27
2-17-76
sup to NTIS

IS-3800

REFERENCE MANUAL FOR THE SDS-910 CROSS-ASSEMBLER

MASTER

Kent R. Brobst,
Leonard C. Moon, and
Barbara J. Helland

AMES LABORATORY, USERDA
IOWA STATE UNIVERSITY
AMES, IOWA

Date Transmitted: January 1976

PREPARED FOR THE U. S. ENERGY RESEARCH AND DEVELOPMENT ADMINISTRATION
UNDER CONTRACT W-7405-eng-82

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

IS-3800

REFERENCE MANUAL FOR THE SDS-910 CROSS-ASSEMBLER

By

Kent R. Brobst, Leonard C. Moon,
and Barbara J. Helland

NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Energy Research and Development Administration, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

Ames Laboratory, ERDA
Iowa State University
Ames, Iowa 50011

Date Transmitted: January 1976

PREPARED FOR THE U.S. ENERGY RESEARCH AND
DEVELOPMENT ADMINISTRATION UNDER CONTRACT NO. W-7405-eng-82

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Energy Research and Development Administration, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

Available from: National Technical Information Service
U. S. Department of Commerce
P.O. Box 1553
Springfield, VA 22161

Price: Microfiche $2.25
Paper Copy $5.00

TABLE OF CONTENTS

	PAGE
ABSTRACT	iv
INTRODUCTION	1
METHOD OF SOLUTION FOR ASM	3
FLOWCHARTS FOR ASM	5
SOURCE LISTING FOR ASM	7
VARIABLE DICTIONARY FOR ASM	26
METHOD OF SOLUTION FOR XREF	34
FLOWCHART FOR XREF	35
SOURCE LISTING FOR XREF	36
VARIABLE DICTIONARY FOR XREF	39
APPENDIX A: USAGE	40
SAMPLE RESULTS	42
APPENDIX B: EXTENSION TO ANOTHER ASSEMBLY LANGUAGE	44

ABSTRACT

An IBM-360 cross-assembler for a SDS-910 computer system was broken down into two major programs. The first program assembles SDS-910 Assembly Language code, and the second program uses information supplied by the first to produce a reference list of all identifiers used in the source SDS-910 Assembly Language program. Listings, documentation, and sample results are given for the two programs.

Distribution List

Chicago Patent Office	1
USERDA-TIC	27
Ames Laboratory Library	5
Kent Brobst	15
Leonard Moon	15
Barbara Helland	3
Total	66

Introduction

To effectively use the SDS-910 computers at the Ames Lab Research Reactor and the X-ray diffraction facility, a procedure was needed to translate programs into SDS-910 executable code. It was decided that a cross-assembler run on an IBM-360 would be the most appropriate translator for this situation. This would free the SDS-910 from translating the programs it must execute, allowing it to devote itself entirely to execution. The handling of programs that the SDS-910 system would do is therefore simplified, and the limited amount of core would be more effectively utilized. In addition, this would allow the assembler package to be written in a higher level language (PL/I), greatly aiding the writing, debugging, and understandability of the assembler.

Using assembly language for programs to be executed on the ALRR SDS-910 has certain advantages. The time and effort needed to write a compiler for a higher level language over the time needed to write an assembler would be saved. To this saving could be added the saving of defining a language, since the SDS-910 Assembly Language had already been defined. Much of the programming that would use the SDS-910 involves the handling of interrupts, and assembly language is especially well-suited to this task. Since the programs that would use this system would be fairly short and would run in a static environment, the lack of flexibility of the storage structures available in assembly language would present little difficulty. It

was felt that these advantages of using assembly language outweighed the difficulties of writing assembly language programs.

The available card readers for the SDS-910 dictated the formats of the object deck produced by the assembler. Column binary and Hollerith formats could read by the card readers in use at the time these programs were written. For this reason the user of the assembler can specify in which of these two formats he wants his object deck. Column binary format is generally preferred because it packs the code more densely on a card, making it less expensive. Column binary format has been used exclusively on the Ames Lab SDS-910 since the Hollerith format card reader broke down. As a result, requesting the reference listing generating program automatically specifies that the column binary format for the object deck will be used.

The reference listing generating routine, along with error messages generated by the assembler, have been included in this package to aid the user in debugging his programs. The reference listing can save the user from needing to scan an entire program searching for references to identifiers. It can also show the user identifiers that he is declaring but no longer referencing due to changes he's made in the program. Use of this routine is optional.

The cross-assembler (ASM) and the reference listing generator (XREF), if desired, are run as a single procedure. Information passed from the assembler to the reference listing generator is then completely independent of and invisible to the user, simplifying his work.

The two programs, documentation, and sample results are all given in this report.

Method of Solution for ASM

A two-pass technique is used to assemble the source program. The program treats each source instruction individually in sequence.

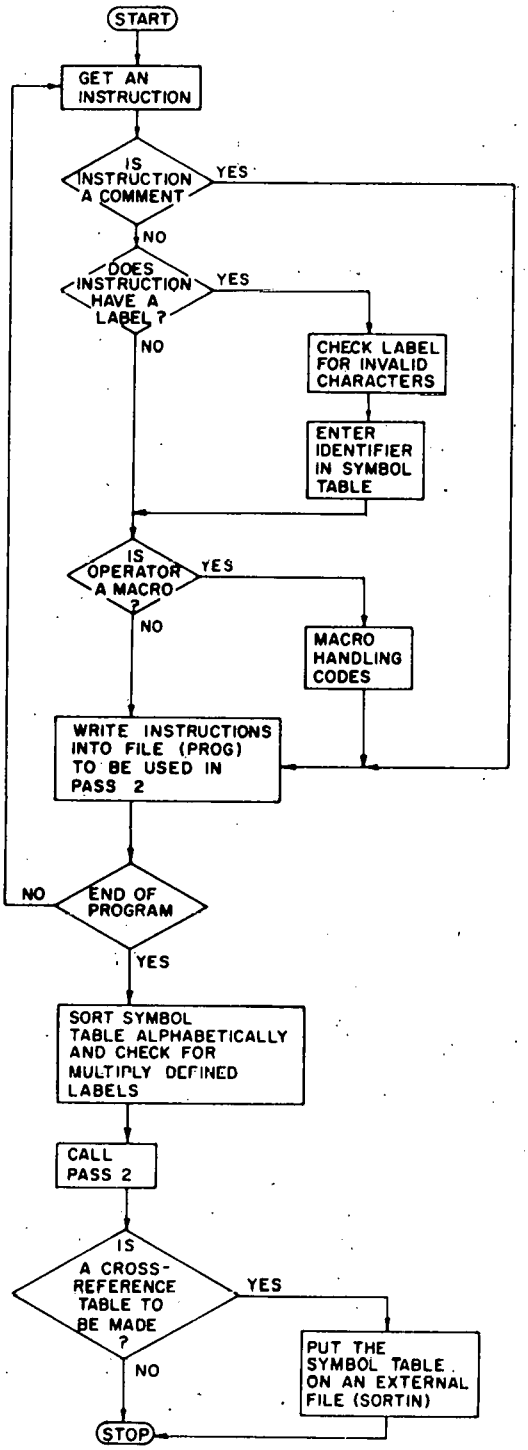
In the first pass each instruction is checked for a label and for a special macro operator. If a valid label is found, it and its address of occurrence are entered in the symbol table. If a macro operator (AORG, BSS, BCD, BOOL, EQU, POPD, OPD, or PAGE) is encountered, modifications commanded by the operator will be performed, such as changing the relative address counter (PCB) or creating a special temporary operator.

After each source instruction has been examined, it is put on an external file (PROG) for processing during the second pass of the assembler. When the source instructions have been exhausted, the symbol table is sorted alphabetically and checked for multiply defined labels.

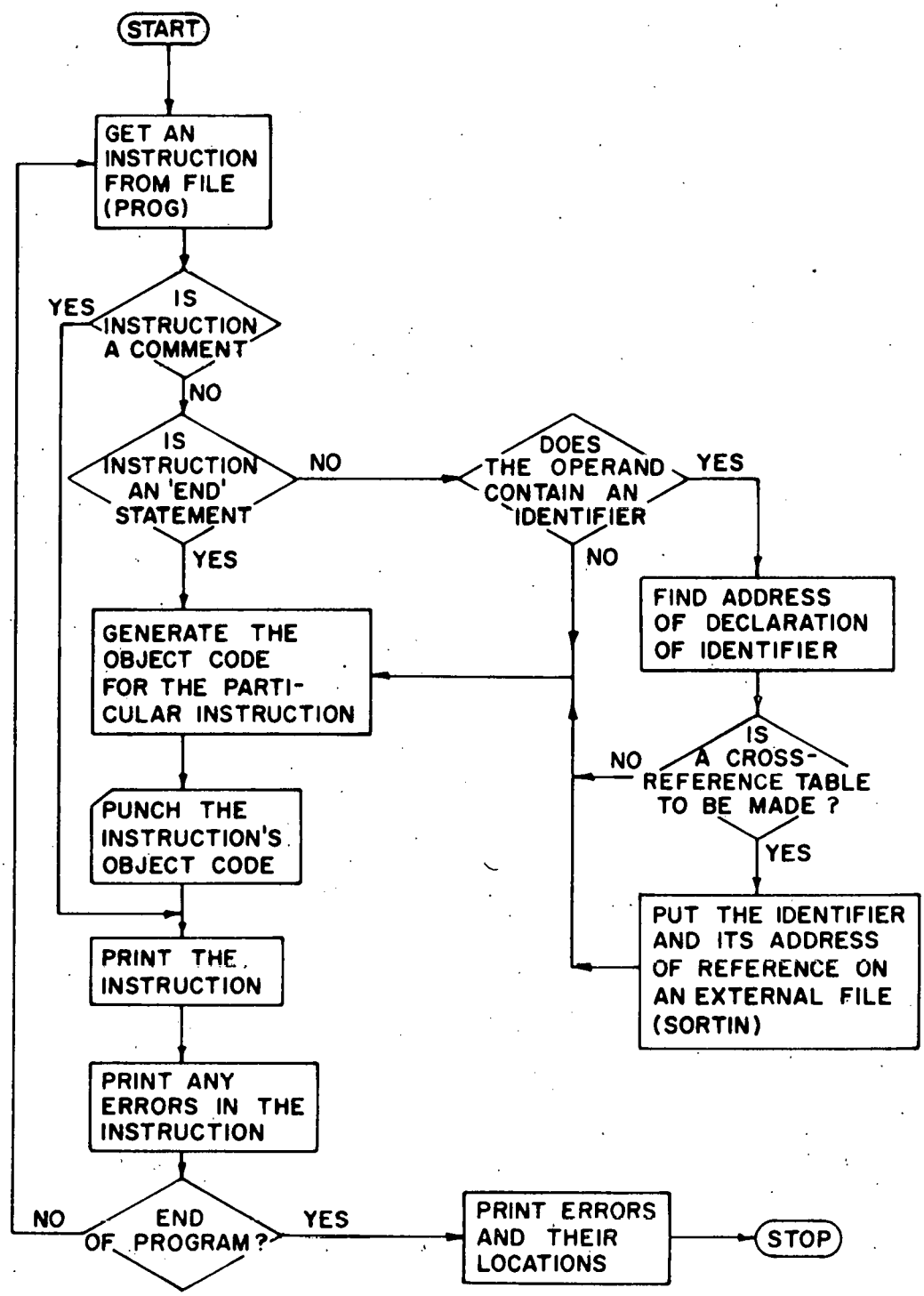
In the second pass each source instruction is retrieved from the external file (PROG). Its operator is checked and control passes to an appropriate section of code for the particular operator. The object code for the statement is generated and put into an external file (CB) for punching in the desired format (either Hollerith or column binary). If an identifier is found in the operand of the source instruction and a cross-reference listing is requested, the name of the identifier and the address of the particular reference are entered in the external

file SORTIN for processing by the cross-reference generating program. Of course, no object code is generated for comment statements.

After the source instruction has been processed, it is printed, along with any errors found in the statement. When the source program is completely translated and printed, the word 'DONE' is printed, followed by the number of errors found in the program and the statement numbers in which they occurred.



FLOWCHART FOR ASM THE SDS-910 CROSS-ASSEMBLER, FIRST PASS



FLOWCHART FOR SDS-910
CROSS-ASSEMBLER, SECOND PASS

Source Listing for ASM

```
ASM:   PROC OPTIONS (MAIN);                                          ASM00010
    DCL 1 XTEMP,                                                     ASM00020
          2 XNAM CHAR(6),                                            ASM00030
          2 XADDR FIXED BIN;                                         ASM00040
 DCL (I,J,L,M,IL(0:10),IU(0:10)) FIXED BIN STATIC;                   ASM00050
DCL 1 T STATIC, 2 TN CHAR(6), 2 TA FIXED BIN;                        ASM00060
DCL 1 TT STATIC, 2 TTN CHAR(6), 2 TTA FIXED BIN;                     ASM00070
    DCL ADR1 FIXED BIN(31,0);                                        ASM00080
        DCL 1 INST  STATIC,                                          ASM00090
             2 CARD CHAR (80),                                       ASM00100
               2 ERR FIXED BIN INIT(0);                              ASM00110
DCL LOC CHAR(6) DEFINED CARD, OP CHAR (4) DEF CARD POS(8),           ASM00120
        AND CHAR(10) DEFINED CARD POS(16);                           ASM00130
        DCL  LAB CHAR(6) DEF CARD, OPC CHAR(4) DEF CARD              ASM00140
            POS(8), OPER CHAR (10) DEF CARD POS (16);                ASM00150
      DCL TOPC  CHAR(6) DEF CARD POS(8);                             ASM00160
      DCL OPER1 CHAR(6) DEF CARD POS (16);                           ASM00170
        DCL OPERA(10) CHAR (1) DEF CARD POSITION (16);               ASM00180
        DCL PNUM BIT (32) STATIC;                                    ASM00190
        DCL PC1(5) BIT(3) DEF PNUM POSITION(18);                     ASM00200
        DCL CODE (9) LABEL INIT                                      ASM00210
          (AORG,RES,BCD,BOOL,EQU,POPD,POPD,PAG,NUN);                 ASM00220
        DCL BNUMB CHAR(1) DEF CARD POS (16);                         ASM00230
        DCL COM CHAR(1) DEF CARD;                                    ASM00240
        DCL PROG FILE RECORD SEQUENTIAL;                             ASM00250
        DCL DUMMY CHAR(8) STATIC;                                    ASM00260
    DCL PNAME CHAR(75);                                              ASM00270
DCL 1 SYM(400) STATIC, 2 NAM CHAR(6), 2 ADDR FIXED BIN;              ASM00280
DCL SP FIXED BIN STATIC INIT(0);                                     ASM00290
        DCL TMP FIXED BIN (31,0) STATIC;                             ASM00300
      DCL PCB FIXED BIN (31,0) INIT(-1) STATIC;                      ASM00310
        DCL  1 TOPER STATIC,                                         ASM00320
          2 TOP (30) CHAR(6),                                        ASM00330
            2 TMAX FIXED BIN INIT(0),                                ASM00340
            2 TCODE (30) BIT(24);                                    ASM00350
```

```
DCL ONUM BIT (32) STATIC, ONUM1 BIT(24) DEF ONUM POSITION(9);          ASM00360
        DCL OCON ENTRY (CHAR(10)) RETURNS (FIXED BIN (31,0));          ASM00370
      DCL SEARCH ENTRY (CHAR(6),FIXED BIN(31,0))                       ASM00380
     RETURNS(FIXED BIN (31,0));                                        ASM00390
        DCL OPC1(8) CHAR(4) INIT                                       ASM00400
          ('AORG','BSS ','BCD ','BOOL','EQU ','POPD','OPD ','PAGE');   ASM00410
        DCL TEMP BIT(24) STATIC;                                       ASM00420
        DCL FIVFOR BIT(24) STATIC INIT('101100101100101100101100'B);   ASM00430
        DCL ENDFLG BIT(1) STATIC INIT('0'B);                           ASM00440
        DCL EFLG BIT(1) STATIC INIT ('0'B);                            ASM00450
        DCL KK FIXED BIN STATIC;                                       ASM00460
        DCL PFLG BIT(1) STATIC INIT('0'B), PARM CHAR(5);               ASM00470
        DCL ADCON CHAR(10) STATIC;                                     ASM00480
      DCL ERRCTR FIXED BIN INIT(0) STATIC;                             ASM00490
        DCL NFLG BIT(1) INIT ('0'B) STATIC;                            ASM00500
        DCL FLG BIT(1) INIT ('1'B);                                    ASM00510
        DCL OB_PTR FIXED BIN INIT (16) STATIC;                         ASM00520
  DCL CB FILE RECORD OUTPUT;                                           ASM00530
  DCL PUNCH ENTRY,SEQ ENTRY;                                           ASM00540
  DCL  I            FIXED BIN STATIC INIT(0);                          ASM00550
        DCL BLNK BIT(8) INIT ('00000000'B);                            ASM00560
        DCL BLNK1 BIT(6) DEF BLNK POS (3);                             ASM00570
        DCL OB_MAX FIXED BIN INIT(144) STATIC;                         ASM00580
        DCL SEQ_NO FIXED BIN INIT (0) STATIC;                          ASM00590
      DCL OB BIT (24);                                                 ASM00600
        DCL OB6(4) BIT (6) DEF OB;                                     ASM00610
        DCL OB3(8) BIT(3) DEF OB;                                      ASM00620
  DCL B(160) BIT(8) STATIC INIT((160)(8)'0'B);                         ASM00630
        DCL CHR_CBL(0:63)BIT(8)STATIC INIT(                            ASM00640
          '00000000'B,'00100100'B,'00100010'B,'00100001'B, /* ABC*/    ASM00650
          '00100000'B,'00100000'B,'00100000'B,'00100000'B, /*DEFG*/    ASM00660
          '00100000'B,'00100000'B,'00010100'B,'00010010'B, /*HIJK*/    ASM00670
          '00010001'B,'00010000'B,'00010000'B,'00010000'B, /*LMNO*/    ASM00680
          '00010000'B,'00010000'B,'00010000'B,'00001010'B, /*PQRS*/    ASM00690
          '00001001'B,'00001000'B,'00001000'B,'00001000'B, /*TUVW*/    ASM00700
```

```
        '00001000'B,'00001000'B,'00001000'B,'00100010'B, /*XYZ */     ASM00710
        '00010000'B,'00010010'B,'00001010'B,'00001000'B, /*¬ -_*/     ASM00720
        '00000000'B,'00100000'B,'00000000'B,'00000001'B, /*@|"#*/     ASM00730
        '00010001'B,'00001000'B,'00100000'B,'00000000'B, /*$%&'*/     ASM00740
        '00100000'B,'00010000'B,'00010000'B,'00100000'B, /*()*+*/     ASM00750
        '00001001'B,'00010000'B,'00100001'B,'00001100'B, /*,-./*/     ASM00760
        '00001000'B,'00000100'B,'00000010'B,'00000001'B, /*0123*/     ASM00770
        '00000000'B,'00000000'B,'00000000'B,'00000000'B, /*4567*/     ASM00780
        '00000000'B,'00000000'B,'00000010'B,'00010000'B, /*89:;*/     ASM00790
        '00100000'B,'00000000'B,'00001000'B,'00001000'B);/*<=>?*/     ASM00800
      DCL CHR_CBR(0:63)BIT(8)STATIC INIT(                             ASM00810
        '00000000'B,'00000000'B,'00000000'B,'00000000'B, /* ABC*/     ASM00820
        '00100000'B,'00010000'B,'00001000'B,'00000100'B, /*DEFG*/     ASM00830
        '00000010'B,'00000001'B,'00000000'B,'00000000'B, /*HIJK*/     ASM00840
        '00000000'B,'00100000'B,'00010000'B,'00001000'B, /*LMNO*/     ASM00850
        '00000100'B,'00000010'B,'00000001'B,'00000000'B, /*PQRS*/     ASM00860
        '00000000'B,'00100000'B,'00010000'B,'00001000'B, /*TUVW*/     ASM00870
        '00000100'B,'00000010'B,'00000001'B,'00000010'B, /*XYZ */     ASM00880
        '00000110'B,'00000010'B,'00000010'B,'00010010'B, /*¬ -_*/     ASM00890
        '00100010'B,'00000110'B,'00000110'B,'00000010'B, /*@|"#*/     ASM00900
        '00000010'B,'00100010'B,'00000000'B,'00010010'B, /*$%&'*/     ASM00910
        '00010010'B,'00010010'B,'00100010'B,'00001010'B, /*()*+*/     ASM00920
        '00000010'B,'00000000'B,'00000010'B,'00000000'B, /*,-./*/     ASM00930
        '00000000'B,'00000000'B,'00000000'B,'00000000'B, /*0123*/     ASM00940
        '00100000'B,'00010000'B,'00001000'B,'00000100'B, /*4567*/     ASM00950
        '00000010'B,'00000001'B,'00000010'B,'00001010'B, /*89:;*/     ASM00960
        '00100010'B,'00001010'B,'00001010'B,'00000110'B);/*<=>?*/     ASM00970
    DCL (II,J,K,MM) FIXED BIN;                                        ASM00980
    DCL (SW,SW1) BIT(1);                                              ASM00990
/*_____________________________________________________________________*/ASM01000
/*_____________________________________________________________________*/ASM01010
      ON ENDFILE (SYSIN) GO TO SORT;                                  ASM01020
    OPEN FILE (PROG) OUTPUT;                                          ASM01030
    GET EDIT (PARM,PNAME) (A(5),A(75));                               ASM01040
    IF PARM ¬=' ' THEN PFLG='1'B;                                     ASM01050
```

```
/*_____________________________________________________________________*/ASM01060
/*_____________________________________________________________________*/ASM01070
/*     GET AN INSTRUCTION.     */                                        ASM01080
                                                                         ASM01090
LOOP: GET EDIT (CARD) (A(80));                                           ASM01100
      SW='0'B;                                                           ASM01110
                                                                         ASM01120
/*     IF THE STATEMENT IS A COMMENT, PUT IT IN THE FILE (PROG) AND      ASM01130
      GET THE NEXT INSTRUCTION.     */                                   ASM01140
                                                                         ASM01150
      IF COM='*' THEN GO TO NUN;                                         ASM01160
        PCB=PCB+1;                                                       ASM01170
/*_____________________________________________________________________*/ASM01180
/*_____________________________________________________________________*/ASM01190
/*     IF THE INSTRUCTION HAS A LABEL, CHECK IT FOR INVALID CHARACTERS   ASM01200
      AND ENTER IT IN THE SYMBOL TABLE.     */                           ASM01210
                                                                         ASM01220
      IF LOC¬='       ' THEN DO;                                         ASM01230
        SW='1'B;                                                         ASM01240
        IF SP=400 THEN DO; PUT SKIP LIST ('***TOO MANY SYMBOLS');        ASM01250
          GO TO SORT; END;                                               ASM01260
        IF COM > 'Z' THEN DO; ERR=10; GO TO LOOP; END;  /*ERR 10 */      ASM01270
        M=INDEX(LOC,' '); IF M=0 THEN M=7;                               ASM01280
          DO I=1 TO M-1;                                                 ASM01290
            IF SUBSTR(LOC,I,1) < 'A' THEN DO; ERR=10; GO TO LOOP; END;   ASM01300
          END;                                                           ASM01310
        SP=SP+1; NAM(SP)=LOC;  ADDR(SP)=PCB; END;                        ASM01320
/*_____________________________________________________________________*/ASM01330
/*_____________________________________________________________________*/ASM01340
/*     IF THE OPERATOR IS A MACRO, USE THE APPROPRIATE MACRO-HANDLING    ASM01350
      INSTRUCTIONS.     */                                               ASM01360
                                                                         ASM01370
      DO I=1 TO 8 WHILE (OP¬= OPC1(I)); END; GO TO CODE(I);              ASM01380
AORG: TMP=OCON(AND); PCB=TMP-1;  GO TO NUN;                              ASM01390
EQU:  PCB=PCB-1; DO I=1 TO SP-1 WHILE (NAM(I)¬=AND); END;                ASM01400
```

```
      IF I=SP THEN ERR=3;      /* ERR 3 */                          ASM01410
      SW='0'B;                                                      ASM01420
      ADDR(SP)=ADDR(I);  GO TO NUN;                                 ASM01430
RES:  PCB=PCB+AND-1; SW='0'B; GO TO NUN;                            ASM01440
BCD:  IF BNUMB > '4' THEN PCB=PCB+1; SW='0'B; GO TO NUN;            ASM01450
PAG:  PCB=PCB-1;  SW='0'B; GO TO NUN;                               ASM01460
BOOL: PCB=PCB-1; ADDR(SP)=OCON(AND);                                ASM01470
      IF ADDR(SP)=-1 THEN ADDR(SP)=0;  SW='0'B; GO TO NUN;          ASM01480
POPD: TMAX=TMAX+1;                                                  ASM01490
      PCB=PCB-1;                                                    ASM01500
      IF TMAX>30 THEN DO; ERR=8; GO TO NUN; END;                    ASM01510
      TOP(TMAX)=LOC;                                                ASM01520
      ONUM=UNSPEC(OCON(AND));                                       ASM01530
      TCODE(TMAX)=ONUM1;                                            ASM01540
      SW='0'B;                                                      ASM01550
/*____________________________________________________________*/ASM01560
/*____________________________________________________________*/ASM01570
/*    PUT THE INSTRUCTION IN THE FILE (PROG) FOR USE DURING THE   ASM01580
      SECOND PASS.    */                                            ASM01590
                                                                    ASM01600
NUN:  WRITE FILE(PROG) FROM  (INST);                                ASM01610
      IF SW THEN ADDR(SP)=PCB;                                      ASM01620
      ERR=0;  GO TO LOOP;                                           ASM01630
/*____________________________________________________________*/ASM01640
/*____________________________________________________________*/ASM01650
/*    SORT THE SYMBOL TABLE ALPHABETICALLY AND CHECK FOR MULTIPLY  ASM01660
      DEFINED LABELS.    */                                         ASM01670
                                                                    ASM01680
SORT: SW1='1'B;  J=SP;  II=1;  MM=0;  GO TO L4;                     ASM01690
L1:   IIJ=(II+J)/2;  T=SYM(IIJ);  K=II; L=J;                        ASM01700
L1_1:  IF NAM(II) > TN THEN DO;                                     ASM01710
        SYM(IIJ)=SYM(II);  SYM(II)=T;  T=SYM(IIJ);  END;            ASM01720
      IF  SW1 THEN DO;  IF NAM(J)<TN THEN DO;                       ASM01730
          SYM(IIJ)=SYM(J);  SYM(J)=T;                               ASM01740
          T=SYM(IIJ);  SW1='0'B;  GO TO L1_1;  END;                 ASM01750
```

```
        END;  SW1='1'B;                                                  ASM01760
L2:   L=L-1;  IF NAM(L)> TN THEN GO TO L2;  TT=SYM(L);                   ASM01770
L3:   K=K+1;  IF NAM(K) < TN THEN GO TO L3;                              ASM01780
      IF K<L THEN DO; SYM(L)=SYM(K);  SYM(K)=TT;  GO TO L2; END;         ASM01790
      IL(MM)=K;  IU(MM)=J;  J=L;  MM=MM+1;                               ASM01800
L4:   IF J-II > 10 THEN GO TO L1;                                        ASM01810
      IF II=1 THEN II=II+1;                                              ASM01820
      DO II=II TO J;  T=SYM(II); K=II-1;                                 ASM01830
        IF NAM(K) > TN THEN DO;                                          ASM01840
L5:   SYM(K+1)=SYM(K);  K=K-1;  IF K=0 THEN GO TO L5_1;                  ASM01850
      IF NAM(K) > TN THEN GO TO L5;                                      ASM01860
      IF NAM(K) = TN THEN                                                ASM01870
        PUT SKIP LIST('***MULTIPLY DEFINED LABEL',NAM(K));               ASM01880
L5_1: SYM(K+1)=T;  GO TO L6; END;                                        ASM01890
      IF NAM(K) = TN THEN PUT SKIP EDIT                                  ASM01900
        ('***MULTIPLY DEFINED LABEL',NAM(K)) (A(25),X(5),A(6));          ASM01910
L6:   END;                                                               ASM01920
      MM=MM-1;  IF MM>=0 THEN DO; II=IL(MM);  J=IU(MM); GO TO L4; END;   ASM01930
/*_____________________________________________________________________*/ASM01940
/*_____________________________________________________________________*/ASM01950
/*    START THE SECOND PASS.    */                                       ASM01960
                                                                         ASM01970
      CLOSE FILE (PROG);                                                 ASM01980
      CALL PASS2;                                                        ASM01990
/*_____________________________________________________________________*/ASM02000
/*_____________________________________________________________________*/ASM02010
OCON: PROC (CHA) RETURNS(FIXED BIN (31,0));                              ASM02020
      DCL CHA CHAR(*);                                                   ASM02030
      DCL IADR FIXED BIN (31,0);                                         ASM02040
      DCL (N1,LL) FIXED BIN;                                             ASM02050
      DCL CARDI CHAR (10);                                               ASM02060
      DCL OPR(10) CHAR (1) DEF CARDI;                                    ASM02070
      CARDI=CHA;                                                         ASM02080
      IADR=0;                                                            ASM02090
      M=INDEX(CARDI,' ');                                                ASM02100
```

```
      DO LL=1 TO M-1;                                                   ASM02110
      N1=INDEX('01234567',OPR(LL))-1;                                   ASM02120
      IF N1<0 THEN DO; ERR=2; RETURN(-1); END;    /* ERR02 */           ASM02130
      N1=OPR(LL);                                                       ASM02140
      IADR=IADR*8+N1;                                                   ASM02150
      END;                                                              ASM02160
      RETURN(IADR);                                                     ASM02170
      END OCON;                                                         ASM02180
/*_____________________________________________________________________*/ASM02190
/*_____________________________________________________________________*/ASM02200
SEARCH: PROC(LABL,PCA) RETURNS(FIXED BIN(31,0));                        ASM02210
      DCL LABL CHAR(*);                                                 ASM02220
      DCL N FIXED BIN;                                                  ASM02230
      DCL PCA FIXED BIN (31,0);                                         ASM02240
      DCL IA FIXED BIN (31,0) STATIC;                                   ASM02250
      DCL  (ULIMIT,LLIMIT) FIXED BIN STATIC;                            ASM02260
  DCL 1 A,                                                              ASM02270
     2 NN CHAR(6),                                                      ASM02280
     2 ADDR1 FIXED BIN (31,0),                                          ASM02290
     2 FILLER (4) FIXED BIN (15,0);                                     ASM02300
      IA = 0;                                                           ASM02310
      ULIMIT=SP;  LLIMIT=1;                                             ASM02320
     IF PARM='XREF ' THEN DO;                                           ASM02330
 /* THIS CODE CREATES THE ORDERED PAIRS (LOCATION IN SYMBOL TABLE,      ASM02340
     ADDRESS OF REFERENCE */                                            ASM02350
     NN=LABL;                                                           ASM02360
     A.ADDR1=PCA;                                                       ASM02370
     WRITE FILE (SORTIN) FROM (A);                                      ASM02380
  END;                                                                  ASM02390
SLOOP: IF (ULIMIT=LLIMIT) THEN DO; N=ULIMIT;                            ASM02400
        IF LABL=NAM(ULIMIT) THEN GO TO DONE;  ELSE RETURN (IA);         ASM02410
      END;                                                              ASM02420
      N=(ULIMIT+LLIMIT)/2;                                              ASM02430
      IF LABL=NAM(N) THEN GO TO DONE;                                   ASM02440
      IF LABL<NAM(N) THEN ULIMIT=N-1;                                   ASM02450
```

```
                    ELSE LLIMIT=N+1;                                    ASM02460
      IF ULIMIT < LLIMIT THEN RETURN (IA);                              ASM02470
      GO TO SLOOP;                                                      ASM02480
DONE: IA=SYM.ADDR(N);                                                   ASM02490
      RETURN(IA);                                                       ASM02500
      END SEARCH;                                                       ASM02510
/*_____________________________________________________________________*/ASM02520
/*_____________________________________________________________________*/ASM02530
SEQ:  PROC;        /*  NAME FIELD IS ALREADY SET UP */                 ASM02540
                   /*  INSERT SEQUENCE NUMBER AND PUNCH CARD */        ASM02550
      DCL IB FIXED BIN STATIC;                                          ASM02560
        SEQ_NO=SEQ_NO + 1;                                              ASM02570
        KK=SEQ_NO/100+48;   B(155)=CHR_CBL(KK); B(156)=CHR_CBR(KK);     ASM02580
        KK=MOD(SEQ_NO,100)/10+48;                                       ASM02590
                            B(157)=CHR_CBL(KK); B(158)=CHR_CBR(KK);     ASM02600
        KK=MOD(SEQ_NO,10)+48;                                           ASM02610
                            B(159)=CHR_CBL(KK); B(160)=CHR_CBR(KK);     ASM02620
     /* PUNCH CARD  */                                                  ASM02630
        WRITE FILE(CB) FROM(B);                                         ASM02640
        DO IB=1 TO 160; B(IB)='00000000'B;  END;                        ASM02650
        RETURN;                                                         ASM02660
        END;                                                            ASM02670
/*_____________________________________________________________________*/ASM02680
/*_____________________________________________________________________*/ASM02690
PUNCH: PROC;                                                            ASM02700
      DCL ID FIXED BIN STATIC;                                          ASM02710
      IF FLG THEN DO; FLG='0'B; GO TO PUTADD; END;                      ASM02720
      /* SEQUENCE CARD AND PUNCH  PUT NEW ADDR ON NEXT CARD */          ASM02730
      IF OB_PTR >= OB_MAX THEN DO;                                      ASM02740
      CALL SEQ;  OB_PTR=16;                                             ASM02750
PUTADD: TEMP=OB; OB=(24)'0'B;                                           ASM02760
      IF ¬ENDFLG THEN                                                   ASM02770
        SUBSTR(OB,3,22)=SUBSTR(PNUM,11,22);                             ASM02780
      ELSE OB=FIVFOR;                                                   ASM02790
      IF PFLG THEN DO ID=1 TO 4;                                        ASM02800
```

```
        BLNK1=OB6(ID);  B(12+ID)=BLNK;                                  ASM02810
        END;                                                            ASM02820
      ELSE DO ID=1 TO 8;                                                ASM02830
        KK=OB3(ID)+48;                                                  ASM02840
        B(ID*2-1)=CHR_CBL(KK);                                          ASM02850
        B(ID*2)=CHR_CBR(KK); END;                                       ASM02860
        OB=TEMP;                                                        ASM02870
      END;                                                              ASM02880
      IF EFLG THEN GO TO ENDP;                                          ASM02890
      IF PFLG THEN DO ID=1 TO 4;                                        ASM02900
        OB_PTR=OB_PTR+1;                                                ASM02910
        BLNK1=OB6(ID);  B(OB_PTR)=BLNK;                                 ASM02920
      END;                                                              ASM02930
      ELSE DO ID=1 TO 8;                                                ASM02940
        KK=OB3(ID)+48;                                                  ASM02950
        OB_PTR=OB_PTR+1;  B(OB_PTR)=CHR_CBL(KK);                        ASM02960
        OB_PTR=OB_PTR+1;  B(OB_PTR)=CHR_CBR(KK);   END;                 ASM02970
ENDP: RETURN;                                                           ASM02980
      END;                                                              ASM02990
/*_____________________________________________________________________*/ASM03000
/*_____________________________________________________________________*/ASM03010
PASS2: PROC;                                                            ASM03020
      /* SECOND PASS OF 910 ASSEMBLER   */                              ASM03030
      DCL INSTR BIT (24) STATIC;                                        ASM03040
      DCL INSTRO(8) BIT (3) DEF INSTR;                                  ASM03050
      DCL (INDIRECT BIT (24) INIT('000000000100000000000000'B),         ASM03060
           ZERO15 BIT(15) INIT('000000000000000'B),                     ASM03070
           ZERO24 BIT(24) INIT('000000000000000000000000'B),            ASM03080
           BLANK CHAR(10) INIT ('          ')) STATIC;                  ASM03090
      DCL (ADR FIXED BIN(31,0), BNUM BIT(32)) STATIC;                   ASM03100
      DCL ADR1 FIXED BIN(31,0);                                         ASM03110
      DCL BNUM1 BIT (24) DEF BNUM POSITION(9);                          ASM03120
      DCL FWD BIT (24) STATIC INIT ('000000000000000110000000'B);       ASM03130
      DCL OVRF BIT(24) STATIC INIT('100000000000000000000000'B);        ASM03140
      DCL X BIT(24) INIT('010000000000000000000000'B)STATIC;            ASM03150
```

```
DCL PC FIXED BIN (31,0) INIT(-1) STATIC;                                ASM03160
DCL (CONST CHAR(6), OPCODE CHAR(3)) STATIC;                             ASM03170
DCL FACT FIXED BIN STATIC;                                              ASM03180
DCL ADRONE BIT (24) STATIC INIT ('000000000011111111111111'B);          ASM03190
DCL DIGIT CHAR(10) STATIC INIT ('0123456789');                          ASM03200
DCL OPA(0:71) CHAR (3) STATIC INIT                                      ASM03210
     ('LDA','STA','LDB','STB','LDX','STX','EAX','ADD','MIN',            ASM03220
      'MDE','SUB','ETR','MRG','EOR','BRU','BRX','BRM','BRR',            ASM03230
      'SKG','SKM','SKA','SKN','PIN','POT','MIW','MIY','WIM',            ASM03240
      'YIM','MUS','DIS','EXU','RSH','RCY','LSH','LCY','NOD',            ASM03250
      'EOM','SKS','NOP','HLT','EIR','DIR','IET','IDT','AIR',            ASM03260
      'OVT','ROV','XAB','BAC','ABC','CLR','ALC','DSC','TOP',            ASM03270
      'ASC','BRT','BET','SRC','CRT','FCT','CFT','OCT','BOO',            ASM03280
      'DEC','EQU','BCD','PAG','BSS','AOR','PZE','OPD','POP'             ASM03290
    );                                                                  ASM03300
DCL IOP(0:6) CHAR (3) STATIC INIT                                       ASM03310
  ('RCB','RCD','RKB','TYP','RPT','PPT','PTL');                          ASM03320
DCL IOP2 (0:6) BIT (24) STATIC INIT                                     ASM03330
      ('000000010000011000000110'B,'000000010000010000000110'B,         ASM03340
        '000000010000010000000001'B,'000000010000010000100001'B,        ASM03350
        '000000010000010000000100'B,'000000010000010000100100'B,        ASM03360
        '000000010000000000100100'B                                     ASM03370
      );                                                                ASM03380
DCL OPB(0:30) BIT (9) STATIC INIT                                       ASM03390
      ('000111110'B,'000011101'B,'000111101'B,'000011110'B,             ASM03400
       '000111001'B,'000011111'B,'000111111'B,'000101101'B,             ASM03410
       '000110001'B,'000110000'B,'000101100'B,'000001100'B,             ASM03420
       '000001110'B,'000001111'B,'000000001'B,'000100001'B,             ASM03430
       '000100011'B,'000101001'B,'000111011'B,'000111000'B,             ASM03440
       '000111010'B,'000101011'B,'000011011'B,'000001011'B,             ASM03450
       '000001010'B,'000001000'B,'000011010'B,'000011000'B,             ASM03460
       '000110100'B,'000110101'B,'000010011'B);                         ASM03470
DCL  OPB2(0:29) BIT (24) STATIC INIT                                    ASM03480
    ('000110110000000000000000'B,'000110110010000000000000'B,           ASM03490
     '000110111000000000000000'B,'000110111010000000000000'B,           ASM03500
```

```
          '0001101110010000000000'B,'000000010000000000000000'B,     ASM03510
          '0001000000000000000000'B,'000010000000000000000000'B,     ASM03520
          '0000000000000000000000'B,'000000010010000000000010'B,     ASM03530
          '0000000100100000000100'B,'000100000010000000000100'B,     ASM03540
          '0001000000100000000010'B,'000000010010000000010000'B,     ASM03550
          '0001000000100000000001'B,'000000010010000000000001'B,     ASM03560
          '0001001100000000000000'B,'000100110001000000000000'B,     ASM03570
          '0001001100100000000000'B,'000100110011000000000000'B,     ASM03580
          '0000000101010000000000'B,'000000010000000000000000'B,     ASM03590
          '0000000100011000000000'B,'000000010001010000000000'B,     ASM03600
          '0001000000100010000000'B,'000100000010000000010000'B,     ASM03610
          '0000000100010100000110'B,'000100000001010000000110'B,     ASM03620
          '0001000000101000000110'B,'000100000001001000000110'B      ASM03630
          );                                                         ASM03640
     DCL (IC,IM,IP,JJ,N2) FIXED BIN;                                 ASM03650
     DCL OPL(11) LABEL INIT                                          ASM03660
       (BOC,BOUTA,DECM,EQUA,BCDT,PAGES,RESV,AORG,OUTPUT,BOUT,BOUT);  ASM03670
     DCL VAL1 BIT(6) DEF BNUM POSITION (27);                         ASM03680
     DCL CHARSET CHAR(64) STATIC INIT(                               ASM03690
'0123456789 #1:>V+ABCDEFGHI?.) < -JKLMNOPQR $* ;  /STUVWXYZ ,(    ');  ASM03700
     DCL ERRTAB(20) CHAR(20) STATIC INIT                             ASM03710
          ('***ILLEGAL ORG ADD  ','***ILLEGAL OCT NUM  ',            ASM03720
           '***MISSING LABEL    ','***ILLEGAL OPERAND  ',            ASM03730
           '***ILLEGAL INDIRECT ','***ILLEGAL INDEX    ',            ASM03740
           '***NUMBER TOO BIG   ','***TOO MANY TEMP OPS',            ASM03750
           '***ILLEGAL OP CODE  ','***ILLEGAL LABEL    ',            ASM03760
           (10)'***                  ');                             ASM03770
  DCL ERRLINE (50) FIXED BIN (31,0);                                 ASM03780
     DCL BLANK50 CHAR(50) STATIC INIT                                ASM03790
      ('                                                  ');        ASM03800
/*_____________________________________________________________*/ASM03810
/*_____________________________________________________________*/ASM03820
     ON CONVERSION GO TO OUTPUT;                                     ASM03830
     ON ENDFILE (PROG) GO TO FINI;                                   ASM03840
     OPEN FILE (PROG) INPUT;                                         ASM03850
```

```
/*_____________________________________________________________________*/ASM03860
/*_____________________________________________________________________*/ASM03870
/*     GET AN INSTRUCTION.     */                                        ASM03880
                                                                         ASM03890
LOOP: READ FILE (PROG) INTO (INST);                                      ASM03900
      IC=0;                                                              ASM03910
      NFLG='0'B;                                                         ASM03920
      INSTR=ZERO24; PC=PC+1;                                             ASM03930
/*_____________________________________________________________________*/ASM03940
/*_____________________________________________________________________*/ASM03950
/*     IF THE STATEMENT IS A COMMENT, PRINT IT.     */                   ASM03960
                                                                         ASM03970
      IF INDEX(LAB,'*')=1 THEN GO TO COUTPT;                             ASM03980
      OPCODE=SUBSTR(OPC,1,3);                                            ASM03990
/*_____________________________________________________________________*/ASM04000
/*_____________________________________________________________________*/ASM04010
/*     IF IT'S AN 'END' STATEMENT, GENERATE THE APPROPRIATE OBJECT      ASM04020
      CODE, PUNCH THE CODE, AND PRINT THE STATEMENT.     */              ASM04030
                                                                         ASM04040
      IF OPCODE = 'END' THEN DO;                                         ASM04050
        OB_PTR=144;                                                      ASM04060
        ENDFLG='1'B;                                                     ASM04070
        IF OPER = BLANK THEN EFLG='1'B;                                  ASM04080
      ELSE DO;                                                           ASM04090
        ADR=SEARCH(OPER1,PC);  IF ADR = 0 THEN DO; ERR=3; GO TO EOUT;    ASM04100
                 END;                                                    ASM04110
        BNUM=UNSPEC(ADR);  INSTR=INSTR|BNUM1;                            ASM04120
        OB=INSTR;   END;                                                 ASM04130
      CALL PUNCH;                                                        ASM04140
EOUT: CALL SEQ;                                                          ASM04150
      PUT SKIP EDIT (CARD) (X(25),A(80));                                ASM04160
      GO TO ERRCHK;                                                      ASM04170
          END;                                                           ASM04180
/*_____________________________________________________________________*/ASM04190
/*_____________________________________________________________________*/ASM04200
```

```
/*     GENERATE THE OBJECT CODE FOR THE PARTICULAR INSTRUCTION.     */   ASM04210
                                                                        ASM04220
       DO I=0 TO 71 WHILE (OPCODE ¬= OPA(I)); END;                      ASM04230
       IF I < 72 THEN GO TO BEGIN;                                      ASM04240
       DO I=0 TO 6 WHILE (OPCODE ¬= IOP(I)); END;                       ASM04250
       IF I < 7 THEN GO  TO ICHDLR;                                     ASM04260
         DO I=1 TO TMAX WHILE (TOPC ¬= TOP(I)); END;                    ASM04270
         IF I>TMAX THEN DO; ERR=9; ADR=SEARCH(OPER1,PC); GO TO OUTPUT;  ASM04280
         END;                    INSTR=TCODE(I);                        ASM04290
         IF INSTR & '001C0CC00000000000000000'B                         ASM04300
         THEN GO TO OPATG1;  ELSE GC TO OPTG;                           ASM04310
/*_____________________________________________________________________*/ASM04320
/*_____________________________________________________________________*/ASM04330
       /* POPS NOT CHECKED FOF INDEX OR INDIRECT BIT    */               ASM04340
                                                                        ASM04350
BEGIN: IF I>37 & I<=60 THEN  DO; INSTR=OPB2(I-31);                      ASM04360
         IF OPER ¬= BLANK THEN GO TO OPATG1;                            ASM04370
         GO TO OUTPUT;   END;                                           ASM04380
       IF I > 60 THEN GO TO OPL(I-60);                                  ASM04390
       N=INDEX(OPER,' ');                                               ASM04400
       IF N¬=0 THEN IF SUBSTR(OPER,N)¬=' ' THEN DO;                     ASM04410
         ERR=4; GO TO OUTPUT;  END;  /* ERRO4  */                       ASM04420
       IF I<=30 THEN GO TO OPATAG;                                      ASM04430
       INSTR=OPB2(I-31);                                                ASM04440
       IF SUBSTR(OPC,4)='*' THEN DO; ERR=5; GO TO OUTPUT; END; /*E05*/  ASM04450
       IC=INDEX(OPER,',');                                              ASM04460
       IF OPERA(1)<'0' THEN DO; ERR=4; GO TO OUTPUT; END; /*E04*/       ASM04470
       IF I=36 | I=37 THEN                                              ASM04480
       IF IC ¬= 0 THEN GC TO ERRT;                                      ASM04490
       ELSE DO;  ADR =OCCN(OPER); GO TO BINST; END;                     ASM04500
       IF IC¬=0 THEN IF CPERA(IC+1)='2'                                 ASM04510
         THEN DO; INSTR=INSTR|X; DUMMY=SUBSTR(OPER,1,IC-1);             ASM04520
         ADR=DUMMY; END;                                                ASM04530
       ELSE GO TO ERRT;                                                 ASM04540
       ELSE ADR=OPER;                                                   ASM04550
```

```
      IF ADR>63 THEN DO ERR=7; GO TO OUTPUT; END;    /*E07*/          ASM04560
      GO TO BINST;                                                   ASM04570
ERRT: ERR=6;  GO TO OUTPUT;                                          ASM04580
/*_____________________________________________________________*/ASM04590
/*_____________________________________________________________*/ASM04600
OPATAG: INSTR=OPB(I)||ZERO15;                                        ASM04610
OPTG: IF SUBSTR(OPC,4)='*' THEN INSTR=INSTR|INDIRECT;                ASM04620
      IC,M=INDEX(OPER,',');                                          ASM04630
      IF M¬=0 THEN                                                   ASM04640
        IF OPERA(M+1)='2' THEN INSTR=INSTR|X;                        ASM04650
        ELSE IF OPERA(M+1)='4' THEN INSTR=INSTR|OVRF;                ASM04660
        ELSE DO; ERR=6; GO TO OUTPUT; END;                           ASM04670
                                                                     ASM04680
OPATG1: IF OPERA(1)='*' THEN DO;                                     ASM04690
        M=INDEX(OPER,' ');                                           ASM04700
        IF M=2 THEN DO; ADR=PC; GO TO BINST; END;                    ASM04710
        IF IC=0 THEN ADCON=SUBSTR(OPER,3,M-3);                       ASM04720
                ELSE ADCON=SUBSTR(OPER,3,IC-3);                      ASM04730
        ADR=OCON(ADCON);                                             ASM04740
        IF OPERA(2)='-'  THEN ADR=PC-ADR;                            ASM04750
                ELSE IF OPERA(2)='+' THEN ADR=PC+ADR;                ASM04760
                  ELSE DO; ERR=4; GO TO OUTPUT; END; /* E04 */       ASM04770
        GO TO BINST;                                                 ASM04780
      END;                                                           ASM04790
/*_____________________________________________________________*/ASM04800
/*_____________________________________________________________*/ASM04810
/*    IF THE OPERAND CONTAINS AN IDENTIFIER, FIND ITS ADDRESS OF     ASM04820
      DECLARATION.    */                                             ASM04830
                                                                     ASM04840
      IF OPERA(1) >='A' & OPERA(1) < '0' THEN DO;                    ASM04850
        M=IC;                                                        ASM04860
        IP=INDEX(OPER,'+');                                          ASM04870
        IM=INDEX(OPER,'-');                                          ASM04880
        IF IP¬=0 THEN M=IP;                                          ASM04890
        IF IM¬=0 THEN M=IM;                                          ASM04900
```

```
        IF M=0 THEN M=7;                                                ASM04910
        CONST=SUBSTR(OPER,1,M-1);                                       ASM04920
        ADR=SEARCH (CONST,PC); IF ADR=0 THEN DO; ERR=3; GO TO OUTPUT;   ASM04930
        END;                                                            ASM04940
        IF IP=0 & IM=0 THEN GO TO BINST;                                ASM04950
        IF IC=0 THEN IC=INDEX(OPER,' ');                                ASM04960
        IF IC=0 THEN IC=26;                                             ASM04970
        IC=IC-(M+1);                                                    ASM04980
        IF IC <=0 THEN DO; ERR=4; GO TO OUTPUT; END;  /* ERR04 */       ASM04990
        IF VERIFY(SUBSTR(OPER,M+1,IC),DIGIT)¬=0 THEN DO;                ASM05000
          ERR=11; GO TO OUTPUT; END;  /* ERR11 */                       ASM05010
        ADCON=SUBSTR(OPER,M+1,IC); ADR1=OCON(ADCON);                    ASM05020
        IF IP=0 THEN ADR=ADR-ADR1; ELSE ADR=ADR+ADR1;                   ASM05030
        GO TO BINST;                                                    ASM05040
      END;                                                              ASM05050
/*_____________________________________________________________________*/ASM05060
/*_____________________________________________________________________*/ASM05070
      IF (OPERA(1) >= '0' & OPERA(1)<='9') | OPERA(1)='-' THEN DO;      ASM05080
        IF OPERA(1)='-' THEN I=2; ELSE I=1;                             ASM05090
        M=IC;                                                           ASM05100
        IF M=0 THEN M=INDEX(OPER,' ');                                  ASM05110
        IF VERIFY (SUBSTR(OPER,I,M-I),DIGIT)¬=0 THEN DO;                ASM05120
          ERR=4; GO TO OUTPUT; END;  /* ERR04 */                       ASM05130
        ADCON=SUBSTR(OPER,I,M-I);                                       ASM05140
        ADR=OCON(ADCON);                                                ASM05150
        IF I=2 THEN ADR=-ADR;                                           ASM05160
        GO TO BINST;                                                    ASM05170
      END;                                                              ASM05180
      ERR=4; GO TO OUTPUT;   /* ERR04 */                                ASM05190
/*_____________________________________________________________________*/ASM05200
/*_____________________________________________________________________*/ASM05210
IOHDLR: M=INDEX (OPER,',');                                             ASM05220
      INSTR=IOP2(I);                                                    ASM05230
      IF OPERA(M+1)='4' THEN INSTR=INSTR|FWD;                           ASM05240
      GO TO OUTPUT;                                                     ASM05250
```

```
/*_____________________________________________________________________*/ASM05260
/*_____________________________________________________________________*/ASM05270
BOC:   ADR=OCCN(OPER); IF ADR=-1 THEN GO TO OUTPUT;                       ASM05280
BOC1:  BNUM=UNSPEC(ADR);                                                  ASM05290
       INSTR=INSTR|BNUM1;                                                 ASM05300
       GO TO OUTPUT;                                                      ASM05310
DECM:  M=INDEX(OPER,' ');                                                 ASM05320
       IF VERIFY (SUBSTR(OPER,1,M-1),DIGIT)¬=0 THEN DO;                   ASM05330
         ERR=4; GO TO  OUTPUT; END;  /* ERR04 */                         ASM05340
       ADR=SUBSTR(OPER,1,M-1);                                            ASM05350
       GO TO BOC1;                                                        ASM05360
EQUA:  ADR=SEARCH(LAB,PC); IF ADR=0 THEN DO; ERR=3; GO TO OUTPUT; END;    ASM05370
       GO TO BOUT;                                                        ASM05380
PAGES: PUT PAGE; PC=PC-1; GO TO LOOP;                                     ASM05390
/*_____________________________________________________________________*/ASM05400
/*_____________________________________________________________________*/ASM05410
BCDT:  JJ=2;                                                              ASM05420
BCDT2: DO I=0 TO 3;                                                       ASM05430
         ADR=INDEX(CHARSET,OPERA(I+JJ+1));                                ASM05440
         BNUM=UNSPEC(ADR-1);                                              ASM05450
         SUBSTR(INSTR,I*6+1,6)=VAL1;                                      ASM05460
       END;                                                               ASM05470
       PNUM=UNSPEC(PC);                                                   ASM05480
       IF JJ>2 THEN CARD=BLANK50;                                         ASM05490
       PUT SKIP EDIT(PC1,INSTRO,CARD)                                     ASM05500
         (5 F(1),X(5),8 F(1),X(7),A(80));                                 ASM05510
       OB=INSTR;  CALL PUNCH;                                             ASM05520
       IF JJ=2 & OPERA(1)>'4' THEN DO; JJ=6; PC=PC+1;                     ASM05530
         GO TO BCDT2; END;                                                ASM05540
       GO TO LOOP;                                                        ASM05550
/*_____________________________________________________________________*/ASM05560
/*_____________________________________________________________________*/ASM05570
RESV:  M=INDEX(OPER,' ');                                                 ASM05580
       IF VERIFY (SUBSTR(OPER,1,M-1),DIGIT)¬=0 THEN DO;                   ASM05590
         ERR=4; GO TO OUTPUT; END;  /* ERR04 */                          ASM05600
```

```
      ADR=SUBSTR(OPER,1,M-1);                                          ASM05610
      NFLG='1'B;                                                       ASM05620
      IF PFLG THEN FACT=4; ELSE FACT=16;                               ASM05630
      IF ADR*FACT+OB_PTR < OB_MAX THEN OB_PTR=OB_PTR+ADR*FACT;         ASM05640
                                  ELSE OB_PTR=144;                     ASM05650
      GO TO BINST;                                                     ASM05660
/*_____________________________________________________________________*/ASM05670
/*_____________________________________________________________________*/ASM05680
BOUT: PC=PC-1;  NFLG='1'B;                                             ASM05690
      PUT SKIP EDIT (CARD) (COL(26),A(80));                            ASM05700
      GO TO ERRCHK;                                                    ASM05710
BOUTA: PC=PC-1; NFLG='1'B;                                             ASM05720
      ADR=OCON(OPER);  IF ADR=-1 THEN GO TO OUTPUT;                    ASM05730
      BNUM=UNSPEC(ADR);                                                ASM05740
      INSTR=INSTR|BNUM1;                                               ASM05750
      PUT SKIP EDIT (INSTRO, CARD)                                     ASM05760
        (X(10), 8 F(1),X(7),A(80));                                    ASM05770
      GO TO ERRCHK;                                                    ASM05780
/*_____________________________________________________________________*/ASM05790
/*_____________________________________________________________________*/ASM05800
AORG: ADR=OCON(OPER); IF ADR=-1 THEN GO TO OUTPUT; PC=0;               ASM05810
      IF ¬FLG THEN OB_PTR=144;                                         ASM05820
      NFLG='1'B;                                                       ASM05830
BINST: BNUM=UNSPEC(ADR);                                               ASM05840
      INSTR=INSTR | (BNUM1 & ADRONE);                                  ASM05850
OUTPUT: PNUM=UNSPEC(PC);                                               ASM05860
/*_____________________________________________________________________*/ASM05870
/*_____________________________________________________________________*/ASM05880
/*    PRINT THE INSTRUCTION.    */                                     ASM05890
                                                                       ASM05900
      PUT SKIP EDIT(PC1,INSTRO,CARD)                                   ASM05910
        (5 F(1),X(5),8 F(1),X(7),A(80));                               ASM05920
      IF I=67 THEN PC=PC+ADR-1;                                        ASM05930
      IF I=68 THEN PC=ADR-1;                                           ASM05940
      IF NFLG THEN GO TO ERRCHK;                                       ASM05950
```

```
/*_____________________________________________________________*/ASM05960
/*_____________________________________________________________*/ASM05970
/*    PUNCH THE INSTRUCTION'S OBJECT CODE.    */                  ASM05980
                                                                  ASM05990
      OB=INSTR;  CALL PUNCH;                                      ASM06000
ERRCHK: IF ERR=0 THEN GO TO LOOP;                                 ASM06010
/*_____________________________________________________________*/ASM06020
/*_____________________________________________________________*/ASM06030
/*    PRINT ANY ERRORS IN THE INSTRUCTION.    */                  ASM06040
                                                                  ASM06050
      PUT SKIP(2) EDIT (ERRTAB(ERR))(A(20));                      ASM06060
      ERRCTR=ERRCTR+1;                                            ASM06070
      ERRLINE(ERRCTR)=PC+ADR;                                     ASM06080
      GO TO LOOP;                                                 ASM06090
/*_____________________________________________________________*/ASM06100
/*_____________________________________________________________*/ASM06110
COUTPT:  PUT SKIP EDIT (CARD) (X(25),A(80));                      ASM06120
      PC=PC-1;                                                    ASM06130
      GO TO LOOP;                                                 ASM06140
/*_____________________________________________________________*/ASM06150
/*_____________________________________________________________*/ASM06160
FINI: CLOSE FILE (PROG);                                          ASM06170
     IF ¬ENDFLG THEN DO;                                          ASM06180
       OB_PTR=144; ENDFLG='1'B; EFLG='1'B;                        ASM06190
       CALL PUNCH;  CALL SEQ;                                     ASM06200
       PUT SKIP EDIT('***END STATEMENT MISSING, END GENERATED') (A);  ASM06210
       END;                                                       ASM06220
      PUT SKIP EDIT ('DONE') (A);                                 ASM06230
/*_____________________________________________________________*/ASM06240
/*_____________________________________________________________*/ASM06250
/*    PRINT THE NUMBER OF ERRORS AND WHERE EACH OCCURRED.    */  ASM06260
                                                                  ASM06270
      PUT SKIP (4) EDIT                                           ASM06280
        ('THERE ARE ',ERRCTR,'ERRORS IN THIS LISTING')            ASM06290
        (COL(10),A(10),F(4),X(2),A(22));                          ASM06300
```

```
    DO I=1 TO ERRCTR;                                                   ASM06310
    PNUM=UNSPEC(ERRLINE(I));                                            ASM06320
    PUT SKIP EDIT('ADDRESS',PC1) (COL(10),A(7),X(2),5 F(1)); END;       ASM06330
     END PASS2;                                                         ASM06340
    CLOSE FILE(SORTIN);                                                 ASM06350
/*_____________________________________________________________________*/ASM06360
/*_____________________________________________________________________*/ASM06370
/*    IF A CROSS-REFERENCE TABLE IS DESIRED, PUT THE SYMBOL TABLE       ASM06380
     ON FILE (SYMX).    */                                              ASM06390
                                                                        ASM06400
    IF PARM='XREF' THEN DO I=1 TO SP;                                   ASM06410
      XTEMP=SYM(I);                                                     ASM06420
      WRITE FILE (SYMX) FROM (XTEMP);                                   ASM06430
    END;                                                                ASM06440
     END ASM;                                                           ASM06450
```

Variable Dictionary for ASM

A - a structure used to create the file SORTIN. A contains the name of an identifier and an address of reference for it. Element NN contains the name of the identifier, ADDR1 contains the address of reference, and FILLER is unused space necessary to make A 18 bytes long, which is the minimum record length for IHESRTA.

ADCON - contains an address constant given in the operand field of an instruction.

ADDR - see SYM.

ADDR1 - see A.

ADR - an address used in generating the object code.

ADR1 - the fixed binary equivalent of ADCON.

ADRONE - the bit string constant ('000000000011111111111111'B).

AND - the ten character operand field of an instruction.

B - array of 160 8-bit strings representing the object code to be punched on cards.

BLANK - string of 10 blanks.

BLANK50 - string of 50 blanks.

BLNK - string of 8 bits which contains column-binary code to be transferred to an element in B (one of B(12) through B(19)).

BLNK1 - the last 6 bits of BLNK.

BNUM - 32-bit string which contains the bit-string representation of the address of declaration of an identifier.

BNUM1 - the last 24 bits of BNUM.

BNUMB - the first character of the operand of an instruction.

CARD - see INST.

CARDI - the 10 character string containing the number to be converted to octal by OCON.

CB - the output file which receives the generated object code.

CHA - the formal parameter (character string) for OCON.

CHARSET - the array of characters allowed by the SDS-910 assembler language.

CHR_CBL - an array of 64 8-bit strings representing the allowed characters. The first two bits of each element are 0, and the last 6 are the first half of the Hollerith representation of the character.

CHR_CBR - an array of 64 8-bit strings representing the allowed characters. The first two bits of each element are 0, and the last 6 are the last half of the Hollerith representation of the character.

CODE - array of nine labels, each referring to a section of code to handle the occurrence of a macro corresponding to the label name. NUN, CODE (9), corresponds to the absence or completion of a macro.

COM - the first character of an instruction.

CONST - the name of an identifier in the operand of an instruction.

DIGIT - the character string '0123456789'.

DUMMY - the part of an operand preceding a comma.

EFLG - a bit that is set to '1' when the operand of an END statement is blank.

ENDFLG - a bit that is set to '1' when the END statement is encountered.

ERR - see INST.

ERRCRT - a counter of the number of errors in the assembler language program.

ERRLINE - an array containing the addresses of statements in which errors occurred.

ERRTAB - an array containing diagnostic messages for errors.

FACT - a multiplier for determining values of OB_PTR, 4 when using column-binary code and 16 when using Hollerith object code.

FILLER - see A.

FIVFOR - the bit string '101100101100101100101100' which is used to indicate the end of a program.

FLG - a bit that is '1' until PUNCH is called, then is reset to '0'.

FWD - the bit string '000000000000000110000000'.

I - a counter for DO statements and other loops.

IA - the address of declaration of an identifier found in SEARCH.

IADR - the decimal equivalent to the octal argument of OCON.

IB - counter used to reinitialize B to '000...0'.

IC - the position of the first comma (or blank) in the operand field of an instruction.

ID - a counter for DO statements.

II - the position of the first element in a sublist of the symbol table.

IJ - indicates the midpoint position in a list to be sorted.

IL - an array of size 11 containing the positions of the first elements of sublists to be sorted.

IM - the position of the first '-' in the operand field of an instruction.

INDIRECT - the bit string '000000000100000000000000'.

INST - a structure containing CARD, an 80 character instruction from the program to be assembled; and ERR, which contains an integer indicating the type of error that occurred in the statement (0, if no error).

INSTR - the object code translation of an instruction.

INSTRO - array of the 8 3-bit segments of INSTR.

IOP - array of 7 3-character input-output operators.

IOP2 - array of the 7 object code bit strings corresponding to the elements of IOP.

IP - the position of the first '+' in the operand field of an instruction.

IU - an array of size 11 containing the positions of the last elements of sublists to be sorted.

J - the position of the last element in a sublist of the symbol table.

JJ - an integer used to create code for the macro BCD.

K - an indicator of the position of an element in the symbol table as it is sorted.

KK - indicates which element of the Hollerith representation of the character set is to be put in a column of a card.

L - an indicator of the position of an element in the symbol table as it is sorted.

LL - counter for a DO statement.

LAB - the label field of an instruction (first six characters).

LABL - a formal parameter of SEARCH, the identifier whose address of declaration is being sought.

LLIMIT - the lower bound of the area of the symbol table being searched for an identifier.

LOC - the first six characters of an instruction.

M - the position of the first blank (or comma) in a specified field of an instruction.

MM - indicates which sublist of the symbol table is currently being created or sorted.

N - the position of an element of the symbol table being compared to the argument of SEARCH.

N1 - the value of an octal digit in the operand field of an instruction.

N2 - the position of the first blank in the operand field of an instruction.

NAM - see SYM.

NFLG - a bit that is '1' if the current instruction (involving a macro) is to be punched on the object deck.

NN - see A.

OB - stores the address and instruction to be punched on the object deck.

OB3 - array of the 8 3-bit segments of OB.

OB6 - array of the 4 6-bit segments of OB.

OB_MAX - 144, the first position of B that cannot contain information from OB.

OB_PTR - indicates the element of B that the current object code is to be stored in.

ONUM - the 32-bit string of code for a temporary operator.

ONUM1 - the last 24 bits of ONUM.

OP - the first four characters of the operator of an instruction.

OPA - a 72 element array of the first three letters of the possible operators.

OPB - an array of the 9-bit object codes corresponding to the first 31 elements of OPA.

OPB2 - an array of the 24-bit object codes corresponding to elements 32 through 61 of OPA.

OPC - the first four characters of the operator of an instruction.

OPC1 - array of the 8 4-character macro operators.

OPCODE - the first three characters of the operator of an instruction.

OPER - the first ten characters of the operand field of an instruction.

OPER1 - the first six characters of the operand field of an instruction.

OPERA - an array of the first ten characters of the operand field.

OPL - an array of 11 labels used to dictate branching when certain operators are encountered.

OPR - an array of the ten characters of CARDI.

OVRF - the bit string '100000000000000000000000'.

PARM - the first five characters of the first card of the assembly-language deck. If PARM='XREF ' a cross-reference listing will be printed.

PC - a counter which gives the address of the current instruction.

PC1 - an array of the last five 3-bit segments of PNUM.

PCA - a formal parameter of SEARCH which is an address of reference for the identifier whose address of declaration is being sought.

PCB - a counter from which the addresses of declaration for the symbol table are taken.

PFLG - a bit that is '1' if the object code is to be in column-binary form and '0' if the object code is to be in Hollerith form.

PNAME - the name of the assembler language program.

PNUM - the 32-bit string representation of the address of the current instruction.

PROG - an external file on which the instructions are written during the first pass of the assembler and from which they are read during the second pass.

SEQ_NO - a counter used for putting sequence numbers on the punched object deck.

SORTIN - an external file which contains the list of identifiers and all their addresses of reference; it is used when a cross-reference listing is requested.

SP - a counter for the number of declarations of identifiers encountered.

SW - a bit that is '1' if a new identifier is being added to the symbol table and '0' otherwise.

SW1 - a bit that is '1' if comparisons against the top element of a list are necessary during a particular phase of sorting.

SYM - an array of 400 structures which contains the symbol table of the program. NAM contains the name of an identifier and ADDR contains the address of declaration of the identifier.

SYMX - an external file that passes the symbol table to the program that generates a cross-reference listing.

T - a structure used as temporary storage for an element of the symbol table while it is being sorted into alphabetical order. TN contains the name of the identifier, and TA contains its address of declaration.

TA - see T.

TCODE - see TOPER.

TEMP - a 24-bit string that temporarily stores an object code instruction while its address is being put into array B.

TMAX - see TOPER.

TMP - the octal equivalent of the address in an AORG statement. It is used to update the location counter.

TN - see T.

TOP - see TOPER.

TOPC - contains the name of a temporary or illegal operator.

TOPER - a structure which contains information about temporary operators. TCODE is an array of 30 24-bit strings which represent the object code for the temporary operators. TMAX is the number of temporary operators in the program. TOP is an array of the names assigned to the temporary operators.

TT - a structure used as temporary storage for an element of the symbol table while it is being sorted into alphabetical order. TTN contains the name of an identifier, and TTA contains its address of declaration.

TTA - see TT.

TTN - see TT.

ULIMIT - the upper bound on the area of the symbol table being searched for an element during SEARCH.

VAL1 - the last six bits of BNUM.

X - the bit string '010000000000000000000000'.

XADDR - see XTEMP.

XNAM - see XTEMP.

XTEMP - a structure used to transfer the symbol table into the file SYMX. XNAM contains the name of an identifier and XADDR contains its address of declaration.

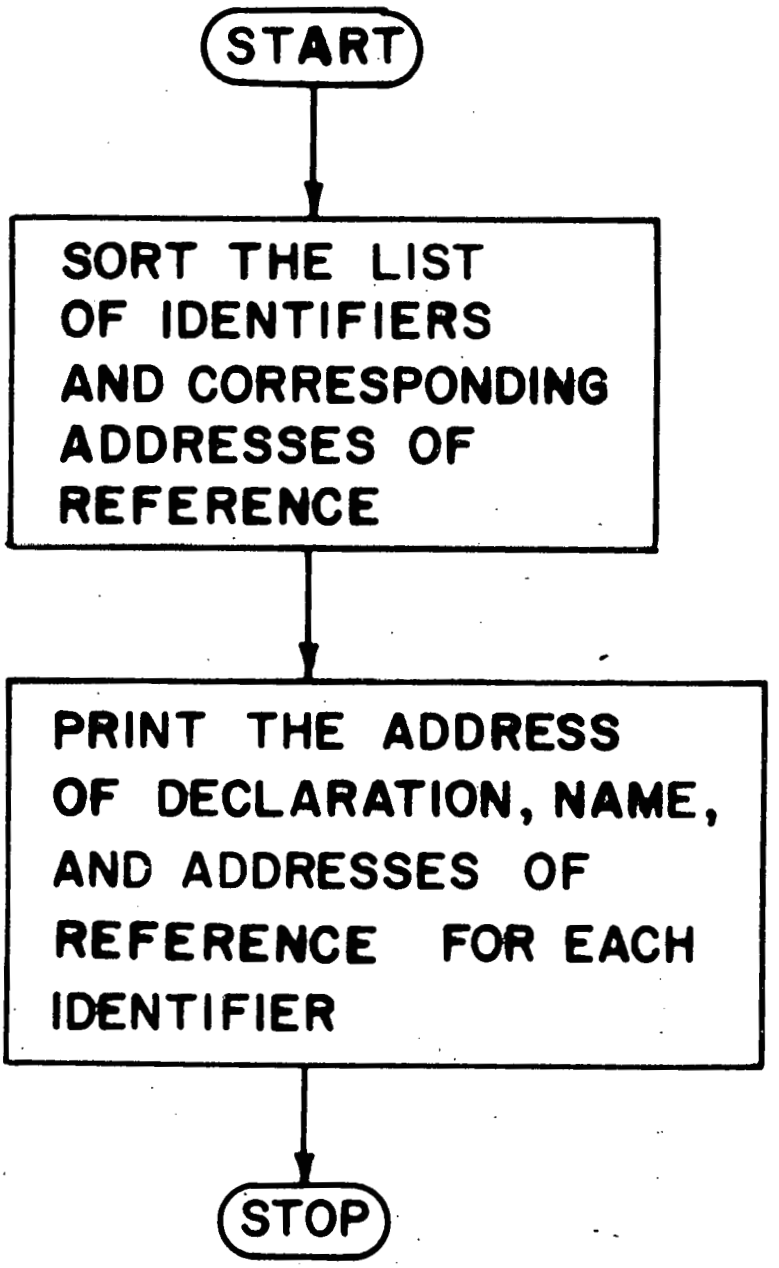
ZERO15 - the bit-string '000000000000000'.

ZERO24 - the bit-string '000000000000000000000000'.

Method of Solution for XREF

IHESRTA, the PL/I interface to IBM'S sort-merge program is used to put the ordered pairs of identifier names and addresses of reference into alphabetical order according to the identifier names. Entries with the same identifier name are ordered numerically according to the address of reference. The external file SORTIN, which is constructed in ASM is used as input to IHESRTA.

The symbol table is also constructed in ASM and is transferred to XREF by the external file SYMX. Each element of the symbol table is printed along with its address of declaration. The identifier name is compared with the name field of the address of reference ordered pairs and the addresses of reference paired with an identifier name are then printed along with it.



FLOWCHART FOR XREF THE SDS-910 CROSS-REFERENCE LISTING PROGRAM

Source Listing for XREF

```
XREF:  PROC OPTICNS (MAIN);                                          XREF0010
/* SORTS ORDERED PAIRS SO THEY CAN BE LISTED. */                      XREF0020
    DCL 1 SYM,                                                        XREF0030
          2 NAM CHAR(6),                                              XREF0040
          2 ADDR FIXED BIN;                                           XREF0050
        DCL PNUM BIT (32) STATIC;                                     XREF0060
        DCL PC1(5) BIT(3) DEF PNUM POS(2);                            XREF0070
  DCL PNUM1 BIT (32) STATIC;                                          XREF0080
  DCL PC2(5) BIT(3) DEF PNUM1 POSITION(18);                           XREF0090
  DCL IHESRTA ENTRY(CHAR(33),CHAR(27),FIXED BIN (31,0),               XREF0100
     FIXED BIN (31,0)), RETURN_CODE FIXED BIN(31,0);                  XREF0110
  DCL 1 A,                                                            XREF0120
     2 NN CHAR(6),                                                    XREF0130
     2 ADDR1 FIXED BIN (31,0),                                        XREF0140
     2 FILLER (4) FIXED BIN (15,0);                                   XREF0150
  DCL K FIXED BIN;                                                    XREF0160
  DCL J BIT(1) INIT('0'B);                                            XREF0170
    ON ENDFILE (SYMX) GO TO ENDSORT;                                  XREF0180
/*_____________________________________________________________*/XREF0190
/*_____________________________________________________________*/XREF0200
/*    SORT THE LIST OF IDENTIFIERS AND CORRESPONDING ADDRESSES OF   XREF0210
      REFERENCE.    */                                                XREF0220
                                                                      XREF0230
     CALL IHESRTA(' SORT FIELDS=(1,6,CH,A,7,4,FI,A) ',                XREF0240
       ' RECORD TYPE=F,LENGTH=(18) ', 4000,RETURN_CODE);              XREF0250
     IF RETURN_CODE=16 THEN DO;                                       XREF0260
       PUT SKIP EDIT ('SORT FAILED') (A);                             XREF0270
     GO TO ENDSORT; END;                                              XREF0280
     ON ENDFILE (SORTOUT) GO TO TESTI;                                XREF0290
     PUT PAGE EDIT('CROSS REFERENCE LISTING') (A);                    XREF0300
/*_____________________________________________________________*/XREF0310
/*_____________________________________________________________*/XREF0320
/*    PRINT THE ADDRESS OF DECLARATION, NAME, AND ADDRESSES OF      XREF0330
      REFERENCE FOR EACH IDENTIFIER.    */                            XREF0340
                                                                      XREF0350
```

```
     READ FILE (SYMX) INTO  (SYM);                                      XREF0360
LOOP: READ FILE (SORTOUT) INTO (A);                                     XREF0370
     PNUM1=UNSPEC(A.ADDR1);                                             XREF0380
LOOPER:  PNUM=UNSPEC(SYM.ADDR);                                         XREF0390
     DO WHILE (NN>SYM.NAM);                                             XREF0400
     IF ¬J THEN                                                         XREF0410
       PUT SKIP(2) EDIT (PC1,SYM.NAM)                                   XREF0420
         (5 F(1),X(3),A(6),X(4));                                       XREF0430
     J='0'B;                                                            XREF0440
     READ FILE (SYMX) INTO (SYM);                                       XREF0450
     PNUM=UNSPEC(SYM.ADDR);                                             XREF0460
     END;                                                               XREF0470
     PNUM=UNSPEC(SYM.ADDR);                                             XREF0480
     IF NN=SYM.NAM THEN DO;                                             XREF0490
     IF ¬J THEN DO;                                                     XREF0500
 /* PRINT NAME FIRST */                                                 XREF0510
     PUT SKIP(2) EDIT(PC1,SYM.NAM,PC2) (5 F(1),X(3),A(6),X(4),5 F(1));  XREF0520
    K=1;                                                                XREF0530
     J='1'B; GO TO LOOP; END;                                           XREF0540
    K=K+1;                                                              XREF0550
    IF MOD(K,17)=1 THEN DO;                                             XREF0560
      PUT EDIT (',') (A(1));                                            XREF0570
      PUT SKIP EDIT (PC2) (COLUMN(19),5 F(1));                          XREF0580
      GO TO LOOP;                                                       XREF0590
    END;                                                                XREF0600
     PUT EDIT(',',PC2) (A(1),5 F(1)); GO TO LOOP; END;                  XREF0610
     /* IGNORE RECORD, SYNTAX ERROR RECORD */                           XREF0620
  GO TO LOOP;                                                           XREF0630
/*_____________________________________________________________________*/XREF0640
/*_____________________________________________________________________*/XREF0650
TESTI:                                                                  XREF0660
    DO WHILE ('1'B);                                                    XREF0670
      READ FILE (SYMX) INTO (SYM);                                      XREF0680
      PNUM=UNSPEC(SYM.ADDR);                                            XREF0690
      PUT SKIP(2) EDIT(PC1,SYM.NAM) (5 F(1),X(3),A(6));                 XREF0700
```

```
END;                                                    XREF0710
 ENDSORT: END XREF;                                     XREF0720
```

Variable Dictionary for XREF

A - a structure that receives elements from the sorted list of identifiers and their addresses of reference. NN contains the name of the identifier, ADDR1 contains an address of reference, and FILLER is unused space necessary to make A 18 bytes long which corresponds to the length of records in the file SORTOUT.

ADDR - see SYM.

ADDR1 - see A.

FILLER - see A.

J - a bit that is '1' if the identifier's name has already been printed and is '0' otherwise.

K - a counter that determines when a printed line is filled (17 addresses).

NAM - see SYM.

NN - see A.

PC1 - an array of the five digits of the address of declaration of an identifier.

PC2 - an array of the five digits of an address of reference of an identifier.

PNUM - a 32-bit string containing the address of declaration of an identifier.

RETURN_CODE - an actual parameter returned by IHESRTA, the built-in PL/I interface with IBM's sort-merge program. RETURN_CODE=0 if the sort worked and 16 if the sort failed.

SORTOUT - the alphabetically and numerically sorted list of identifiers and their corresponding addresses of reference which is generated by IHESRTA. SORTOUT is an external file.

SYM - a structure that contains an entry of the symbol table. NAM contains the name of an identifier and ADDR contains its address of declaration.

SYMX - an external file used to pass the symbol table from ASM to XREF.

Appendix A

Usage

Accessing the cross-assembler reference generating package requires only three control cards. If the use of the reference generating routine is not desired, the JOB card must be followed by these two JCL cards:

```
//S1 EXEC SDSASM
//ASM.SYSIN DD *
```

and the next card must contain a blank in the first column if Hollerith format punched object code is desired, or any character in the first column if column-binary format punched object code is desired. Columns six through eighty are reserved for the program's name. This card is followed by the program to be assembled.

The control cards necessary if the use of the reference generating routine is desired are similar. Following the JOB card must be:

```
//S1 EXEC SDSXREF
//ASM.SYSIN DD *
```

and the next card must contain XREF in the first four columns. Again, columns six through eighty of this card are reserved for the program's name. The assembler language program is placed after this card.

The first part of the output of this package is a list of multiply defined labels (if any). This is immediately followed by the listing of the source program. Each line of this listing contains the address of the instruction, the octal object code for the instruction, and the assembler language source instruction itself.

Error messages appear under the instruction in which the assembler detected the error(s). These are marked by three asterisks. The source listing of the program is followed by a statement of the number of errors in the program and a list of the addresses at which these errors occurred.

If the cross-reference listing is requested, the next page will contain messages generated by IHESRTA. For more information see pp. 189-205 of the IBM System/360 Operating System PL/I (F) Programmer's Guide. These messages are followed by a blank page and the cross-reference listing itself. Each entry first contains the address of declaration of the identifier, followed by the identifiers name, and then a list of the addresses at which the identifier is referenced.

Sample Results

```
***MULTIPLY DEFINED LABEL      Z
                          *
                          *    THIS RCUTINE ILLUSTRATES THE USE OF THE SDS 910 ASSEMBLER.
                          *
00000      00000000       XX      OCT      0
00001      00000002       Y       OCT      2
C0002      00000000       Z       OCT      A

***ILLEGAL OCT NUM
00003      07600000               LDA      XX

***MISSING LABEL
00004      04100000               BRX      *-4
00000      00005450               AORG     5450
05450      01300000               POT      CCC

***MISSING LABEL
05451      07500001               LDB      Y
05452      01400002               ETR      Z
05453      00232301       Z       EOM      32301
05454      00000000               XYZ      Y+2

***ILLEGAL OP CODE
05455      04305456               BRM      SYS
05456      20347001       SYS     OCT      20347001
                                  END
DONE


          THERE ARE     4   ERRORS IN THIS LISTING
          ADDRESS   00001
          ADDRESS   00003
          ADDRESS   05450
          ADDRESS   05454
```

CROSS REFERENCE LISTING

05456	SYS	05455
00000	XX	00003
00001	Y	05451
00002	Z	05452
05453	Z	

Appendix B

Extension to another Assembly Language

With certain changes, this package could be used to assemble programs written in other assembly languages. In the main procedure ASM it would be necessary to change the definitions of some variables to reflect the assembly language's format for the cards containing the source program. The lengths of the variables that contain all or part of the object code would also have to be changed to correspond to the word length of the particular computer. In addition, the list of macro instructions, along with the section of code which performs the modifications commanded by them, would have to be altered.

In the internal subprocedure PASS2, the list of acceptable operators and the object code representations for them would have to be changed to fit the set of operators in the assembly language. Finally, the bit string constants which are used in the generation of the object code would also have to be altered.

The procedure XREF requires no changes to function for a different assembly language.