROCEDURE ARG1FORLIST(R1); COMMENT SAVE FOR VAR ADDR, INITIALIZE STACK;
29 02D8  07 0AFC   4643       IF R7 >= 0 THEN
29 02DE  07 0AFC   4644       BEGIN  R0 := 0;  STACKP4(R8) := R0; STACKP8(R8) := R0;
29 02EA  07 0AFC   4645          R0 := IDLOC1(R7) SHLL 12 + IDLOC2(R7);
29 02F6  07 0AFC   4646          R5 := R5 + 4;  LSTACKM4(R5) := R0;
29 02FE  07 0AFC   4647       END;
29 0300  07 0AFC   4648
29 0300  07 0AFC   4649             SAVER1 := R1;
29 0304  07 0AFC   4650             COMMENT ARG1;
29 0304  07 0AFC   4651             RESET(ARGFLAG);
```

```
29 03C8  07 0AFC  4652          CASE R3 OF
29 03C8  07 0AFC  4653          BEGIN
29 03C8  07 0AFC  4654             BEGIN  COMMENT   +  ;                                  1
29 03D0  07 0AFC  4655                CONVERT; LOADREG; R6 := R6 OR R7;
29 03D8  07 0AFC  4656             END;
29 03D8  07 0AFC  4657             BEGIN  COMMENT   -  ;                                  2
29 03DC  07 0AFC  4658                CONVERT; LOADREG; R6 := R6 OR R7;
29 03E4  07 0AFC  4659             END;
29 03E4  07 0AFC  4660             BEGIN  COMMENT   *  ;                                  3
29 03E8  07 0AFC  4661                CONVERT; SET(PAIRFLAG); LOADREG;
29 03F2  07 0AFC  4662                RESET(PAIRFLAG);
29 03F6  07 0AFC  4663                R6 := R6 OR R7;
29 03F8  07 0AFC  4664             END;
29 03F8  07 0AFC  4665             BEGIN  COMMENT   /  ;                                  4
29 03FC  07 0AFC  4666                CONVERT; SET(PAIRFLAG); SET(DIVIDE); LOADREG;
29 040A  07 0AFC  4667                RESET(PAIRFLAG); RESET(DIVIDE);
29 0412  07 0AFC  4668                R6 := R6 OR R7;
29 0414  07 0AFC  4669             END;
29 0414  07 0AFC  4670             BEGIN  COMMENT   **  ;                                 5
29 0418  07 0AFC  4671                CONVERT;  COMMENT ON BASE SIDE;
29 0422  07 0AFC  4672                LOADREG;  R6 := R6 OR R7;  STACKP4(R8) := R7;
29 0424  07 0AFC  4673             END;
29 0424  07 0AFC  4674             BEGIN  COMMENT   L:=  ;                                6
29 0428  07 0AFC  4675                LCOLEQARG1;
29 042C  07 0AFC  4676             END;
29 042C  07 0AFC  4677             BEGIN  COMMENT   A:=  ;                                7
29 0430  07 0AFC  4678                R10 := TREE(R6); IF R10 < 0 THEN
29 043A  07 0AFC  4679                BEGIN CONVERT; LOADREG; END;
29 043E  07 0AFC  4680                R6 := R6 OR R7;
29 0442  07 0AFC  4681          END;
29 0442  07 0AFC  4682          BEGIN  COMMENT   S:=  ;                                   8
29 0446  07 0AFC  4683             SCOLEQARG1;
29 044A  07 0AFC  4684          END;
29 044A  07 0AFC  4685          BEGIN  COMMENT   R:=  ;                                   9
29 044E  07 0AFC  4686             RCOLEQARG1;
29 0452  07 0AFC  4687          END;
29 0452  07 0AFC  4688          NULL;                                                     10
29 0456  07 0AFC  4689          NULL;                                                     11
29 045A  07 0AFC  4690          BEGIN  COMMENT   STEPUNTIL  ;                             12
29 045E  07 0AFC  4691             LOGICAL SAVER6;  SAVER6 := R6;  R6 := 0;
29 0466  07 0B00  4692             R1 := R7 SHRL 24;  IF R1 = NUMBER THEN
29 0474  07 0B00  4693             BEGIN  R1 := R7 AND #FFFF + CPTBASE;
29 047E  07 0B00  4694                R1 := CONSPTTAB(R1) AND #FFFF;  R6 := PROGRAMM(R1);
29 048A  07 0B00  4695             END;
29 048A  07 0B00  4696             IF R6 < 0 THEN
29 0490  07 0B00  4697             BEGIN  COMMENT  NEGATIVE CONSTANT;
29 0490  07 0B00  4698                R5 := R5+4;  R1 := R1 OR #E000; LSTACKM4(R5) := R1;
29 049C  07 0B00  4699                R0 := 1;  COMMENT TYPE 1;
29 04A0  07 0B00  4700             END ELSE
29 04A0  07 0B00  4701             BEGIN  LOADREG;  R10 := NEXTADDR;
29 04B4  07 0B00  4702                IF R6 > 0 THEN R10 := R10 + 7 AND #FFF8
29 04BE  07 0B00  4703                          ELSE R10 := R10 + 3 AND #FFFC;
29 04CE  07 0B00  4704                NEXTADDR := R10;  R3 := LSTACKM4(R5) AND #00F00000;
29 04DA  07 0B00  4705                LSTACKM4(R5) := R10;
29 04DE  07 0B00  4706                R0 := #50000000 OR R3 OR R10;  EMIT;
29 04E2  07 0B00  4707                IF R6 > 0 THEN R10 := 8 ELSE
29 04EC  07 0B00  4708                BEGIN  R0 := R3 SHRL 4 OR R3 OR #12000000; EMIT;
29 04B8  07 0B00  4709                   R0 := #41000020 OR R3;  EMIT;
```

```
29 04CA  07 0B00  4710                    R0 := #47ACE006 ++ INSCOUNTER;  EMIT;
29 04DE  07 0B00  4711                    R0 := R3 SHRL 4 OR R3 OR #1E000000; EMIT;
29 04F6  07 0B00  4712                    R0 := #50000008 OR R3 ++ R10; EMIT;  R10 := 12;
29 050E  07 0B00  4713                 END;
29 050E  07 0B00  4714                 R6 := 1; R3 := R3 OR SIGN; RELEASE;
29 0522  07 0B00  4715                 R0 := R10 SHRL 2;  COMMENT TYPES 2, 3; INCRADDR;
29 0534  07 0B00  4716              END;
29 0534  07 0B00  4717              R3 := R8 - SSIZE; STACKP4(R3) := R0; STACKP4(R8) := R0;
29 0542  07 0B00  4718              R6 := SAVER6;
29 0546  07 0B00  4719           END;
29 0546  07 0B00  4720           BEGIN  COMMENT   DIV  ;                                    13
29 054A  07 0B00  4721              R10 := TREE(R6); IF R10 > 0 THEN
29 0554  07 0B00  4722              BEGIN
29 0554  07 0B00  4723              SET(PAIRFLAG); SET(DIVIDE); LOADREG;
29 0568  07 0B00  4724              RESET(PAIRFLAG); RESET(DIVIDE);
29 0570  07 0B00  4725              END ELSE LOADREG;
29 0580  07 0B00  4726              R6 := R6 OR R7;
29 0582  07 0B00  4727           END;
29 0582  07 0B00  4728           BEGIN  COMMENT   REM  ;                                    14
29 0586  07 0B00  4729              R10 := TREE(R6); IF R10 > 0 THEN
29 0590  07 0B00  4730              BEGIN
29 0590  07 0B00  4731              SET(PAIRFLAG); SET(DIVIDE); LOADREG;
29 05A4  07 0B00  4732              RESET(PAIRFLAG); RESET(DIVIDE);
29 05AC  07 0B00  4733              R6 := R6 OR R7;
29 05AE  07 0B00  4734              END ELSE LOADREG;
29 05BE  07 0B00  4735           END;
29 05BE  07 0B00  4736           BEGIN  COMMENT   <  ;                                      15
29 05C2  07 0B00  4737              ARG1EQUALITIES;
29 05C6  07 0B00  4738           END;
29 05C6  07 0B00  4739           BEGIN  COMMENT   <=  ;                                     16
29 05CA  07 0B00  4740              ARG1EQUALITIES;
29 05CE  07 0B00  4741           END;
29 05CE  07 0B00  4742           BEGIN  COMMENT   >  ;                                      17
29 05D2  07 0B00  4743              ARG1EQUALITIES;
29 05D6  07 0B00  4744           END;
29 05D6  07 0B00  4745           BEGIN  COMMENT   >=  ;                                     18
29 05DA  07 0B00  4746              ARG1EQUALITIES;
29 05DE  07 0B00  4747           END;
29 05DE  07 0B00  4748           BEGIN  COMMENT   =  ;                                      19
29 05E2  07 0B00  4749              ARG1EQUALITIES;
29 05E6  07 0B00  4750           END;
29 05E6  07 0B00  4751           BEGIN  COMMENT   ¬=  ;                                     20
29 05EA  07 0B00  4752              ARG1EQUALITIES;
29 05EE  07 0B00  4753           END;
29 05EE  07 0B00  4754           NULL;                                                      21
29 05F2  07 0B00  4755           BEGIN  COMMENT   L:=2  ;                                   22
29 05F6  07 0B00  4756              LCOLEQARG1;
29 05FA  07 0B00  4757           END;
29 05FA  07 0B00  4758           BEGIN  COMMENT   A:=2  ;                                   23
29 05FE  07 0B00  4759              CONVERT;    LOADREG;
29 0614  07 0B00  4760           END;
29 0614  07 0B00  4761           BEGIN  COMMENT   S:=2  ;                                   24
29 0618  07 0B00  4762              SCOLEQARG1;
29 061C  07 0B00  4763           END;
29 061C  07 0B00  4764           BEGIN  COMMENT   R:=2  ;                                   25
29 0620  07 0B00  4765              RCOLEQARG1;
29 0624  07 0B00  4766           END;
29 0624  07 0B00  4767           NULL;                                                      26
```

```
29 0628  07 0B00  4768          NULL;                                                                    27
29 062C  07 0B00  4769          NULL;                                                                    28
29 0630  07 0B00  4770          BEGIN  COMMENT   AP)  ;                                                  29
29 0634  07 0B00  4771             IF R7 >= 0 THEN
29 063A  07 0B00  4772             BEGIN  R1 := R7 SHRL 24;
29 0640  07 0B00  4773                IF R1 = FUNCID THEN
29 0648  07 0B00  4774                BEGIN  COMMENT PROCEDURE CALL;
29 0648  07 0B00  4775                   R3 := 1;  PARAMETERS;
29 0656  07 0B00  4776                END ELSE IF R1 = STFUNCID THEN
29 0662  07 0B00  4777                BEGIN  R1 := R7 AND MASK;  R3 := R1 OR SIGN;
29 066E  07 0B00  4778                   STACKP8(R8) := R3;  R0 := R0 - R0;  R1 := R1 / 12;
29 0674  07 0B00  4779                END ELSE IF R1 = STPROCID THEN
29 0684  07 0B00  4780                BEGIN  R1 := R7 AND MASK OR SIGN;  STACKP8(R8) := R1;
29 068E  07 0B00  4781                   IF R1 = SREAD THEN
29 069A  07 0B00  4782                   BEGIN  R0 := MVII OR #FF0000 OR READFLAG;  EMIT;
29 06B2  07 0B00  4783                   END ELSE
29 06B2  07 0B00  4784                   IF R1 = SWRITE THEN
29 06BE  07 0B00  4785                   BEGIN  R0 := MVII OR #FF0000 OR WRITEFLAG; EMIT;
29 06D6  07 0B00  4786                   END;
29 06D6  07 0B00  4787                END;
29 06D6  07 0B00  4788             END;
29 06D6  07 0B00  4789          END;
29 06D6  07 0B00  4790          BEGIN  COMMENT   INDX  ;                                                 30
29 06DA  07 0B00  4791             R1 := R7 SHRL 24;  IF R1 = ARRAYID THEN
29 06E8  07 0B00  4792             BEGIN   R3 := @INITAREF;  ARRAYS;   END;
29 06F6  07 0B00  4793          END;
29 06F6  07 0B00  4794          BEGIN  COMMENT   REFX  ;                                                 31
29 06FA  07 0B00  4795             IF  R7 < 0 THEN
29 0700  07 0B00  4796             BEGIN  R0 := LSTACKM4(R5);
29 0704  07 0B00  4797                IF R0 > 0 THEN
29 070A  07 0B00  4798                BEGIN  R3 := R0 SHLL 8 AND MASK5;  R0 := R0 OR R3;
29 0716  07 0B00  4799                   R3 := R3 OR SIGN;  LSTACKM4(R5) := R3;
29 071E  07 0B00  4800                   R0 := R0 OR #58000000;  EMIT;
29 072E  07 0B00  4801                END;
29 072E  07 0B00  4802             END ELSE
29 072E  07 0B00  4803             LOADREG;
29 073E  07 0B00  4804             R7 := #89000000;
29 0742  07 0B00  4805          END;
29 0742  07 0B00  4806          BEGIN  COMMENT   IFEXP  ;                                                32
29 0746  07 0B00  4807             IFSTANDEXP;
29 0752  07 0B00  4808          END;
29 0752  07 0B00  4809          NULL;                                                                    33
29 0756  07 0B00  4810          NULL;                                                                    34
29 075A  07 0B00  4811          BEGIN  COMMENT   SHL  ;                                                  35
29 075E  07 0B00  4812             R10 := TREE(R6);
29 0762  07 0B00  4813             IF R10 < 0 THEN SHIFTAMOUNT ELSE LOADREG;
29 0784  07 0B00  4814             R6 := R6 OR R7;
29 0786  07 0B00  4815          END;
29 0786  07 0B00  4816          BEGIN  COMMENT   SHR  ;                                                  36
29 078A  07 0B00  4817             R10 := TREE(R6);
29 078E  07 0B00  4818             IF R10 < 0 THEN SHIFTAMOUNT ELSE LOADREG;
29 07B0  07 0B00  4819             R6 := R6 OR R7;
29 07B2  07 0B00  4820          END;
29 07B2  07 0B00  4821          BEGIN  COMMENT   BB  ;                                                   37
29 07B6  07 0B00  4822          END;
29 07B6  07 0B00  4823          BEGIN  COMMENT   END  ;                                                  38
29 07BA  07 0B00  4824          END;
29 07BA  07 0B00  4825          BEGIN  COMMENT   PCL  ;                                                  39
```

```
29 07BE  07 0B00  4826  END;
29 07BE  07 0B00  4827  BEGIN  COMMENT   SUBSTRIN  ;                                  40
29 07C2  07 0B00  4828  END;
29 07C2  07 0B00  4829  BEGIN  COMMENT   |  ;                                       41
29 07C6  07 0B00  4830     R3 := R7 SHRL 24;  IF R3 = NUMBER THEN
29 07D4  07 0B00  4831     BEGIN  R1 := CPTBASE;  R3 := R7 + R1;
29 07DC  07 0B00  4832        R3 := CONSPTTAB(R3) AND MASK;
29 07E4  07 0B00  4833        R0 := CONSPTTAB(R1) AND MASK;  R3 := R3 - R0;
29 07EE  07 0B00  4834        R3 := R3 + CONSTAB;  R3 := B3;
29 07F6  07 0B00  4835        IF R3 < 0 THEN R3 := 0 ELSE R3 := R3 OR SIGN;
29 0808  07 0B00  4836     END ELSE R3 := 0;
29 C810  07 0B00  4837     STACKP4(R8) := R3;
29 0814  07 0B00  4838     LOADREG;  R6 := R6 OR R7;
29 0822  07 0B00  4839  END;
29 0822  07 0B00  4840  BEGIN  COMMENT   AP,  ;                                     42
29 0826  07 0B00  4841     IF R7 >= 0 THEN
29 082C  07 0B00  4842     BEGIN  R1 := R7 SHRL 24;
29 0832  07 0B00  4843        IF R1 = FUNCID THEN
29 083A  07 0B00  4844        BEGIN  COMMENT PROCEDURE CALL;
29 083A  07 0B00  4845           R3 := 1;  PARAMETERS;
29 0848  07 0B00  4846        END ELSE IF R1 = STPROCID THEN
29 0854  07 0B00  4847        BEGIN  R1 := R7 AND MASK OR SIGN;  STACKP8(R8) := R1;
29 085E  07 0B00  4848           IF R1 = SREAD THEN
29 086A  07 0B00  4849           BEGIN  R0 := MVII OR #FF0000 OR READFLAG;  EMIT;
29 0882  07 0B00  4850           END ELSE
29 0882  07 0B00  4851           IF R1 = SWRITE THEN
29 088E  07 0B00  4852           BEGIN  R0 := MVII OR #FF0000 OR WRITEFLAG; EMIT;
29 08A6  07 0B00  4853           END;
29 08A6  07 0B00  4854        END;
29 08A6  07 0B00  4855     END;
29 08A6  07 0B00  4856  END;
29 08A6  07 0B00  4857  BEGIN  COMMENT   R,  ;                                      43
29 08AA  07 0B00  4858     IF R7 >= 0 THEN
29 08B0  07 0B00  4859     BEGIN  R3 := R7 AND MASK + 12;  STACKP4(R8) := R3;
29 08BE  07 0B00  4860        R3:=R3+12; STACKP4(R8+12) := R3;RECORDALLOCATE;
29 08D2  07 0B00  4861     END ELSE
29 08D2  07 0B00  4862     BEGIN  R3 :=    STACKP4(R8) + 12; STACKP4(R8+12):=R3;
29 08E2  07 0B00  4863     END;
29 08E2  07 0B00  4864     R7 := #89000000;
29 08E6  07 0B00  4865  END;
29 08E6  07 0B00  4866  BEGIN  COMMENT   AR,  ;                                     44
29 08EA  07 0B00  4867  END;
29 08EA  07 0B00  4868  BEGIN  COMMENT   AR)  ;                                     45
29 08EE  07 0B00  4869  END;
29 08EE  07 0B00  4870  BEGIN  COMMENT   R)  ;                                      46
29 08F2  07 0B00  4871     IF R7 >= 0 THEN
29 08F8  07 0B00  4872     BEGIN  R3 := R7 AND MASK + 12;  STACKP4(R8) := R3;
29 0906  07 0B00  4873        RECORDALLOCATE;
29 0912  07 0B00  4874     END;
29 0912  07 0B00  4875  END;
29 0912  07 0B00  4876  BEGIN  COMMENT   LOGOR  ;                                   47
29 0916  07 0B00  4877     RESET(ANDFLAG);  ANDORARG1;
29 0926  07 0B00  4878  END;
29 0926  07 0B00  4879  BEGIN  COMMENT   BITOR  ;                                   48
29 092A  07 0B00  4880     LOADREG; R6 := R6 OR R7;
29 0938  07 0B00  4881  END;
29 0938  07 0B00  4882  BEGIN  COMMENT   LOGAND  ;                                  49
29 093C  07 0B00  4883     SET(ANDFLAG);  ANDORARG1;
```

```
29 094C  07 0B00   4884          END;
29 094C  07 0B00   4885          BEGIN  COMMENT   BITAND  ;                    50
29 0950  07 0B00   4886             LOADREG; R6 := R6 OR R7;
29 095E  07 0B00   4887          END;
29 095E  07 0B00   4888          BEGIN  COMMENT   ITERST  ;                    51
29 0962  07 0B00   4889          END;
29 0962  07 0B00   4890          BEGIN  COMMENT   ITERST2  ;                   52
29 0966  07 0B00   4891          END;
29 0966  07 0B00   4892          BEGIN  COMMENT   FORLIST  ;                   53
29 096A  07 0B00   4893             ARG1FORLIST;   R6 := R6 OR #81000000;
29 0972  07 0B00   4894          END;
29 0972  07 0B00   4895          BEGIN  COMMENT   FORCL  ;                     54
29 0976  07 0B00   4896          END;
29 0976  07 0B00   4897          BEGIN  COMMENT   ENDFORLI  ;                  55
29 097A  07 0B00   4898             ARG1FORLIST;
29 097E  07 0B00   4899          END;
29 097E  07 0B00   4900          BEGIN  COMMENT   UJIFEXP  ;                   56
29 0982  07 0B00   4901          END;
29 0982  07 0B00   4902          BEGIN  COMMENT   UJ  ;                        57
29 0986  07 0B00   4903          END;
29 0986  07 0B00   4904          BEGIN  COMMENT   CL  ;                        58
29 098A  07 0B00   4905          END;
29 098A  07 0B00   4906          BEGIN  COMMENT   IFST  ;                      59
29 098E  07 0B00   4907             IFSTANDEXP;
29 099A  07 0B00   4908          END;
29 099A  07 0B00   4909          BEGIN  COMMENT   ::  ;                        60
29 099E  07 0B00   4910             LOADREG;        R6 := R6 OR R7;
29 09AC  07 0B00   4911             R3 := @LBOUND;     ARRAYS;
29 09BA  07 0B00   4912          END;
29 09BA  07 0B00   4913          BEGIN  COMMENT   IS  ;                        61
29 09BE  07 0B00   4914             IF R7 >= 0 THEN
29 09C4  07 0B00   4915             BEGIN
29 09C4  07 0B00   4916                GETADDRESS;  IF R3 = #3000 THEN
29 09D8  07 0B00   4917                BEGIN  GENREG;  R1 := R0 SHRL 8;  LSTACKM4(R5) := R1;
29 09EA  07 0B00   4918                   R0 := R0 OR LAA OR #3000;  EMIT;
29 0A02  07 0B00   4919                END ELSE
29 0A02  07 0B00   4920                BEGIN  LSTACK(R5) := R3;  R5 := R5 + 4;
29 0A0E  07 0B00   4921                END;
29 0A0E  07 0B00   4922             END;
29 0A0E  07 0B00   4923          END;
29 0A0E  07 0B00   4924          NULL;                                         62
29 0A12  07 0B00   4925          BEGIN  COMMENT   ,  ;                         63
29 0A16  07 0B00   4926          END;
29 0A16  07 0B00   4927          BEGIN  COMMENT   WHILEOP  ;                   64
29 0A1A  07 0B00   4928             R0 := INSCOUNTER;  STACKP4(R8+12) := R0;
29 0A22  07 0B00   4929          END;
29 0A22  07 0B00   4930          BEGIN  COMMENT   WHILEST  ;                   65
29 0A26  07 0B00   4931          END;
29 0A26  07 0B00   4932          BEGIN  COMMENT   IFJ  ;                       66
29 0A2A  07 0B00   4933             MARKRECORDS;  DUMPALLGENREG;  DUMPALLFLREG;
29 0A4E  07 0B00   4934          END;
29 0A4E  07 0B00   4935       END;
29 0B5A  07 0B00   4936    R1 := SAVER1;
29 0B5E  07 0B00   4937 END;

                  SEGMENT 29  NAME = SEG#29      LENGTH = 0BE8  BASE REG = 15

21 0960  07 0B00   4938
```

```
21 0960   07 0B00          4939
21 0960   07 0B00          4940     SEGMENT PROCEDURE OUTPUTSEG(R1);
21 0960   07 0B00          4941        BEGIN  COMMENT  PRODUCE OBJECT MODULE FOR SEGMENT;
30 0000   07 0B00          4942               COMMENT  SEE SRL FORM Y28-6610 FOR FORMATS AND CODES;
30 0000   07 0B00          4943           ARRAY 5 LOGICAL SAVERS;
30 0000   07 0B14          4944           ARRAY 20 INTEGER CARD;
30 0000   07 0B64          4945            ARRAY 40 SHORT INTEGER SCARD SYN CARD;
30 0000   07 0B64          4946           SHORT INTEGER SB2 SYN B2;
30 0000   07 0B64          4947           BYTE SD SYN 0, ER SYN 2;
30 0000   07 0B64          4948           ARRAY 2 INTEGER PTLIMS;
30 0000   07 0B6C          4949            INTEGER PTLOW SYN PTLIMS(0);  COMMENT START ADDRESS OF FIELD;
30 0000   07 0B6C          4950            INTEGER PTHIGH SYN PTLIMS(4); COMMENT START ADDRESS OF SEQNO;
30 0000   07 0B6C          4951           ARRAY 3 SHORT INTEGER MVTREE=(#D200S,@B1,@B2),
30 0000   07 0B72          4952              MVTXT=(#D200S,@CARD(16),@B1), MVCARD=(#D200S,@CARD(16),@B3);
30 0000   07 0B7E          4953
30 0000   07 0B7E          4954           PROCEDURE MOVETREE(R1);  COMMENT MOVES TREE TO RECOVER SPACE;
30 0000   07 0B7E          4955             IF COMPACTED THEN
30 000C   07 0B7E          4956             BEGIN  R3 := 8;  EROR;
30 001C   07 0B7E          4957             END ELSE
30 001C   07 0B7E          4958             BEGIN  ARRAY 3 LOGICAL SAVE13; STM(R1,R3,SAVE13); SET(COMPACTED);
30 0024   07 0B8C          4959                R0 := TREELINK;  IF R0 < 0 THEN R12 := COMSTART ELSE
30 0036   07 0B8C          4960                BEGIN  R1 := RUNERRSTART + 1024;
30 0042   07 0B8C          4961                   IF R1 < RUNERRPT THEN R1 := RUNERRPT;  R0 := TREEBASE - R1;
30 0052   07 0B8C          4962                   COMMENT  R0 = SIZE OF FREE SPACE (MUST BE POSITIVE);
30 0054   07 0B8C          4963                   R3 := R12 - TREEBASE;  COMMENT # BYTES TO MOVE;
30 005A   07 0B8C          4964                   R2 := TREEBASE;  TREEBASE := R1;
30 0062   07 0B8C          4965                   WHILE R3 > 256 DO
30 006A   07 0B8C          4966                   BEGIN  MVC(255,B1,B2);  R3 := R3 - 256;
30 0074   07 0B8C          4967                      R1 := @B1(256);  R2 := @B2(256);
30 007C   07 0B8C          4968                   END;
30 0080   07 0B8C          4969                   DECR(R3);  EX(R3,MVTREE);  R1 := @TREELINK;
30 008A   07 0B8C          4970                   WHILE R1 > 0 DO    COMMENT FIX-UP TREE LINKS;
30 0090   07 0B8C          4971                   BEGIN  R2 := MEM(R1) - R0;  MEM(R1) := R2;  R1 := R2;
30 009C   07 0B8C          4972                   END;
30 00A0   07 0B8C          4973                   R12 := R12 - R0;
30 00A2   07 0B8C          4974                END;
30 00A2   07 0B8C          4975                R0 := @CARD; R1 := R12; R3 := APUTCARD; BALR(R2,R3);
30 00AE   07 0B8C          4976                LM(R1,R3,SAVE13);
30 00B2   07 0B8C          4977             END;
30 00B4   07 0B8C          4978
30 00B4   07 0B8C          4979           PROCEDURE CARDOUT(R1);  COMMENT SET LENGTH FIELD, PUNCH, CLEAR;
30 00B4   07 0B8C          4980             BEGIN  INTEGER SAVER1, SAVER3;  SAVER1 := R1; SAVER3 := R3;
30 00BC   07 0B94          4981                IF R2 ¬= PTLOW THEN
30 00C4   07 0B94          4982                BEGIN  R0 := R2 - PTLOW; SCARD(10) := R0;
30 00CE   07 0B94          4983                   R0 := SEQNO+1; SEQNO := R0;  CVD(R0,PKDEC);
30 00DE   07 0B94          4984                   UNPK(4,2,CARD(75),PKDEC(5)); OI("0",CARD(79));
30 00E8   07 0B94          4985                   R0 := @CARD; R1 := R12;  R3 := APUTCARD; BALR(R2,R3);
30 00F4   07 0B94          4986                   WHILE R0 < 0 DO MOVETREE;  COMMENT ATTEMPT ERROR RECOVERY;
30 0102   07 0B94          4987                END;
30 0102   07 0B94          4988                MVI(" ",CARD(16)); MVC(54,CARD(17),CARD(16)); R2 := @CARD(16);
30 010C   07 0B94          4989                R1 := SAVER1; R3 := SAVER3;
30 0118   07 0B94          4990             END;
30 011A   07 0B94          4991
30 011A   07 0B94          4992           PROCEDURE SETNAME(R1);  COMMENT R0 - SEGNO, NAME TO B2;
30 011A   07 0B94          4993             IF R0 = 0 THEN MVC(7,B2,"AWXRCTBL") ELSE
30 0126   07 0B94          4994             BEGIN  MVC(3,B2,"AWXS");
30 0130   07 0B94          4995                IF R0 < 0 THEN MVI("L",B2(4)) ELSE MVI("C",B2(4));
30 0142   07 0B94          4996                R0 := ABS R0; CVD(R0,PKDEC); UNPK(2,1,B2(5),PKDEC(6));
```

```
30 014E  07 0B94  4997          OI("0",B2(7));
30 0152  07 0B94  4998        END;
30 0154  07 0B94  4999
30 0154  07 0B94  5000        STM(R0,R4,SAVERS);  MVC(7,CARD(72),"       ");
30 015E  07 0B94  5001        MVI(#02,CARD);  R2 := @CARD(16); R3 := @CARD(72);
30 016A  07 0B94  5002        STM(R2,R3,PTLIMS);
30 016E  07 0B94  5003        R0 := CODELENGTH + INSCOUNTER;  CODELENGTH := R0;
30 017A  07 0B94  5004        MVC(3,CARD(1),"ESD "); MVC(66,CARD(5),CARD(4)); R2 := @CARD(16);
30 018A  07 0B94  5005        R4 := 1;  SCARD(14) := R4;
30 0192  07 0B94  5006        R0 := LOGSEG; SETNAME;  R0 := 0; B2(8) := R0; MVI(SD,B2(8));
30 01A6  07 0B94  5007        R0 := INSCOUNTER; R1 := LOGSEG;  IF CARDFLAG AND R1 ¬= 0 THEN
30 01BC  07 0B94  5008        BEGIN  R1 := RUNERRSTART;  R0 := R0 + B1(0);
30 01C4  07 0B94  5009        END;
30 01C4  07 0B94  5010        B2(12) := R0;  MVI(" ",B2(12));  R2 := @B2(16);
30 01D0  07 0B94  5011        FOR R3 := 4 STEP 4 UNTIL SEGTABIDX DO
30 01D4  07 0B94  5012        BEGIN  R0 := SEGNO(R3); IF R0 ¬= LOGSEG THEN
30 01E4  07 0B94  5013           BEGIN  R4 := R4 + 1; R0 := @B2(16); IF R0 > PTHIGH THEN
30 01F4  07 0B94  5014              BEGIN  CARDOUT;  SCARD(14) := R4;
30 01FC  07 0B94  5015              END;
30 01FC  07 0B94  5016              R0 := SEGNO(R3); SETNAME;  R0 := 0; B2(8) := R0;
30 020C  07 0B94  5017              MVI(ER,B2(8));  R2 := @B2(16);
30 0214  07 0B94  5018           END;
30 0214  07 0B94  5019        END;
30 0220  07 0B94  5020        CARDOUT;
30 0224  07 0B94  5021        MVC(3,CARD(1),"TXT "); MVC(8,CARD(5),CARD(4)); R3 := 0;
30 0234  07 0B94  5022        WHILE R3 < INSCOUNTER DO
30 023C  07 0B94  5023        BEGIN  R0 := 1;  SCARD(14) := R0;
30 0244  07 0B94  5024           CARD(4) := R3;  MVI(" ",CARD(4)); R0 := INSCOUNTER - R3;
30 0252  07 0B94  5025           IF R0 > 56 THEN R0 := 56;  SCARD(10) := R0;
30 0262  07 0B94  5026           R4 := R0-1; R1 := @PROGRAM(R3); EX(R4,MVTXT); R3 := R3+R0;
30 0272  07 0B94  5027           R2 := R2+R0;  CARDOUT;
30 0278  07 0B94  5028        END;
30 027C  07 0B94  5029        R0 := LOGSEG; IF R0 ¬= 0 THEN
30 0286  07 0B94  5030        BEGIN  R3 := RUNERRSTART;
30 028A  07 0B94  5031           WHILE R3 < RUNERRPT DO
30 0292  07 0B94  5032           BEGIN  R0 := 1; SCARD(14) := R0;
30 029A  07 0B94  5033              R0 := R3 - RUNERRSTART + INSCOUNTER; CARD(4) := R0;
30 02A8  07 0B94  5034              MVI(" ",CARD(4));  R0 := RUNERRPT - R3;
30 02B2  07 0B94  5035              IF R0 > 56 THEN R0 := 56;  SCARD(10) := R0;
30 02C2  07 0B94  5036              R4 := R0 - 1; EX(R4,MVCARD);  R3 := R3 + R0;
30 02CE  07 0B94  5037              R2 := R2 + R0;  CARDOUT;
30 02D4  07 0B94  5038           END;
30 02D8  07 0B94  5039        END;
30 02D8  07 0B94  5040        R0 := LOGSEG; IF R0 ¬= 0 THEN
30 02E2  07 0B94  5041        BEGIN  MVC(3,CARD(1),"RLD "); MVC(10,CARD(5),CARD(4));
30 02EE  07 0B94  5042           R0 := 1; SB2(0) := R0; SB2(2) := R0;
30 02FA  07 0B94  5043           R1 := 8; B2(4) := R1; MVI(#0C,B2(4)); R2 := @B2(8);
30 030A  07 0B94  5044           R0 := INSCOUNTER;  IF R0 > 4096 THEN
30 0316  07 0B94  5045           BEGIN  R0 := 1;  SB2(0) := R0; SB2(2) := R0;
30 0322  07 0B94  5046              R1 := 12; B2(4) := R1; MVI(#0C,B2(4)); R2 := @B2(8);
30 0332  07 0B94  5047           END;
30 0332  07 0B94  5048           R4 := 1;
30 0336  07 0B94  5049           FOR R3 := 4 STEP 4 UNTIL SEGTABIDX DO
30 033A  07 0B94  5050           BEGIN  R0 := @B2(8); IF R0 > PTHIGH THEN CARDOUT;
30 034E  07 0B94  5051              R0 := 1; SB2(2) := R0; R0 := CHAIN(R3); B2(4) := R0;
30 035E  07 0B94  5052              R0 := SEGNO(R3); IF R0 = LOGSEG THEN
30 036A  07 0B94  5053              BEGIN  MVI(#0C,B2(4)); R0 := 1;
30 0372  07 0B94  5054              END ELSE
```

```
30 0372  07 0B94   5055              BEGIN  MVI(#1C,B2(4));  R4 := R4 + 1; R0 := R4;
30 0380  07 0B94   5056              END;
30 038C  07 0B94   5057              SB2(0) := R0;    R2 := @B2(8);
30 0388  07 0B94   5058           END;
30 0394  07 0B94   5059           CARDOUT;
30 0398  07 0B94   5060        END;
30 0398  07 0B94   5061        MVC(3,CARD(1),"END "); MVC(10,CARD(5),CARD(4));
30 03A4  07 0B94   5062        R0 := "  "; R2 := R2 + R0;  COMMENT FORCE BLANK COUNT; CARDOUT;
30 03AE  07 0B94   5063        LM(R0,R4,SAVERS);
30 03B2  07 0B94   5064     END;

                  SEGMENT 30  NAME = SEG#30      LENGTH = 0400  BASE REG = 15

21 0960  07 0B94   5065
21 0960  07 0B94   5066  COMMENT  * * *  EXECUTION  * * * ;
21 0960  07 0B94   5067
21 0960  07 0B94   5068     INTEGER STACKORG;  BYTE FINISHED;  COMMENT USED IN TREE PROCESSING;
21 0960  07 0B99   5069     ARRAY 65 LOGICAL WORKAREA SYN MEM(R14);
21 0960  07 0B99   5070     INTEGER SAVER1;  SAVER1 := R1;
21 0964  07 0BA0   5071     COMMENT SEARCH NAMETABLE TO BUILD RECTABLE IN WORKAREA;
21 0964  07 0BA0   5072     R1:=0; R9:=R1;  FOR R3 := 0 STEP 4 UNTIL 256 DO WORKAREA(R3) := R1;
21 0982  07 0BA0   5073     FOR R3 := 0 STEP 12 UNTIL NAMETABLESIZE DO
21 0986  07 0BA0   5074     BEGIN SHORT INTEGER RECLEN SYN WORKAREA(8), NUMREF SYN WORKAREA(10);
21 098A  07 0BA0   5075        R1 := 0; IC(R1,TYPE(R3)); IF R1 = 4 THEN
21 099A  07 0BA0   5076        BEGIN  COMMENT RECORD CLASS;
21 099A  07 0BA0   5077           R8 := TYPEINFO(R3) AND #F SHLL 4; COMMENT REC CLASS # * 16;
21 09A6  07 0BA0   5078           IF R8 > R9 THEN R9 := R8;
21 09AE  07 0BA0   5079           R1 := SIMTYPEINFO(R3);  RECLEN(R8) := R1; R1 := 0;
21 09BA  07 0BA0   5080           R2 := TYPEINFO(R3) SHRL 8 AND #FF * 12S + R3; R0 := 0;
21 09D0  07 0BA0   5081           FOR R6 := R3+12 STEP 12 UNTIL R2 DO
21 09D6  07 0BA0   5082           BEGIN  IC(R0,SIMPLETYPE(R6));  IF R0 = 9 THEN R1 := R1 + 1;
21 09EA  07 0BA0   5083           END;
21 09F4  07 0BA0   5084           NUMREF(R8) := R1;
21 09F8  07 0BA0   5085        END;
21 09F8  07 0BA0   5086     END;
21 0AC4  07 0BA0   5087     R0 := #FFFFFFFF; WORKAREA(R9+16) := R0; COMMENT MARK END OF RECTAB;
21 0A0C  07 0BA0   5088     R0 := 0; LOGSEG := R0; SEGTABIDX := R0; R0 := 256; INSCOUNTER := R0;
21 0A1C  07 0BA0   5089     R12 := TREETOP;  OUTPUTSEG;  R14 := SAVE14;
21 0A32  07 0BA0   5090     R1 := REFRECBASE; RUNERRSTART := R1;  R2 := TREELINK;
21 0A3E  07 0BA0   5091     WHILE R2 >= 0 DO
21 0A44  07 0BA0   5092     BEGIN  R1 := MEM(R2); TREELINK := R1;  R1 := @MEM(R2+12);
21 0A50  07 0BA0   5093        R12 := R1;  COMMENT ESTABLISH TREE BASE;
21 0A52  07 0BA0   5094        R1 := R1 + MEM(R1) + 4;  COMMENT CONSTANT POINTER BASE;
21 0A5A  07 0BA0   5095        R0 := MEM(R1+4) + 7 AND #FFF8;  LITORG := R0;
21 0A6A  07 0BA0   5096        R0 := @MEM(R1+8);  CPTBASE := R0;
21 0A72  07 0BA0   5097        R0 := MEM(R1) - 4;  CONSPTTABL := R0;
21 0A7E  07 0BA0   5098        R1 := R1 + MEM(R1) + 4;  COMMENT CONSTANT TABLE BASE;
21 0A86  07 0BA0   5099        R0 := @MEM(R1+4); CONSTAB := R0;
21 0A8E  07 0BA0   5100        R0 := MEM(R1);  CONSTABL := R0;
21 0A96  07 0BA0   5101        R1 := RUNERRSTART; MVC(7,B1(4),B2(4));  COMMENT PROC ID;
21 0AA0  07 0BA0   5102        R1 := @B1(12);  RUNERRPT := R1;
21 0AA8  07 0BA0   5103        R2 := 0; R4 := 0; R5 := 0; LPOINTER := R5;
21 0AB8  07 0BA0   5104        R8 := STACKTOP-12; STACKORG := R8;  R0 := SIGN; STACK(R8) := R0;
21 0ACC  07 0BA0   5105        R9 := TREE(0);  RESET(FINISHED);
21 0AD4  07 0BA0   5106
21 0AD4  07 0BA0   5107      COMMENT * * *  TREE TRANSVERSAL ALGORITHM  * * *;
21 0AD4  07 0BA0   5108        WHILE ¬FINISHED DO
21 0ADC  07 0BA0   5109        BEGIN  R10 := TREE(R9);
```

```
21 0AE0  07 0BA0   5110          IF R10 < TERMINALNODE THEN
21 0AE8  07 0BA0   5111          BEGIN  COMMENT NON-TERMINAL, TREE SWITCH IS SIGN BIT;
21 0AE8  07 0BA0   5112             R8 := R8 + SSIZE;  IF R8 <= INSCOUNTER THEN
21 0AF4  07 0BA0   5113             BEGIN  R3 := 2; EROR;
21 0BC4  07 0BA0   5114             END;
21 0BC4  07 0BA0   5115             IF R10 < 0 THEN
21 0BCA  07 0BA0   5116             BEGIN  COMMENT BINARY OPERATOR, TREE SWITCH IS ON;
21 0BDA  07 0BA0   5117                STACK(R8) := R9;  R9 := R9 - 4;
21 0BE2  07 0BA0   5118             END ELSE
21 0BE2  07 0BA0   5119             IF R10 < UNARYOP THEN
21 0BEE  07 0BA0   5120             BEGIN  COMMENT BINARY OPERATOR, TREE SWITCH IS OFF;
21 0BFE  07 0BA0   5121                STACK(R8) := R9;  R9 := TREEP(R9);
21 0C26  07 0BA0   5122             END ELSE
21 0C26  07 0BA0   5123             BEGIN  COMMENT UNARY OPERATOR;
21 0C2A  07 0BA0   5124                R10 := R9 OR SIGN; STACK(R8) := R10;  R9 := R9 - 4;
21 0C38  07 0BA0   5125             END;
21 0C38  07 0BA0   5126          END ELSE
21 0C38  07 0BA0   5127          BEGIN  COMMENT TERMINAL NODE - INITIATES PROCESSING;
21 0C3C  07 0BA0   5128             R7 := R10; R6 := STACK(R8); R5 := STACKP(R8);
21 0C46  07 0BA0   5129             WHILE R6 < 0 AND ¬FINISHED DO
21 0C54  07 0BA0   5130             BEGIN  COMMENT ARG2 PROCESSING;
21 0C54  07 0BA0   5131                IC(R3,TREE(R5)); R3 := R3 AND #7F; COMMENT OPERATOR;
21 0C5C  07 0BA0   5132                SAVEL := R5; R5 := LPOINTER;  COMMENT LSTACK INDEX;
21 0C64  07 0BA0   5133                IF R3 <= 34 THEN ARG21 ELSE
21 0C76  07 0BA0   5134                IF R3 <= 60 THEN
21 0C82  07 0BA0   5135                BEGIN  R3 := R3 - 34;  ARG22;
21 0C90  07 0BA0   5136                END ELSE
21 0C90  07 0BA0   5137                BEGIN  R3 := R3 - 60;  ARG23;
21 0CA2  07 0BA0   5138                END;
21 0CA2  07 0BA0   5139                LPOINTER := R5;
21 0CA6  07 0BA0   5140                R8 := R8 - SSIZE; IF R8 = STACKORG THEN SET(FINISHED);
21 0CB6  07 0BA0   5141                R9 := STACKP(R8);  R6 := STACK(R8); R5 := R9;
21 0CC0  07 0BA0   5142             END;
21 0CC4  07 0BA0   5143             IF R6 >= 0 THEN
21 0CCA  07 0BA0   5144             BEGIN  COMMENT ARG1 PROCESSING, STACK NOT EMPTY;
21 0CCA  07 0BA0   5145                IC(R3,TREE(R5)); R3 := R3 AND #7F;  COMMENT OPERATOR;
21 0CD2  07 0BA0   5146                SAVEL := R5;  R5 := LPOINTER;  COMMENT LSTACK INDEX;
21 0CDA  07 0BA0   5147                ARG1;  LPOINTER := R5;  R5 := SAVEL;
21 0CEC  07 0BA0   5148                R10 := SIGN OR R6; STACK(R8) := R10;
21 0CF6  07 0BA0   5149                R10 := TREE(R5);
21 0CFA  07 0BA0   5150                IF R10 >= 0 THEN R9 := R5-4 ELSE R9 := TREEP(R5);
21 0D0E  07 0BA0   5151             END;
21 0D0E  07 0BA0   5152          END;
21 0D0E  07 0BA0   5153       END;
21 0D12  07 0BA0   5154
21 0D12  07 0BA0   5155       R1 := RUNERRPT; R0 := INSCOUNTER SHLL 16; B1(0) := R0;
21 0D22  07 0BA0   5156       R1 := @B1(4); RUNERRPT := R1; R0 := R1;
21 0D2C  07 0BA0   5157       R1 := RUNERRSTART;  R0 := R0-R1; B1(0) := R0;
21 0D36  07 0BA0   5158       OUTPUTSEG;  IC(R1,TRACEIT); R1 := R1 AND #F;
21 0D48  07 0BA0   5159       IF R1 ¬= 0 AND R1 ¬= 6 THEN
21 0D56  07 0BA0   5160       BEGIN  IF R1 ¬= 3 THEN MVI("1",CARRCONT);
21 0D62  07 0BA0   5161          MVC(7,OWBUF(4),"SEGMENT "); MVC(51,OWBUF(12),OWBUF(11));
21 0D6E  07 0BA0   5162          R0 := LOGSEG;  CVD(R0,PKDEC); UNPK(1,7,OWBUF(13),PKDEC);
21 0D7C  07 0BA0   5163          OI("0",OWBUF(14));  R0 := @OWBUF;  WRITE;
21 0D90  07 0BA0   5164          WORK := R2; UNPK(8,4,OWBUF(1),WORK);  WORK := R4;
21 0D9E  07 0BA0   5165          UNPK(8,4,OWBUF(10),WORK);  R5 := LPOINTER; WORK := R5;
21 0DAC  07 0BA0   5166          UNPK(8,4,OWBUF(19),WORK);  TR(25,OWBUF(1),OTRTABLE(_240));
21 0DB8  07 0BA0   5167          MVI(" ",OWBUF(9)); MVI(" ",OWBUF(18));  WRITE;
```

```
21 0CCC   07 0BA0   5168            MVC(62,OWBUF(1),OWBUF); OI("0",CARRCONT); R14 := SAVE14;
21 0CDA   07 0BA0   5169            IF R1 ¬= 3 THEN WRITEALL;
21 0CEE   07 0BA0   5170         END;
21 0CEE   07 0BA0   5171         R2 := TREELINK;
21 0CF2   07 0BA0   5172      END;
21 0CF6   07 0BA0   5173      R1 := SAVER1;
21 0CFA   07 0BA0   5174      END;

                   SEGMENT 21  NAME = SEG#21      LENGTH = 0DA8  BASE REG = 15

11 0BD8   07 0BA0   5175      SAVER1 := R1;  LEVEL2;  R1 := SAVER1;
11 0BEA   07 0BA0   5176      END;

                   SEGMENT 11  NAME = SEG#11      LENGTH = 0CA8  BASE REG = 15

10 09B4   07 0BA0   5177      SAVER1 := R1;  LEVEL1;  R1 := SAVER1;
10 09C6   07 0BA0   5178      END;

                   SEGMENT 10  NAME = SEG#10      LENGTH = 0A78  BASE REG = 15

09 0954   07 0BA0   5179      SAVER1 := R1;  LEVEL0;  R1 := SAVER1;
09 0966   07 0BA0   5180      END;

                   SEGMENT 09  NAME = SEG#09      LENGTH = 0A00  BASE REG = 15

06 0A1C   07 0BA0   5181
06 0A1C   07 0BA0   5182    SEGMENT  BASE  R3;
06 0A20   31 0000   5183     ARRAY  256  INTEGER MNEMONIC
06 0A20   31 0000   5184     =("****************SPM BALRBCTRBCR ********************************"
06 0A20   31 0400   5185     ,"LPR LNR LTR LCR NR  CLR OR  XR  LR  CR  AR  SR  MR  DR  ALR SLR "
06 0A20   31 0400   5186     ,"LPDRLNDRLTDRLCDRHDR ***********LDR CDR ADR SDR MDR DDR AWR SWR "
06 0A20   31 0400   5187     ,"LPERLNERLTERLCERHER ***********LER CER AER SER MER DER AUR SUR "
06 0A20   31 0400   5188     ,"STH LA  STC IC  EX  BAL BCT BC  LH  CH  AH  SH  MH  ****CVD CVB "
06 0A20   31 0400   5189     ,"ST  ***********N   CL  O   X   L   C   A   S   M   D   AL  SL  "
06 0A20   31 0400   5190     ,"STD ****************************LD  CD  AD  SD  MD  DD  AW  SW  "
06 0A20   31 0400   5191     ,"STE ****************************LE  CE  AE  SE  ME  DE  AU  SU  "
06 0A20   31 0400   5192     ,"************************BXH BXLESRL SLL SRA SLA SRDLSLDLSRDASLDA"
06 0A20   31 0400   5193     ,"STM TM  MVI TS  NI  CLI OI  XI  LM  ******************************"
06 0A20   31 0400   5194     ,"************************************************************"
06 0A20   31 0400   5195     ,"************************************************************"
06 0A20   31 0400   5196     ,"************************************************************"
06 0A20   31 0400   5197     ,"****MVN MVC MVZ NC  CLC OC  XC  ****************TR  TRT ED  EDMK"
06 0A20   31 0400   5198     ,"************************************************************"
06 0A20   31 0400   5199     ,"****MVO PACKUNPK***************ZAP CP  AP  SP  MP  DP  ********"
06 0A20   31 0400   5200     );
06 0A20   31 0400   5201     MNBASE := R3;  OLDSAVE := R5;
06 0A28   31 0400   5202     MVC(63,XFERVECTOR,B1);  R1 := ARUNID; MVC(31,HEADING(80),B1);
06 0A38   31 0400   5203     R0 := #2400; R1 := #4800; R3 := AGETMAIN; BALR(R2,R3);
06 0A46   31 0400   5204     R11 := COMSTART;  R14 := R0; SAVE14 := R14; STACKTOP := R1;
06 0A54   31 0400   5205     LEVELA;  R0 := 0;
06 0A62   31 0400   5206   ERXIT:
06 0A62   31 0400   5207     R6 := R0;  MVI("0",CARRCONT);  MVC(54,OWBUF(6),
06 0A63   31 0400   5208       " SECONDS IN COMPILATION, XXXXX BYTES OF CODE GENERATED ");
06 0A6E   31 0400   5209     MVC(70,OWBUF(61),OWBUF(60));
06 0A74   31 0400   5210     R3 := AGETTIME; BALR(R2,R3);  R1 := R0 -- COMMTIME; R1 := R1*5/1920;
06 0A84   31 0400   5211     CVD(R1,PKDEC); UNPK(4,2,OWBUF(1),PKDEC(5));
06 0A92   31 0400   5212     MVC(2,OWBUF(0),OWBUF(1)); MVI(".",OWBUF(3)); OI("0",OWBUF(5));
06 0AA0   31 0400   5213     R0 := CODELENGTH; CVD(R0,PKDEC); UNPK(4,2,OWBUF(31),PKDEC(5));
```

```
06 0AAE   31 0400          5214       OI("0",OWBUF(35));  IF COMPACTED THEN MVI("*",OWBUF(61));
06 0ABE   31 0400          5215       R0 := @OWBUF;  WRITE;
06 0AC6   31 0400          5216       R0 := SAVE14;  R1 := STACKTOP;  R3 := AFREEMAIN; BALR(R2,R3);
06 0AD4   31 0400          5217       R13 := OLDSAVE;   R15 := R6;
06 0ADA   31 0400          5218    END;

                          SEGMENT 31  NAME = SEG#31      LENGTH = 0400  BASE REG = 03

                          SEGMENT 07  NAME = SEG#07      LENGTH = 0BA0  BASE REG = 13

06 0ADA   00 0000          5219    R14 := B13(12);  LM(R0,R12,B13(20));
06 0AE2   00 0000          5220   END.

                          SEGMENT 06  NAME = AWXCMPB1    LENGTH = 0CA8  BASE REG = 15
```

```
IEF285I   T123.PLLIB                                  PASSED
IEF285I   VOL SER NOS= SYS11 .
IEF285I   SYSOUT                                      SYSOUT
IEF285I   VOL SER NOS=       .
IEF285I   AWXCMPB1                                    KEPT
IEF285I   VOL SER NOS= WIR001.
IEF280I K OC2,WIR001,TOTAPE
IEF285I   T123.PLLIB                                  KEPT
IEF285I   VOL SER NOS= SYS11 .
```

```
H A S P  JOB STATISTICS --  5,231 CARDS READ --    5,364 LINES PRINTED --       0 CARDS PUNCHED --   0.81 MINUTES EXECUTION TIME
          43 I/O CALLS              113 SVC CALLS          0.41 MINUTES CPU TM    TIME=  00:08:43 DATE=  11/22/69
```

Moore Business Forms, Inc. sv