File No. S360-21
Form C28-6595-1

OS

IBM Systems Reference Library

IBM System/360 Operating System Assembler (E) Programmer's Guide

This publication complements the IBM System/360 Operating System Assembler Language publication. It provides a guide to program assembling, linkage editing, executing, interpreting listings, and assembler programming considerations.

PREFACE

This publication is a guide to the use of IBM provided cataloged procedures for assembling; assembling and linkage editing; assembling, linkage editing, and executing assembler language source programs. This edition is oriented to the E level assembler program (the assembler) functioning in the IBM System/360 Operating System sequential scheduling environment.

Other System Reference Library publications in the IBM System/360 Operating System series provide fuller, more detailed discussions of the topics introduced in this publication: a careful reading of the publication IBM System/360 Operating System: Concepts and Facilities, Form C28-6535, is recommended. Knowledge of the assembler language is assumed. Where appropriate, the reader is directed to the following publications:

IBM System/360 Operating System: Job Control Language, Form C28-6539

IBM System/360 Operating System: Linkage Editor, Form C28-6538

IBM System/360 Operating System: Control Program Services, Form C28-6541

IBM System/360 Operating System: Assembler Language, Form C28-6514

IBM System/360 Operating System: Utilities, Form C28-6586

IBM System/360 Operating System: Control Program Messages and Completion Codes, Form C28-6608

IBM System/360 Operating System: FORTRAN IV (E), Library Subprograms, Form C28-6596

IBM System/360 Operating System: System Programmers Guide, Form C28-6550

IBM System/360 Operating System: FORTRAN IV (E) Programmer's Guide, Form C28-6603

IBM System/360 Operating System: COBOL (E) Programmer's Guide, Form C24-5029

Second Edition

This edition is a reprint of C28-6595-0 and incorporates changes released in Technical Newsletter N28-2140, dated March 31, 1966.

Significant changes or additions to the specifications contained in this publication will be reported in subsequent revisions or Technical Newsletters.

This publication was prepared for production using an IBM computer to update the text and to control the page and line format. Page impressions for photo-offset printing were obtained from an IBM 1403 Printer using a special print chain.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form is provided at the back of this publication for reader's comments. If the form has been removed, comments may be addressed to IBM Corporation, Programming Publications, Department 452, San Jose, California 95114.

CONTENTS

ILLUSTRATIONS

FIGURES

TABLES

CHARTS

Through the medium of job control statements, the programmer specifies job requirements directly to the operating system, thus eliminating many of the functions previously performed by the machine operator or other installation personnel. The job consists of one or more job steps. For example, the job of assembling, linkage-editing, and executing a source program involves three job steps:

1. Translating the source program, i.e., executing the assembler component of the operating system to produce an object module.

2. Processing the output of the assembler, i.e., executing the linkage-editor component of the operating system to produce a load module.

3. Executing the assembled and linkage-edited program, i.e., executing the load module.

A procedure is a sequence of job control language statements specifying a job. Procedures may enter the system via the input stream or from a library of procedures, which are previously defined and contained in a procedure library. The input stream is the flow of job control statements and, optionally, input data entering the system from one input device. At the sequential scheduling system level of the operating system, only one input stream may exist at a time. (For a description of the operating system environment see IBM System/360 Operating System: Concepts and Facilities.)

The job definition (JOB), execute (EXEC), data definition (DD), and delimiter (/*) job control statements are shown in this publication as they are used to specify assembler processing. Detailed explanations of these statements are given in IBM System/360 Operating System: Job Control Language.

Operating system factors influencing program preparation, such as program termination, saving and restoring general registers, and linking of independently produced object modules are discussed in "Programming Considerations" as are guides to determine whether assembler dictionary sizes and source statement complexity limitations will be exceeded.

The balance of this introductory section discusses the assembler options, data sets, and return codes.

ASSEMBLER OPTIONS

The programmer may specify the following assembler options in the PARM= field of the EXEC statement:

```
      DECK   LOAD   LIST   TEST   XREF
PARM=(NODECK,NOLOAD,NOLIST,NOTEST,NOXREF,
                                LINECNT=nn)
```

These options are defined as follows:

DECK[1] -- The object module is placed on the device specified in the SYSPUNCH DD statement.

LOAD[1] -- The object module is placed on the device specified in the SYSPUNCH DD statement.

LIST -- An assembler listing is produced.

TEST -- The object module (if produced) contains the special source symbol table required by the test translator (TESTRAN) routines.

XREF -- The assembler produces a cross-reference table of symbols as part of the listing.

The prefix NO is used with the above options to indicate that the option is not wanted. If contradictory options are entered, e.g., LIST,NOLIST, the rightmost option, e.g., NOLIST is used. DECK and LOAD can be contradictory.

LINECNT=nn

specifies the number of lines to be printed between headings in the listing. The permissible range is 01 to 99 lines.

[1] The assembler, during a single execution, produces either an object module in punched card form, or an object module in intermediate storage. The UNIT= designation in the SYSPUNCH DD statement determines where the object module is placed. Because of this the DECK and LOAD options are interchangeable. If both are specified the rightmost entry is used: If DECK,NOLOAD is specified, no object deck is produced.

DEFAULT ENTRY

If no options are specified, the assembler assumes the following default entry:

```
PARM=(NOLOAD,DECK,LIST,NOTEST,XREF,
                              LINECNT=56)
```

The cataloged procedures discussed in this guide assume the default entry. However, the programmer may override any or all of the default options (see "Overriding Cataloged Procedures").

ASSEMBLER DATA SET REQUIREMENTS

Seven data sets must be defined for the assembler; they are described in the following text. The ddname that must be used in the DD statement describing the data set appears as the heading for each description.

Ddname SYSLIB

From this data set, the assembler obtains macro definitions and assembler language statements to be called by the COPY assembler instruction. It is a partitioned data set and each macro definition or sequence of assembler statements is a separate member with the member name being the macro-instruction mnemonic or COPY code name. The data set may be defined as SYS1.MACLIB or a user's private macro definition or COPY library. SYS1.MACLIB contains macro definitions for the system macro-instructions provided by IBM. A user's private library may be concatenated with SYS1.MACLIB. The Job Control Language publication explains data set concatenation.

Ddnames SYSUT1, SYSUT2, SYSUT3

These utility data sets are used by the assembler when processing the source program. The input/output device(s) assigned to these data sets must be capable of sequential access to records: the assembler does not support multi-volume utility data sets.

Ddname SYSPRINT

This data set is used by the assembler to produce a listing. Output may be directed to a printer or magnetic tape. The assembler uses the machine code carriage-control characters for this data set.

Ddname SYSPUNCH

The assembler uses this data set to produce the object module. The input/output unit assigned to this data set may be either a card punch or an intermediate storage device (capable of sequential access). In the same execution, the assembler cannot produce a punched card object module and an object module on intermediate storage.

Ddname SYSIN

This data set contains the input to the assembler -- the source statements to be processed. The input/output device assigned to this data set is either the device transmitting the input stream, or a device designated by the programmer. The DD statement describing this data set usually appears in the input stream. The IBM supplied procedures do not contain this statement.

RETURN CODES

Table 1 shows the return codes issued by the assembler for use with the COND= parameter[1] of JOB or EXEC statements.

Table 1. Return Codes

Return Code	Explanation
0	no errors detected
4	minor errors detected; successful program execution is probable
8	errors detected; unsuccessful program execution is possible
12	serious errors detected; unsuccessful program execution is probable
16	critical errors detected; normal execution is impossible
20	unrecoverable I/O error occurred during assembly; assembly terminated

[1] The COND parameter is explained in the Job Control Language publication.

The return code issued by the assembler is the highest severity code that is:

a. Associated with any error detected by the assembler.[1]

b. Associated with MNOTE messages produced by macro-instructions.

c. Associated with an unrecoverable I/O error occurring during the assembly.

The return code of 20 is used only for condition code testing. It is not associated with any diagnostic messages.

[1] See Appendix A for diagnostic messages and severity codes.

CATALOGED PROCEDURES

This section describes three IBM provided cataloged procedures: a procedure for assembling (ASMEC); a procedure for assembling and linkage editing (ASMECL); a procedure for assembling, linkage editing, and executing (ASMECLG). The procedures rely on conventions regarding the naming of device classes. These conventions, shown in Table 2, must be incorporated into the system at system generation time.

Table 2. Device Naming Conventions

Device Classname	Devices Assigned
SYSSQ	Any devices allowing sequential access to records for reading and writing
SYSDA	Direct-access devices
SYSCP	Card punches

To use cataloged procedures, an EXEC statement(s) naming the desired procedure(s) is placed in the input stream following the JOB statement. Subsequently, the specified cataloged procedure is brought from a procedure library and merged into the input stream.

The System Programmer's Guide discusses the placing of procedures in the procedure library.

CATALOGED PROCEDURE FOR ASSEMBLY (ASMEC)

This procedure requests the operating system to load and execute the assembler (IETASM). The name ASMEC must be used to call this procedure. The result of execution is an object module in punched card form, and an assembler listing.

In the following example, input enters via the input stream. The statements entered in the input stream to use this procedure are:

```
//jobname   JOB

//stepname  EXEC PROC=ASMEC

//ASM.SYSIN DD    *
                |
                |
      source program statements
                |
                |
/* (delimiter statement)
```

The statements of the ASMEC procedure are brought from the procedure library and merged into the input stream.

Figure 1 shows the statements that make up the ASMEC procedure.

```
1 //ASM      EXEC PGM=IETASM

2 //SYSLIB   DD   DSNAME=SYS1.MACLIB,DISP=OLD

3 //SYSUT1   DD   UNIT=SYSSQ,SPACE=(400,(400,50))

4 //SYSUT2   DD   UNIT=SYSSQ,SPACE=(400,(400,50))

5 //SYSUT3   DD   UNIT=(SYSSQ,SEP=(SYSUT1,SYSUT2,SYSLIB)),          X
  //              SPACE=(400,(400,50))

6 //SYSPRINT DD   SYSOUT=A

7 //SYSPUNCH DD   UNIT=SYSCP
           ----------
           ----------
```

1 PARM= or COND= parameters may be added to this statement by the EXEC statement that calls the procedure (see "Overriding Cataloged Procedures"). The system name IETASM identifies Assembler E.

2 This statement identifies the macro library data set. The data set name SYS1.MACLIB is an IBM designation.

3 4 5 These statements specify the assembler utility data sets. The device classname used here, SYSSQ, may represent a collection of tape drives, or direct-access units, or both. The I/O units assigned to this name are specified by the installation when the system is generated. A unit name, e.g., 2311 may be substituted for SYSSQ.

The SEP= subparameter in statement 5 and the SPACE= parameter in statements 3,4, and 5 are effective only if the device assigned is a direct-access device: otherwise they are ignored. The space required is dependent on the make-up of the source program. the procedure provides an initial allocation of 160,000 bytes and additional allocations (if needed) of 20,000 bytes.

6 This statement defines the standard system output class, SYSOUT=A, as the destination for the assembler listing.

7 This statement describes the data set that will contain the object module produced by the assembler.

Figure 1. Cataloged Procedure for Assembly

CATALOGED PROCEDURE FOR ASSEMBLY AND LINKAGE-EDITING (ASMECL)

This procedure consists of two job steps: assembling and linkage editing. The name ASMECL must be used to call this procedure. Execution of this procedure results in the production of an assembler listing, a linkage editor listing, and a load module.

The following example assumes input to the assembler via the input job stream. It also makes provision in the //LKED job step for concatenating the input to the linkage editor from the //ASM job step with any additional linkage editor input in the input job stream. This additional input can be a previously produced object module which is to be linked to the object module produced by job step //ASM.

The statements entered in the input stream to use this procedure are:

```
//jobname     JOB

//stepname    EXEC PROC=ASMECL

//ASM.SYSIN   DD    *
                |
                |
          source program statements
                |
                |
/*

//LKED.SYSIN  DD    *  \
                |       | necessary only if
                |       | linkage-editor is
        object module  <  to combine modules
                |       |
                |       |
/*                     /
```

The procedure is brought from the procedure library and merged into the input stream.

Figure 2 shows the statements that make up the ASMECL procedure. Only those statements not previously discussed are explained.

```
  //ASM       EXEC PGM=IETASM

  //SYSLIB    DD   DSNAME=SYS1.MACLIB,DISP=OLD

  //SYSUT1    DD   UNIT=SYSSQ,SPACE=(400,(400,50))

  //SYSUT2    DD   UNIT=SYSSQ,SPACE=(400,(400,50))

  //SYSUT3    DD   UNIT=(SYSSQ,SEP=(SYSUT1,SYSUT2,SYSLIB)),             X
  //               SPACE=(400,(400,50))

  //SYSPRINT DD    SYSOUT=A

1 //SYSPUNCH DD    DSNAME=&LOADSET,UNIT=SYSSQ,SPACE=(80,(200,50)),      X
  //               DISP=(MOD,PASS)

2 //LKED      EXEC PGM=IEWL,PARM=(XREF,LIST,NCAL)

3 //SYSLIN    DD   DSNAME=&LOADSET,DISP=(OLD,DELETE)
4 //          DD   DDNAME=SYSIN

5 //SYSLMOD   DD   DSNAME=&TEMP(PDS),UNIT=SYSDA,SPACE=(1024,(50,20,1))

6 //SYSUT1    DD   UNIT=(SYSDA,SEP=(SYSLIN,SYSLMOD)),SPACE=(1024,(50,20))

7 //SYSPRINT DD    SYSOUT=A
```

[1] In this procedure the SYSPUNCH DD statement describes a temporary data set -- the object module -- which is to be passed to the linkage editor.

[2] This statement initiates linkage editor execution. The linkage editor options in the PARM= field cause the linkage editor to produce a cross-reference table, module map, and a list of all control statements processed by the linkage editor. The NCAL option suppresses the automatic library call function of the linkage editor.

[3] This statement identifies the linkage editor input data set as the same one produced as output by the assembler.

[4] This statement is used to concatenate any input to the linkage editor from the input stream with the input from the assembler.

[5] This statement specifies the linkage-editor output data set (the load module). As specified, the data set will be deleted at the end of the job. If it is desired to retain the load module, the DSNAME parameter must be respecified and a DISP parameter added. See "Overriding Catalog Procedures". If the output of the linkage editor is to be retained, the DSNAME parameter must specify a library name and member name where the load module is to be placed. The DISP parameter must specify either KEEP or CATLG.

[6] This statement specifies the utility data set for the linkage editor.

[7] This statement identifies the standard output class as the destination for the linkage editor listing.

Figure 2. Cataloged Procedure for Assembling and Linkage Editing

CATALOGED PROCEDURE FOR ASSEMBLY, LINKAGE-EDITING, AND EXECUTION (ASMECLG)

This procedure consists of three job steps: assembling, linkage editing, and executing. The name ASMECLG must be used to call this procedure. Assembler and linkage editor listings are produced.

The statements entered in the input stream to use this procedure are:

```
//jobname    JOB

//stepname   EXEC PROC=ASMECLG

//ASM.SYSIN  DD    *
             |
             |
    source program statements
             |
             |
/*

//LKED.SYSIN DD    *
             |
             |          necessary only if
      object module     linkage editor is
             |          to combine modules
             |
/*

//GO.ddname  DD    (parameters)

//GO.ddname  DD    (parameters)

//GO.ddname  DD    *                only if
             |                      necessary
             |

   problem program input
             |
             |
/*
```

Figure 3 shows the statements that make up the ASMECLG procedure. Only those statements not previously discussed are explained in the figure.

OVERRIDING STATEMENTS IN CATALOGED PROCEDURES

EXEC and DD statements appearing in cataloged procedures can be overridden, in full or part. Such overriding of statements or fields is effective only for the duration of the job step in which the statements appear. The statements, as stored in the procedure library of the system, remain unchanged.

Overriding for the purposes of respecification, addition, or nullification is accomplished by including in the input stream statements containing the desired changes and identifying the statements to be overridden.

EXEC Statements

The PARM= and COND= parameters can be added or, if present, modified by including in the EXEC statement calling the procedure the notation PARM.stepname=, or COND.stepname=, followed by the desired change. "Stepname" identifies the EXEC statement within the procedure to which the modification applies. Overriding the PGM= parameter is not possible.

If the procedure consists of more than one job step, a PARM.stepname= or COND.stepname= parameter may be entered for each step. The entries must be in order, i.e., PARM.step1=, PARM.step2=, etc.

DD Statements

All parameters in the operand field of DD statements may be overridden by including in the input stream (following the EXEC card calling the procedure) a DD statement with the notation //stepname.ddname in the name field. "Stepname" refers to the job step in which the statement identified by "ddname" appears.

Examples

In the assembly procedure ASMEC (Figure 1), the production of a punched object deck could be suppressed and the UNIT= and SPACE= parameters of data set SYSUT1 respecified, by including the following statements in the input stream:

```
//stepname    EXEC PROC=ASMEC,              X
//                 PARM.ASM=NODECK

//ASM.SYSUT1 DD    UNIT=2311,               X
//                 SPACE=(200,(300,40))
```

In procedure ASMECLG (Figure 3) suppressing production of an assembler listing and adding the COND= parameter to the EXEC statement which specifies execution of the linkage editor might be desired. In this case, the EXEC statement in the input stream would appear as follows:

```
//stepname   EXEC PROC=ASMECLG,                X
//                PARM.ASM=NOLIST,             X
//                COND.LKED=(4,LT,ASM)
```

For current execution of procedure ASMECLG, no assembler listing would be produced, and execution of the linkage editor job step //LKED would be suppressed if the return code issued by the assembler (step ASM) was greater then 4.

The Job Control Language and System Programmer's Guide publications provide additional description of overriding techniques.

```
    //ASM       EXEC PGM=IETASM

    //SYSLIB    DD   DSNAME=SYS1.MACLIB,DISP=OLD

    //SYSUT1    DD   UNIT=SYSSQ,SPACE=(400,(400,50))

    //SYSUT2    DD   UNIT=SYSSQ,SPACE=(400,(400,50))

    //SYSUT3    DD   UNIT=(SYSSQ,SEP=(SYSUT1,SYSUT2,SYSLIB)),           X
    //             SPACE=(400,(400,50))

    //SYSPRINT DD   SYSOUT=A

    //SYSPUNCH DD   DSNAME=&LOADSET,UNIT=SYSSQ,SPACE=(80,(200,50)),     X
    //             DISP=(MOD,PASS)

[1] //LKED      EXEC PGM=IEWL,PARM=(XREF,LET,LIST,NCAL)

    //SYSLIN    DD   DSNAME=&LOADSET,DISP=(OLD,DELETE)
    //          DD   DDNAME=SYSIN

[2] //SYSLMOD   DD   DSNAME=&GOSET(GO),UNIT=SYSDA,SPACE=(1024,(50,20,1)), X
    //             DISP=(NEW,PASS)

    //SYSUT1    DD   UNIT=(SYSDA,SEP=(SYSLIN,SYSLMOD)),SPACE=(1024,(50,20))

    //SYSPRINT DD   SYSOUT=A

[3] //GO        EXEC PGM=*.LKED.SYSLMOD
              ----------
              ----------
```

[1] The LET linkage editor option specified in this statement causes the linkage editor to mark the load module as executable even though errors were encountered during processing.

[2] The output of the linkage editor is specified as a member of a temporary data set, residing on a direct-access device, and is to be passed to a succeeding job step.

[3] This statement initiates execution of the assembled and linkage edited program. The notation *.LKED.SYSLMOD identifies the program to be executed as being in the data set described in job step LKED by the DD statement named SYSLMOD.

Figure 3. Cataloged Procedure for Assembly, Linkage Editing, and Execution

The assembler listing, Figure 4, consists of five sections, ordered as follows: external symbol dictionary items; the source and object program statements; relocation dictionary items; symbol cross-reference table; and diagnostic messages.

In addition two statistical messages may appear in the listing. They are:

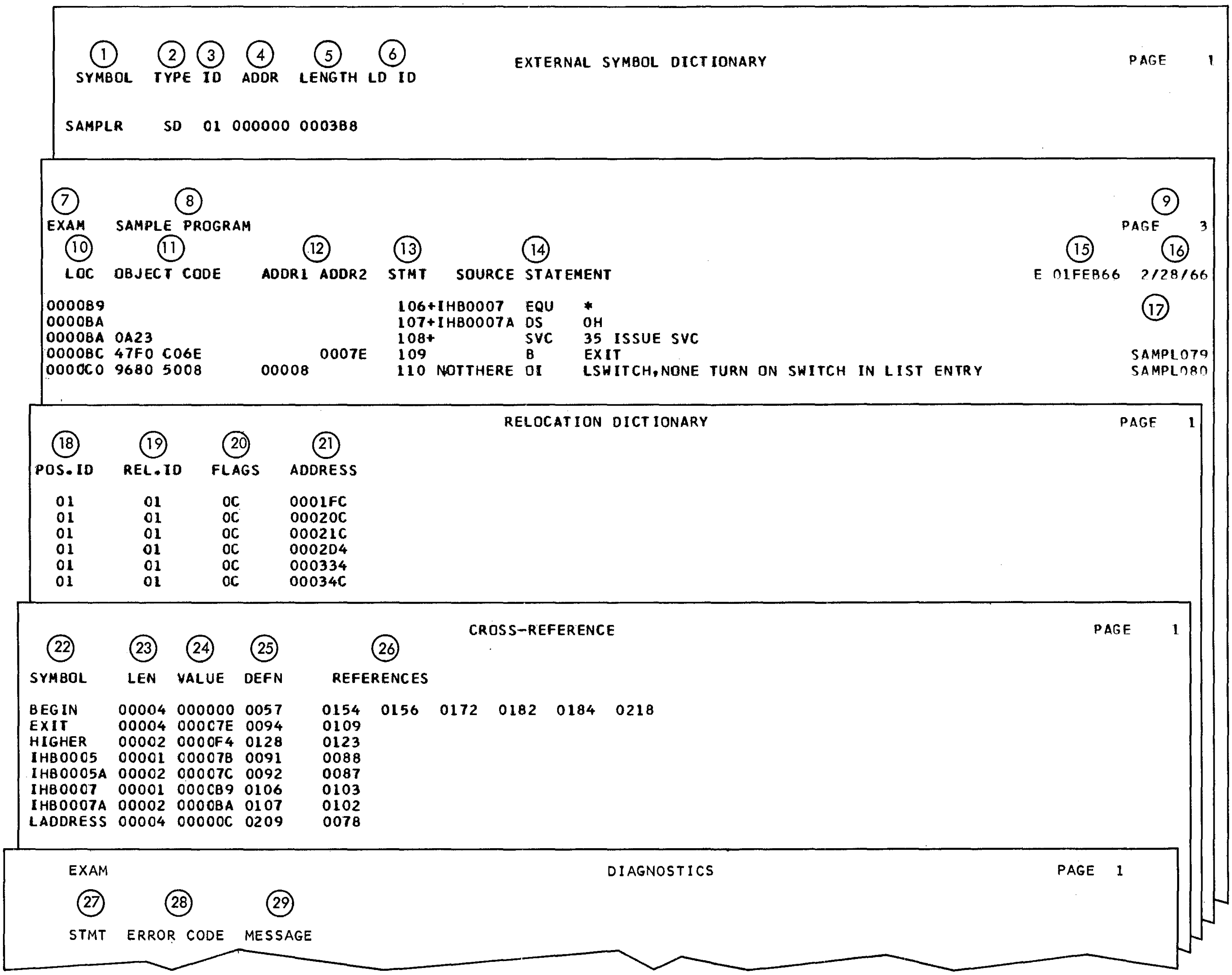
A message if one or more Y-type address constants appear in the program.

Message: AT LEAST ONE RELOCATABLE Y-TYPE CONSTANT IN ASSEMBLY.

A message indicating the total number of statements in error.

Message: nnn STATEMENTS FLAGGED IN THIS ASSEMBLY.

If issued, the Y-type address constant message appears before the diagnostic message section; the statements-flagged message appears after the diagnostics.


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29

```
SYMBOL  TYPE ID  ADDR   LENGTH LD ID         EXTERNAL SYMBOL DICTIONARY                          PAGE    1

SAMPLR   SD  01 000000 0003B8
```

```
EXAM    SAMPLE PROGRAM                                                                              PAGE    3
 LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                   E 01FEB66  2/28/66

0000B9                            106+IHB0007  EQU   *
0000BA                            107+IHB0007A DS    0H
0000BA 0A23                       108+         SVC   35 ISSUE SVC
0000BC 47F0 C06E         0007E    109          B     EXIT                                          SAMPL079
0000C0 9680 5008    00008         110 NOTTHERE OI    LSWITCH,NONE TURN ON SWITCH IN LIST ENTRY     SAMPL080
```

```
                                                RELOCATION DICTIONARY                               PAGE    1
POS.ID     REL.ID    FLAGS    ADDRESS

  01         01        0C      0001FC
  01         01        0C      00020C
  01         01        0C      00021C
  01         01        0C      0002D4
  01         01        0C      000334
  01         01        0C      00034C
```

```
                                            CROSS-REFERENCE                                     PAGE    1

SYMBOL      LEN   VALUE  DEFN      REFERENCES

BEGIN     00004 000000 0057     0154  0156  0172  0182  0184  0218
EXIT      00004 0000C7E 0094    0109
HIGHER    00002 0000F4 0128     0123
IHB0005   00001 00007B 0091     0088
IHB0005A  00002 00007C 0092     0087
IHB0007   00001 0000B9 0106     0103
IHB0007A  00002 0000BA 0107     0102
LADDRESS  00004 00000C 0209     0078
```

```
    EXAM                                               DIAGNOSTICS                         PAGE  1

    STMT   ERROR CODE    MESSAGE
```

Figure 4. Assembler Listing

EXTERNAL SYMBOL DICTIONARY (ESD)

This section of the listing contains the external symbol dictionary information passed to the linkage-editor in the object module. The entries described the control sections, external references, and entry points in the assembled program. There are five types of entries, shown in Table 3, along with their associated fields. The circled numbers refer to the corresponding heading in the sample listing (Figure 4).

Table 3. Types of ESD Entries

① SYMBOL	② TYPE	③ ID	④ ADDR	⑤ LENGTH	⑥ LDID
X	SD	X	X	X	-
X	LD	-	X	-	X
X	ER	X	-	-	-
-	PC	X	X	X	-
-	CM	X	X	X	-
The X indicates entries accompanying each type designation.					

①This column contains symbols that appeared in the name field of CSECT or START statements, as operands of ENTRY and EXTRN statements, or in the operand field of V-type address constants.

②This column contains the type designator for the entry, as shown in the table. The type designators are defined as:

SD -- names section definition. The symbol appeared in the name field of a CSECT or START statement.

LD -- The symbol appeared as the operand of an ENTRY statement.

ER -- external reference. The symbol appeared as the operand of an EXTRN statement, or was defined as a V-type address constant.

PC -- unnamed control section definition.

CM -- common control section definition.

③This column contains the external symbol dictionary identification number (ID). The number is a unique two digit hexadecimal number identifying the entry. It is used by the LD entry of the ESD and by the relocation dictionary to cross reference to the ESD.

④The column contains the address of the symbol (hexadecimal notation) for SD and LD type entries, and zeros for ER type entries. For PC and CM type entries, it indicates the beginning address of the control section.

⑤This column contains the assembled length, in bytes, of the control section (hexadecimal notation).

⑥This column contains, for LD type entries, the identification (ID) number assigned to the ESD entry that identifies the control section in which the symbol was defined.

SOURCE AND OBJECT PROGRAM

This section of the listing documents the source statements and the resulting object program.

⑦This is the deck identification. It is the symbol that appears in the name field of the first TITLE statement.

⑧This is the information taken from the operand field of a TITLE statement.

⑨Listing page number.

⑩ This column contains the assembled address (hexadecimal notation) of the object code.

⑪ This column contains the object code produced by the source statement. The entries are always left-justified. The notation is hexadecimal. Entries are machine instructions or assembled constants. Machine instructions are printed in full with a blank inserted after every four digits (two bytes). Constants may be only partially printed (see the PRINT assembler instruction in the Assembler Language publication).

⑫ These two columns contain effective addresses (the result of adding together a base register value and displacement value):

1. The column headed ADDR1 contains the effective address for the first operand of an SS instruction.

2. The column headed ADDR2 contains the effective address of the second operand of any instruction referencing storage.

Both address fields contain six digits; however, if the high order digit is a zero, it is not printed.

(13) This column contains the statement number. A plus sign (+) to the right of the number indicates that the statement was generated as the result of macro-instruction processing.

(14) This column contains the source program statement. The following items apply to this section of the listing:

a. Source statements are listed, including those brought into the program by the COPY assembler instruction, and macro-definitions submitted with the main program for assembly. Listing control instructions are not printed, except for the following case: PRINT is listed when PRINT ON is in effect and a PRINT statement is encountered.

b. Macro-definitions for system macro-instructions are not listed.

c. The statements generated as the result of a macro-instruction follow the macro-instruction in the listing.

d. Assembler or machine instructions in the source program that contain variable symbols are listed twice: as they appear in the source input, and with values substituted for the variable symbols.

e. Diagnostic messages are not listed in-line in the source and object program section. An error indicator, ***ERROR***, appears following the statement in error. The message appears in the diagnostic section of the listing.

f. MNOTE messages are listed in-line in the source and object program section. An MNOTE indicator appears in the diagnostic section of the listing. The MNOTE message format is: severity code, message text.

g. The MNOTE* form of the MNOTE statement results in an in-line message only. An MNOTE indicator does not appear in the diagnostic section of the listing.

h. When an error is found in a programmer macro-definition, it is treated like any other assembly error: the error indication appears after the statement in error, and a diagnostic is placed in the list of diagnostics. However, when an error is encountered during the expansion of a macro-instruction (system or programmer defined), the error indication appears in place of the erroneous statement, which is not listed. The error indication appears following the last statement listed before the erroneous statement was encountered, and the associated diagnostic message is placed in the list of diagnostics.

i. Literals that have been assigned locations by a LTORG statement appear in the listing following the END statement. Literals are identified by the equals (=) sign preceding them.

j. If the END statement contains an operand, the transfer address appears in the location column (LOC).

k. In the case of COM, CSECT, and DSECT statements, the location field contains the beginning address of these control sections i.e., the first occurrence.

l. For a USING statement, the location field contains the value of the first operand.

m. For LTORG and ORG statements, the location field contains the location assigned to the literal pool or the value of the ORG operand.

n. For an EQU statement the location field contains the value assigned.

o. Generated statements always print in normal statement format. Because of this, it is possible for a generated statement to occupy three or more continuation lines on the listing. This is unlike source statements which are restricted to two continuation lines.

(15) This field indicates the assembler level and release number for the month it was issued, e.g., E01FEB66 reads as Assembler E, first release of February 1966.

(16) Current date (date run is made).

(17) Identification-sequence field from the source statement.

RELOCATION DICTIONARY

This section of the listing contains the relocation dictionary information passed to the linkage editor in the object module. The entries describe the address constants in the assembled program that are affected by relocation.

⑱ This column contains the external symbol dictionary ID number assigned to the ESD entry that describes the control section in which the address constant is used as an operand.

⑲ This column contains the external symbol dictionary ID number assigned to the ESD entry that describes the control section in which the referenced symbol is defined.

⑳ The two-digit hexadecimal number in this column is interpreted as follows:

First Digit -- a zero indicates that the entry describes an A-type address constant.

-- a one indicates that the entry describes a V-type address constant.

Second Digit -- the first three bits of this digit indicate the length and sign of the address constant as follows:

Bits 0 and 1	Bit 2
00 = 1 byte	0 = +
01 = 2 bytes	1 = -
10 = 3 bytes	
11 = 4 bytes	

㉑ This column contains the assembled address of the field where the address constant is stored.

CROSS-REFERENCE

This section of the listing information concerns symbols -- where they are defined and used in the program.

㉒ This column contains the symbols.

㉓ This column states the length (decimal notation), in bytes, of the field occupied by the symbol value.

㉔ This column contains either the address the symbol represents, or a value to which the symbol is equated.

㉕ This column contains the statement number of the statement in which the symbol was defined.

㉖ This column contains the statement numbers of statements in which the symbol appears as an operand.

The following notes apply to the cross-referencing section:

- Symbols appearing in V-type address constants do not appear in the cross-reference listing.
- A PRINT OFF listing control instruction does not affect the production of the cross-reference section of the listing.
- Undefined symbols appear in the cross-reference section. However, only the symbol column and the reference column have entries.

DIAGNOSTICS

This section contains the diagnostic messages issued as a result of error conditions encountered in the program. Explanatory notes and the severity code for each message are contained in Appendix A.

㉗ This column contains the number of the statement in error.

㉘ This column contains the message identifier.

㉙ This column contains the message.

Example:

STMT	ERROR CODE	MESSAGE
101	IET035	ADDRESSABILITY ERROR

The following notes apply to the diagnostics section:

- An MNOTE indicator of the form MNOTE STATEMENT appears in the diagnostic section, if an MNOTE statement is issued by a macro-instruction. The MNOTE statement itself is in-line in the source and object program section of the listing.
- A message identifier consists of six characters and is of the form:

IETxxx

IET
identifies the issuing agent as assembler E.

xxx
is a unique number assigned to the message.

PROGRAMMING CONSIDERATIONS

This section consists of a number of discrete subjects about assembler language programming.

SAVING AND RESTORING GENERAL REGISTER CONTENTS

A problem program should save the values contained in the general registers upon commencing execution, and, upon completion, restore to the general registers these same values. Thus, as control is passed from the operating system to a problem program and in turn, to a subprogram, the status of the registers used by each program is preserved. This is done through use of the SAVE and RETURN system macro-instructions.

The SAVE macro-instruction should be the first statement in the program. It stores the contents of registers 14 and 15, and 0 through 12 in an area provided by the program passing control. When a problem program is given control, register 13 points to an area in which the general register contents should be saved.

If the program calls any subprograms, or uses any operating system services other than GETMAIN, FREEMAIN, ATTACH, and XCTL, it must first save the contents of register 13, and then the address of an 18 full-word save area must be loaded into register 13. This save area is in the problem program and is used by any subprograms or operating system services called by the problem program.

At completion, the problem program restores the contents of general registers 14, 15, and 0-12 by use of the RETURN system macro-instruction (which also indicates program completion). The content of register 13 must be restored before execution of the RETURN macro-instruction.

The coding sequence that follows illustrates the basic process of saving and restoring the registers. A complete discussion of the SAVE and RETURN macro-instructions and the saving and restoring of registers is contained in IBM System/360 Operating System: Control Program Services.

Name	Operation	Operand
BEGIN	SAVE	(14,12)
	ST	13,SAVEBLK+4
	LA	13,SAVEBLK
	.	
	.	
	L	13,SAVEBLK+4
	RETURN	(14,12)
SAVEBLK	DC	18F'0'

PROGRAM TERMINATION

Completion of an assembler source program is indicated by using the RETURN system macro-instruction to pass control from the terminating program to the program that initiated it. The initiating program may be the operating system, or, if a subprogram issued the RETURN, the program that called it.

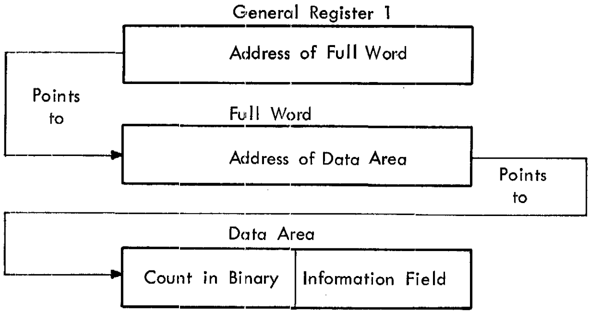
In addition to indicating program completion and restoring registers, the RETURN macro-instruction may also pass a return code - a condition indicator that may be used by the program receiving control. If the return is to the operating system, the return code is compared against the condition stated in the COND= parameter of JOB or EXEC statements. If return is to another problem program, the return code is available in general register 15, and may be used as desired. Register 13 should be restored before issuing the RETURN macro-instruction.

The RETURN system macro-instruction is discussed in detail in the Control Program Services publication.

PARM FIELD ACCESS

Access to information in the PARM field of an EXEC statement is gained through general register 1. When control is given to the problem program, general register 1 contains the address of a full word which, in turn, contains the address of the data area containing the information.

The data area consists of a half word containing the count (in binary) of the number of information characters, followed by the information field. The information field is aligned to a full-word boundary. The following diagram illustrates this process.



MACRO-DEFINITION LIBRARY ADDITIONS

Source statement coding to be retrieved by the COPY assembler instruction, and macro-definitions may be added to the macro-library. The IEBUPDAT utility program is used for this purpose. Details of this program and its control statements are contained in IBM System/360 Operating System: Utilities. The following sequence of job control statements can be used to call the utility program and identify the needed data sets. It is assumed that the job control statements, IEBUPDAT program control statements, and data are to enter the system via the input stream.

```
//jobname  JOB
//stepname EXEC PGM=IEBUPDAT,PARM=NEW
//SYSUT2   DD   DSNAME=SYS1.MACLIB,DISP=OLD
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
                .
                .
                .
  IEBUPDAT control statements and source
  statements or macro-definitions to be
  added to the macro-library (SYS1.MACLIB)
                .
                .
                .
/*(delimiter statement)
```

LOAD MODULE MODIFICATION-ENTRY POINT RESTATEMENT

If the editing functions of the linkage editor are to be used to modify a load module, the entry point to the load module must be restated when the load module is reprocessed by the linkage editor. Otherwise, the first byte of the first control section processed by the linkage editor will become the entry point. To enable restatement of the original entry point, or designation of a new entry point, the entry point must have been identified originally as an external symbol, i.e., appeared as an entry in the external symbol dictionary. External symbol identification is done automatically by the assembler if the entry point is the name of a control section or START statement; otherwise an assembler ENTRY statement must be used to identify the entry point name as an external symbol.

When a new object module is added to or replaces part of the load module, the entry point is restated in either of three ways:

- By placing the entry point symbol in the operand field of an EXTRN statement and an END statement in the new object module.
- By using an END statement in the new object module to designate a new entry point in the new object module.
- By using a linkage editor ENTRY statement to designate either the original entry point or a new entry point for the load module.

Further discussion of load module entry points is contained in the linkage editor publication.

OBJECT MODULE LINKAGE

Object modules, whether Assembler, FORTRAN or COBOL generated, may be combined by the linkage editor to produce a composite load module provided each object module conforms to the data formats and Linkage conventions required. This topic discusses the use of the CALL system macro-instruction to link an assembler language "main" program to subprograms produced by FORTRAN and COBOL.[1]

Figure 5 shows the statements used to establish the linkage from the assembler program to the called subprograms.

---

[1] See the Control Program Services publication for additional details concerning linkage conventions and the CALL system macro-instruction.

```
          SAVE   (14,12)
          ST     13,SVAREA+4
1         LA     13,SVAREA
          .
          .
          .
2         CALL   name,(V1,V2,V3),VL
          .
          .
          .
          L      13,SVAREA+4
          RETURN (14,12)
3 SVAREA  DC     18F'0'
4 V1      DC     (data)
5 V2      DC     (data)
6 V3      DC     (data)
          END
```

[1] The address of this program's (the calling program) save area is placed in general register 13 for use by the called subprogram.

[2] The symbol used for "name" in this statement is:

1. The name of a subroutine or function, when linking to a FORTRAN written subprogram.

2. The name defined by the following COBOL statements in the procedure division:

   ENTER LINKAGE. ENTRY'name'.

3. The name of a CSECT or START statement, or a name used in the operand field of an ENTRY statement in an assembler subprogram.

The order in which the parameter list is written must reflect the order in which the called subprogram expects the argument. If the called routine is a FORTRAN written function, the returned argument is not in the parameter list: a real or double precision function returns the value in floating point register zero; an integer function returns the value in general purpose register zero.

CAUTION: When linking to FORTRAN written subprograms, consideration must be given to the storage requirements of IBCOM (FORTRAN execution-time I/O and interrupt handling routines) which accompanies the compiled FORTRAN subprogram. In some instances the call for IBCOM is not automatically generated during the FORTRAN compilation. The FORTRAN IV Library publication provides information about IBCOM requirements and assembler statements used to call IBCOM.

FORTRAN written subprograms and FORTRAN library subprograms allow variable length parameter lists in linkages which call them; therefore all linkages to FORTRAN subprograms are required to have the high-order bit in the last parameter in the linkage set to 1. COBOL written subprograms have fixed length calling linkages; therefore, for COBOL the high order bit in the last parameter need not be set to 1.

[3] This statement reserves the save area needed by the called subprogram. When control is passed to the subprogram, register 13 contains the address of this area.

[4] [5] [6] When linking to a FORTRAN or COBOL subprogram, the data formats declared in these statements are determined by the data formats required by the FORTRAN or COBOL subprograms.

Figure 5. Linkage Statements

If any input/output operations are performed by called subprograms, appropriate DD statements for the data sets used by the subprograms must be supplied. See the FORTRAN (E) Programmer's Guide for explanation of the DD statements used to describe data sets for FORTRAN programs and a description of the special FORTRAN data set record formats. The COBOL (E) Programmer's Guide provides DD statement information for COBOL programs.

DICTIONARY SIZE AND SOURCE STATEMENT COMPLEXITY

The following material: (1) describes the composition of the assembler dictionaries and their entry sizes, and (2) describes methods for determining if the limits on source statement complexity will be exceeded.

Dictionary entries e.g., sequence symbol names or prototype symbolic parameters, vary in length. Therefore, the number of entries a dictionary can hold is determined by the types of entries.

Source statement complexity -- the number of symbols, characters, operators, delimiters, references to length attributes, self-defining terms, literals, and expressions appearing in a source statement -- determines whether or not the source statement can be successfully processed.

DICTIONARIES USED IN CONDITIONAL ASSEMBLY AND MACRO-INSTRUCTION EXPANSION

For the macro generator portion of the assembler to accomplish macro-instruction expansion and conditional assembly, two or more dictionaries must be constructed: a global dictionary and one or more local dictionaries.

These dictionaries take two forms: one which is used at the time the dictionary entries are collected, i.e., picked up from the initial scan of the source program: and one which is used during the actual conditional assembly and macro generation process. The next five topics describe the global and local dictionaries at collection and generation time.

Global Dictionary at Collection Time

One global dictionary is built for the entire program. It contains macro-instruction mnemonics and global SET variable symbols. One entry is made for each unique global SET variable symbol. One entry is made for each macro-instruction mnemonic that is not defined in the program; two identical entries are made when the macro-instruction mnemonic is referred to before it is defined; three identical entries are made when the macro-instruction mnemonic is defined before it is referred to. The capacity of the global dictionary is 64 blocks of 256 bytes each. Each block contains complete entries. Any entry not fitting into a block is placed in the next block; the remaining bytes in the current block are not used.

The size of each entry is shown in Table 4.

Table 4. Global Dictionary Entries at Collection Time

Entry	Size
Each macro mnemonic operation code	10 bytes plus mnemonic*
Each global SET variable symbol	6 bytes plus name*
*One byte is used for each character in the name or mnemonic.	

Fixed overhead for this dictionary is:

8 bytes for the first block
4 bytes for each succeeding block
5 bytes for the last block

There is a limit of 400 unique global symbols per assembly, regardless of the amount of storage available.

Local Dictionary at Collection Time

For the main portion of the program, (those statements not within a macro definition) one local dictionary is constructed in which ordinary symbols (relevant to macro generation and conditional assembly), sequence symbols, and local SET variable symbols are entered. Relevant ordinary symbols are those which occur in macro-instructions or conditional assembly statements. In addition, one local dictionary is constructed for each different macro definition in the program. These local dictionaries contain one entry for each local SET variable symbol, sequence symbol, and prototype symbolic parameter declared within the macro definition.[1] The capacity of each local dictionary is 64 blocks of 256 bytes each. Each block contains complete entries. Any entry not fitting into a block is placed in the next block; the remaining bytes in the current block are not used. Table 5 indicates the size of each type of entry and relates dictionary capacities to the structure of any given program.

Table 5. Local Dictionary Entries at Collection Time

Entry	Size
Each sequence symbol	10 bytes plus name*
Each local SET variable symbol	6 bytes plus name*
Each prototype symbolic parameter	5 bytes plus name*
Each relevant ordinary symbol appearing in the main portion of the program	10 bytes plus name*
*One byte is used for each character in the name or mnemonic	

Fixed overhead for this dictionary is:

8 bytes for the first block (if in the main program)

32 bytes for the first block (if in a macro definition)

4 bytes for each succeeding block

5 bytes for the last block

Global Dictionary at Generation Time

The structure of the global dictionary at generation time is shown in Table 6.

Table 6. Global Dictionary Entries at Generation Time

Entry	Size
Each macro mnemonic operation code	3 bytes
Each global SETA symbol (dimensioned)	1 byte plus 4N*
Each global SETA symbol (undimensioned)	4 bytes
Each global SETB symbol (dimensioned)	1 byte plus (N/8)* (N/8 is rounded to the next highest integer)
Each global SETB symbol (undimensioned)	1 byte
Each global SETC symbol (dimensioned)	1 byte plus 9N*
Each global SETC symbol (undimensioned)	9 bytes
*N=dimension	

Fixed overhead for this dictionary is 4 bytes plus word alignment.

Local Dictionary at Generation Time

The structure of the local dictionary at generation time is shown in Table 7.

[1] If a sequence symbol is defined before it is referenced, an extra entry for the symbol is made.

Table 7. Local Dictionary Entries at Generation Time

Entry	Size
Each sequence symbol	5 bytes
Each local SETA symbol (dimensioned)	1 byte plus 4N*
Each local SETA symbol (undimensioned)	4 bytes
Each local SETB symbol (dimensioned)	1 byte plus (N/8)* (N/8 is rounded to the next highest integer)
Each local SETB symbol (undimensioned)	1 byte
Each local SETC symbol (dimensioned)	1 byte plus 9N*
Each local SETC symbol (undimensioned)	9 bytes
Each relevant ordinary symbol[1] appearing in the main portion of the program	5 bytes

[1] For the main program Local Dictionary only those symbols which appear in macro-instruction operands or whose attributes are referenced are included.

*N=dimension

Fixed overhead for this dictionary is 20 bytes plus word alignment.

Additional Dictionary Requirements

The generation time global dictionary and the generation time local dictionary for the main portion of the program must be resident in main storage.

In addition, if the program contains any macro-instructions, main storage is required for the largest local dictionary of the macro-definitions being processed. Furthermore, during processing of macro-definitions containing inner macro-instructions, main storage is required for the generation time local dictionaries for the inner macro-instructions contained within the macro-definition.

MACRO-DEFINITION LOCAL DICTIONARY REQUIREMENTS: In addition to those requirements specified for the local dictionary of the main portion of the program, each macro-definition local dictionary requires space for the entries shown in Table 8.

Table 8. Macro-Definition Local Dictionary Parameter Table

Entry	Size
Each character string(1)	3 bytes plus L
Each hexadecimal, binary, decimal, and character self-defining term(2)	7 bytes plus L
Each symbol(3)	9 bytes plus L
Each sublist	10 bytes plus 2N bytes plus Y

L = Length of entry in bytes
N = Number of entries in sublist
Y = Total length of the table entries in formats 1,2,and 3

Fixed overhead for the macro-definition local dictionary parameter table is 22 bytes. Each nested macro-instruction also requires space in its local dictionary for the following:

Parameter pointer list	2 bytes plus 2N (N = the number of operands)
Pointers to list in the parameter table	8 bytes plus word alignment

MACRO MNEMONIC TABLE

As the source statements are scanned, a table of macro-instruction mnemonics is constructed in which there is an entry for each macro-instruction used or defined in the program. The entries are made under the premise that every undefined operation is a system macro-instruction mnemonic. This table is then used to locate and edit system macro-definitions from the library.

With 15,360 bytes of main storage available to the assembler, approximately 430 distinct macro-instruction mnemonics can be handled. An entry in this table consists of nine bytes. In the event that this table overflows, processing continues with only those macro-instructions defined to the point of overflow.

SOURCE STATEMENT COMPLEXITY

The complexity of a source statement is limited by both the macro-generator and assembler portions of Assembler E. The following topics provide the information necessary to determine if statement complexity limitations for either portion of the assembler are being exceeded.

Macro-Generation and Conditional Assembly Limitations

For any statement which:

1. Is a conditional assembly statement
2. Is a DC or DS statement
3. Is an EXTRN statement
4. Contains a sequence symbol or a variable symbol
5. Is not a macro-instruction or prototype statement

the total number of literal occurrences of

6. Ordinary symbols (includes machine mnemonics, assembler mnemonics, conditional assembly mnemonics, and macro-instruction mnemonics)
7. Variable symbols
8. Sequence symbols

must not exceed 35 in the name, operation, or operand fields respectively; and the number of literal occurrences of items 6, 7, and 8 above must not exceed 36 for the entire statement.

For macro-instructions and prototype statements the number of occurrences of ordinary symbols, variable symbols and sequence symbols must not exceed 35 in the name and operation fields combined, or in each operand unless the operand is a sublist in which case the limit is applied to each sublist operand.

Examples of counts:

```
&B2 SETB (T'NAME EQ 'W' OR '&C'.'A' EQ 'AA'
    count=4

EXTRN A,B,C,&C
    count=5
```

Assembler Portion Limitations

The space required to process a statement must not exceed 730 bytes for DC and DS statements, and 746 bytes for all others. Buffering considerations may allow statements exceeding these requirements by up to 30 bytes to be processed.

The following formulas (S_1 and S_2) are used to determine if statement complexity will exceed the limitations stated above. The statement must be tested against S_1 and S_2 and must satisfy both.

In general, all statements can be processed if they contain 50 or fewer terms. If a statement contains more than 50 terms, the formulas should be used to determine if the statement can be processed, or if the statement should be shortened using EQU assembler instructions. In the first example, if A+(B-C)*3 were equated to a symbol, that symbol could be used as the displacement field of the first operand in the example.

Formula S_1:

$$S_1=N_b+N_d+4(N_{1a}+N_{sd})+6(N_s+N_1)$$

where

N_b = total number of bytes in name, operation, operand, and comment entries (the maximum value of N is 187)

N_d = number of operators and delimiters in the operand field, except equal (=), period (.), and apostrophe(')

N_{1a} = number of references to length attribute (L'SYMBOL)

N_{sd} = number of self-defining terms

N_s = number of symbols (including*)

N_1 = number of literal operands (maximum of 1)

Example:

```
NAME MVC A+(B-C)*3(L'D,5),=15CL5'ABCDEFG'
```

$$S_1=\overset{N_b}{\overset{\downarrow}{39}}+\overset{N_d}{\overset{\downarrow}{9}}+4(\overset{N_{1a}}{\overset{\downarrow}{1}}+\overset{N_{sd}}{\overset{\downarrow}{4}})+6(\overset{N_s}{\overset{\downarrow}{3}}+\overset{N_1}{\overset{\downarrow}{1}})$$

$$S_1=92$$

Formula S_2:

$$S_2=N_b+9(W_1+W_2+...+W_m)+D$$

where:

N_b = as defined in formula S_1

W = a weight associated with each expression in the statement. The subscript represents the expression number; W_m is the last expression.

D = the number of expression delimiters

W may equal 1, 2, 3, 4, or 5 and is a function of the number of unpaired relocatable terms appearing in each expression as follows:

Number of Unpaired Terms	W
0, 1	1
2, 3, 4, 5	2
6, 7, 8, 9	3
10, 11, 12, 13	4
14, 15, 16	5

The rules for counting expressions and expression delimiters are as follows:

1. A comma is always an expression delimiter, as is the terminating blank.
2. Left and right parentheses can be part of an expression; or they can be expression delimiters. A left parenthesis is an expression delimiter if it is not preceded by an arithmetic operator or a blank. A right parenthesis is an expression delimiter if its paired left parenthesis is an expression delimiter.

Example 1:

NAME L 6,A+20*B(6)

$$S_2=\overset{\substack{N_b\\\downarrow}}{16} + 9(\overset{\substack{W_1\\\downarrow}}{1} + \overset{\substack{W_2\\\downarrow}}{1} + \overset{\substack{W_3\\\downarrow}}{1}) + \overset{\substack{D\\\downarrow}}{4}$$

$$S_2=47$$

In this example the comma, the two parentheses, and the terminating blank are expression delimiters. There are three expressions in this example:

(1) 6
(2) A+20*B
(3) 6

Expressions 1 and 3 are absolute and therefore have a weight (W) of 1. Expression 2 may be absolute or simply relocatable and therefore has a weight (W) of 1. (B must be absolute or the expression is in error.)

Example 2:

MVC A+17*(C-D),(A+20)

$$S_2=\overset{\substack{N_b\\\downarrow}}{20} + 9(\overset{\substack{W_1\\\downarrow}}{1} + \overset{\substack{W_2\\\downarrow}}{1}) + \overset{\substack{D\\\downarrow}}{2}$$

$$S_2=40$$

In this example the comma and the terminating blank are the only expression delimiters and D=2. There are two expressions:

Expression 1 = A+17*(C-D) with a weight (W) of 1

Expression 2 = (A+20) with a weight (W) of 1

Example 3:

MVC 20(5,3),16(5)

$$S_2=\overset{\substack{N_b\\\downarrow}}{16} + 9(\overset{\substack{W_1\\\downarrow}}{1} + \overset{\substack{W_2\\\downarrow}}{1} + \overset{\substack{W_3\\\downarrow}}{1} + \overset{\substack{W_4\\\downarrow}}{1} + \overset{\substack{W_5\\\downarrow}}{1}) + \overset{\substack{D\\\downarrow}}{7}$$

$$S_2=68$$

In this example there are 5 expressions (E) and 7 expression delimiters (ED).

E_1=20	ED_1=(
E_2=5	ED_2=,
E_3=3	ED_3=)
E_4=16	ED_4=,
E_5=5	ED_5=(
	ED_6=)
	ED_7=blank

This appendix lists the diagnostic messages issued by the assembler. The messages are listed by their number (001-109). Note: Explanations of the MNOTE messages issued by system macro-instructions are contained in the Messages and Completion Codes publication.

IET001 DUPLICATION FACTOR ERROR

Explanation: A duplication factor is not a positive absolute expression, or is zero in a literal.

Severity Code: 12

IET002 RELOCATABLE DUPLICATION FACTOR

Explanation: A relocatable expression has been used to specify the duplication factor.

Severity Code: 12

IET003 LENGTH ERROR

Explanation: The length specification is out of permissible range or specified invalidly.

Severity Code: 12

IET004 RELOCATABLE LENGTH

Explanation: A relocatable expression has been used to specify length.

Severity Code: 12

IET005 S-TYPE CONSTANT IN LITERAL

Severity Code: 8

IET006 INVALID ORIGIN

Explanation: The location counter has been reset to a value less than the starting address of the control section.

Severity Code: 12

IET007 LOCATION COUNTER ERROR

Explanation: The location counter has exceeded 2^{31}-1.

Severity Code: 12

IET008 INVALID DISPLACEMENT

Explanation: The displacement in an explicit address does not fall within the range of 0 to 4095.

Severity Code: 8

IET009 MISSING OPERAND

Severity Code: 12

IET010 INCORRECT REGISTER SPECIFICATION

Explanation: The value specifying the register is greater than 15, or an odd register is specified where an even register is required.

Severity Code: 8

IET011 SCALE MODIFIER ERROR

Explanation: The scale modifier is out of range.

Severity Code: 8

IET012 RELOCATABLE SCALE MODIFIER

Explanation: A relocatable expression has been used to specify the scale modifier.

Severity Code: 8

IET013 EXPONENT MODIFIER ERROR

Explanation: The exponent is not specified as an absolute expression or is out of range.

Severity Code: 8

IET014 RELOCATABLE EXPONENT MODIFIER

Explanation: A relocatable expression has been used to specify the exponent modifier.

Severity Code: 8

IET015 INVALID LITERAL USAGE

Explanation: A literal is used illegally. For example, it specifies a receiving field or a register.

Severity Code: 8

IET016 INVALID NAME

Explanation: A name entry is incorrectly specified. For example, it contains more than 8 characters, it does not begin with a letter, or has a special character imbedded.

Severity Code: 8

IET017 DATA ITEM TOO LARGE

Explanation: The constant is too large for the data type or for the explicit length.

Severity Code: 8

IET018 INVALID SYMBOL

Explanation: The symbol is specified invalidly. For example, it is longer than 8 characters.

Severity Code: 8

IET019 EXTERNAL NAME ERROR

Explanation: A CSECT and DSECT statement have same name, or a symbol used more than once in EXTRN.

Severity Code: 8

IET020 INVALID IMMEDIATE FIELD

Explanation: The value of the immediate operand exceeds 255, or the operand requires more than one byte of storage.

Severity Code: 8

IET021 SYMBOL NOT PREVIOUSLY DEFINED

Severity Code: 8

IET022 ESDTABLE OVERFLOW

Explanation: The combined number of control sections and dummy sections plus the number of unique symbols in EXTRN statements and V-type constants exceeds 255. If overflow is due to a V-type constant, message IET025 will also be issued.

Severity Code: 12

IET023 PREVIOUSLY DEFINED NAME

Explanation: The symbol which appears in the name field has appeared in the name field of a previous statement.

Severity Code: 8

IET024 UNDEFINED SYMBOL

Explanation: A symbol being referenced has not been defined in the program.

Severity Code: 8

IET025 RELOCATABILITY ERROR

Explanation: A relocatable or complex relocatable expression is specified where an absolute expression is required, or an absolute expression or complex relocatable expression is specified where a relocatable expression is required.

Severity Code: 8

IET026 TOO MANY LEVELS OF PARENTHESES

Explanation: An expression contains more than 5 levels of parentheses.

Severity Code: 12

IET027 TOO MANY TERMS

Explanation: More than 16 terms are specified in an expression.

Severity Code: 12

IET028 REGISTER NOT USED

Explanation: A register specified in a DROP statement is not currently in use.

Severity Code: 4

IET029 CCW ERROR

Explanation: Bits 37-39 of the CCW are set to nonzero.

Severity Code: 8

IET030 INVALID CNOP

Explanation: The operands are an invalid pair.

Severity Code: 12

IET031 UNKNOWN TYPE

Explanation: Incorrect type designation in a DC, DS or literal.

Severity Code: 8

IET032 OP-CODE NOT ALLOWED TO BE GENERATED

Severity Code: 8

IET033 ALIGNMENT ERROR

Explanation: Referenced address is not aligned to the proper boundary for this instruction.

Severity Code: 4

IET034 INVALID OP-CODE

Explanation: Syntax error: more than 8 characters in operation field; not followed by a blank on first card, etc.

Severity Code: 8

IET035 ADDRESSABILITY ERROR

Explanation: The referenced address does not fall within the range of a USING instruction.

Severity Code: 8

IET036 NO OPERAND ALLOWED

Severity Code: 4

IET037 MNOTE STATEMENT

Explanation: This indicates that an MNOTE statement has been generated from a macro definition. The text and severity code of the MNOTE statement will be found in line in the listing.

IET038 ENTRY ERROR

Explanation: A symbol in the operand of an ENTRY statement appears in more than one ENTRY statement, or is undefined, or is defined in a dummy section or in blank common, or is equated to a symbol defined by an EXTRN statement, or there are more than 100 ENTRY operands in the program.

Severity Code: 8

IET039 INVALID DELIMITER

Explanation: This message can be caused by:

1. Operands not separated by commas in assembler or machine instructions.

2. Last operand not followed by a blank.

3. Invalid sequence of operations and delimiters.

4. Incomplete exponent specification in DC or DS statement.

5. No data item specified between delimiters in a DC or DS statement.

6. No right parenthesis after an explicit base register expression in a S-type constant.

7. Absence of comma, blank, or left or right parenthesis where required in a machine instruction operand.

Severity Code: 12

IET040 STATEMENT TOO LONG

Severity Code: 12

IET041 UNDECLARED VARIABLE SYMBOL

Explanation: A variable symbol is not declared in a SET symbol statement or in a macro-instruction prototype statement.

Severity Code: 8

IET042 SINGLE TERM LOGICAL EXPRESSION IS NOT A SETB SYMBOL

Explanation: The single term logical expression has not been declared as a SETB symbol.

Severity Code: 8

IET043 SET SYMBOL PREVIOUSLY DEFINED

Severity Code: 8

IET044 SET SYMBOL USAGE INCONSISTENT WITH DECLARATION

Explanation: A set symbol has been declared as undimensioned, but is subscripted, or has been dimensioned, but is unsubscripted.

Severity Code: 8

IET045 ILLEGAL SYMBOLIC PARAMETER

Explanation: The system variable symbol is used in a macro-instruction prototype statement.

Severity Code: 8

IET046 AT LEAST ONE RELOCATABLE Y-TYPE CONSTANT IN ASSEMBLY

Severity Code: 4

IET047 SEQUENCE SYMBOL PREVIOUSLY DEFINED

Severity Code: 12

IET048 SYMBOLIC PARAMETER PREVIOUSLY DEFINED OR SYSTEM VARIABLE SYMBOL DECLARED AS SYMBOLIC PARAMETER

Severity Code: 12

IET049 VARIABLE SYMBOL MATCHES A PARAMETER

Severity Code: 12

IET050 INCONSISTENT GLOBAL DECLARATIONS

Explanation: A global SET variable symbol defined in more than one macro-definition, or defined in a macro-definition and in the source program, is inconsistent in SET type or dimension.

Severity Code: 8

IET051 MACRO DEFINITION PREVIOUSLY DEFINED

Explanation: Prototype operation field is the same as a machine or assembler instruction or a previous prototype.

Severity Code: 12

IET052 NAME FIELD CONTAINS ILLEGAL SET SYMBOL

Explanation: SET symbol in name field does not correspond to SET statement type.

Severity Code: 8

IET053 GLOBAL DICTIONARY FULL

Explanation: The global dictionary is full, assembly terminated. See "Dictionary Size and Source Statement Complexity."

Severity Code: 12

IET054 LOCAL DICTIONARY FULL

Explanation: The local dictionary is full, assembly terminated. See "Dictionary Size and Source Statement Complexity."

Severity Code: 12

IET055 INVALID ASSEMBLER OPTION(S) ON THE EXECUTE CARD

Severity Code: 8

IET056 ARITHMETIC OVERFLOW

Explanation: The intermediate or final result of an expression has exceeded $2^{31}-1$.

Severity Code: 8

IET057 SUBSCRIPT EXCEEDS MAXIMUM DIMENSION

Explanation: SYSLIST or symbolic parameter subscript exceeds 200, or is negative, or zero, or SET symbol subscript exceeds dimension.

Severity Code: 8

IET058 ILLEGAL LTORG

Explanation: LTORG appears in a COM or DSECT control section.

Severity Code: 8

IET059 UNDEFINED SEQUENCE SYMBOL

Severity Code: 12

IET060 ILLEGAL ATTRIBUTE NOTATION

Explanation: L', S', or I' requested for a parameter whose type attribute does not allow these attributes to be requested.

Severity Code: 8

IET061 ACTR COUNTER EXCEEDED

Severity Code: 12

IET062 GENERATED STRING GREATER THAN 255 CHARACTERS

Severity Code: 8

IET063 EXPRESSION 1 OF SUBSTRING IS ZERO OR MINUS

Severity Code: 8

IET064 EXPRESSION 2 OF SUBSTRING IS ZERO OR MINUS

Severity Code: 8

IET065 INVALID OR ILLEGAL TERM IN ARITHMETIC EXPRESSION

Explanation: The value of a SETC symbol used in an arithmetic expression is not composed of decimal digits; or, the parameter is not a self-defining term.

Severity Code: 8

IET066 UNDEFINED OR DUPLICATE KEYWORD OPERAND OR EXCESSIVE POSITIONAL OPERANDS

Explanation: The same keyword operand occurs more than once in a macro-instruction, or a keyword is not defined in a prototype statement; or, in a mixed mode macro-instruction, more positional operands are specified than are specified in the prototype.

Severity Code: 12

IET067 EXPRESSION 1 OF SUBSTRING GREATER THAN LENGTH OF CHARACTER EXPRESSION

Severity Code: 8

IET068 GENERATION TIME DICTIONARY AREA OVERFLOWED

Explanation: See "Dictionary Size and Source Statement Complexity."

Severity Code: 12

IET069 EXPRESSION 2 OF SUBSTRING GREATER THAN 8 CHARACTERS

Severity Code: 8

IET070 FLOATING POINT CHARACTERISTIC OUT OF RANGE

Severity Code: 12

IET071 ILLEGAL OCCURRENCE OF LCL, GBL OR ACTR STATEMENT

Explanation: LCL, GBL, or ACTR statement not in proper place in program.

Severity Code: 8

IET072 ILLEGAL RANGE ON ISEQ STATEMENT

Severity Code: 4

IET073 ILLEGAL NAME FIELD

Explanation: Either a statement which requires a name has been written without a name, or a statement has a name which is not allowed to have a name.

Severity Code: 8

IET074 ILLEGAL STATEMENT IN COPY CODE OR SYSTEM MACRO

Severity Code: 8

IET075 ILLEGAL STATEMENT OUTSIDE OF A MACRO DEFINITION

Severity Code: 8

IET076 SEQUENCE ERROR

Severity Code: 12

IET077 ILLEGAL CONTINUATION CARD

Explanation: Either there are too many continuation cards, or there are nonblanks between the begin and continue columns on the continuation card.

Severity Code: 8

IET078 MACRO MNEMONIC OP-CODE TABLE OVERFLOW

Explanation: See "Dictionary Size and Source Statement Complexity."

Severity Code: 12

IET079 ILLEGAL STATEMENT IN MACRO DEFINITION

Explanation: This operation is not allowed within a macro-definition.

Severity Code: 8

IET080 ILLEGAL START CARD

Explanation: Statements affecting or depending on the location counter have been encountered before a START statement.

Severity Code: 8

IET081 ILLEGAL FORMAT IN GBL OR LCL STATEMENTS

Explanation: An operand is not a variable symbol.

Severity Code: 8

IET082 ILLEGAL DIMENSION SPECIFICATION IN GBL OR LCL STATEMENT

Explanation: Dimension is other than 1 to 255.

Severity Code: 8

IET083 SET STATEMENT NAME FIELD NOT A VARIABLE SYMBOL

Severity Code: 8

IET084 ILLEGAL OPERAND FIELD FORMAT

Explanation: Syntax invalid; e.g., AIF statement operand does not start with a left parenthesis, or the operand of an AGO statement is not a sequence symbol, etc.

Severity Code: 8

IET085 INVALID SYNTAX IN EXPRESSION

Explanation: Invalid delimiter, too many terms in expression, too many levels of parentheses, or two operators in succession.

Severity Code: 8

IET086 ILLEGAL USAGE OF SYSTEM VARIABLE SYMBOL

Explanation: A system variable symbol appears in the name field of a SET statement, or is used in a mixed mode or keyword macro-definition, or is declared in a GBL or LCL statement, or is an unsubscripted &SYSLIST in a context other than N'&SYSLIST.

Severity Code: 8

IET087 NO ENDING APOSTROPHE

Explanation: There is an unpaired apostrophe in the statement.

Severity Code: 8

IET088 UNDEFINED OPERATION CODE

Severity Code: 12

IET089 INVALID ATTRIBUTE NOTATION

Explanation: Syntax error; e.g., the argument of the attribute reference is not a symbolic parameter inside a macro-definition.

Severity Code: 8

IET090 INVALID SUBSCRIPT

Explanation: Syntax error; e.g., double subscript where single subscript is required or vice versa, no right parenthesis after subscript, etc.

Severity Code: 8

IET091 INVALID SELF-DEFINING TERM

Explanation: Value is too large or is inconsistent with the data type.

Severity Code: 8

IET092 INVALID FORMAT FOR VARIABLE SYMBOL

Explanation: The first character after the ampersand is not alphabetic or the variable symbol contains more than 8 characters. (A single ampersand in a field or operand is assumed to start a variable symbol.)

Severity Code: 8

IET093 UNBALANCED PARENTHESES OR EXCESSIVE LEFT PARENTHESES

Severity Code: 8

IET094 INVALID OR ILLEGAL NAME OR OPERATION IN PROTOTYPE STATEMENT

Severity Code: 12

IET095 MESSAGE NOT DEFINED FOR THIS ERROR CODE

IET096 MACRO-INSTRUCTION OR PROTOTYPE OPERAND EXCEEDS 255 CHARACTERS IN LENGTH

Severity Code: 12

IET097 INVALID FORMAT IN MACRO-INSTRUCTION OPERAND OR PROTOTYPE PARAMETER

Explanation: This message can be caused by:

1. Illegal "="
2. A single "&" appears in the standard value assigned to a prototype keyword parameter.
3. First character of a prototype parameter is not "&".
4. Prototype parameter is a subscripted variable symbol.
5. Invalid usage of alternate format in prototype statement, e.g.,

```
10      16        72

PROTO   &A,&B,

            or

PROTO   &A,&B,    X

             &C
```

6. Unintelligible prototype parameter, e.g., "&A*" or "&A&&," etc.
7. Illegal (non-assembler) character appears in prototype parameter.

Severity Code: 12

IET098 EXCESSIVE NUMBER OF OPERANDS OR PARAMETERS

Explanation: Either the prototype has more than 200 parameters or, the macro-instruction has more than 200 operands.

Severity Code: 12

IET099 POSITIONAL MACRO-INSTRUCTION OPERAND, PROTOTYPE PARAMETER OR EXTRA COMMA FOLLOWS KEYWORD

Severity Code 12

IET100 STATEMENT COMPLEXITY EXCEEDED

Explanation: See "Dictionary Size and Source Statement Complexity."

Severity Code: 8

IET101 EOD ON SYSIN

Explanation: No END card before delimiter (/*) statement.

Severity Code: 12

IET102 INVALID OR ILLEGAL ICTL

Explanation: The operands of the ICTL are out of range, or the ICTL is not the first statement in the input deck.

Severity Code: 16

IET103 ILLEGAL NAME IN OPERAND FIELD OF COPY CARD

Explanation: Syntax error; e.g., symbol has more than 8 characters, or has an illegal character.

Severity Code: 12

IET104 COPY CODE NOT FOUND

Explanation: The operand of a COPY statement specified COPY text which cannot be found in the library.

Severity Code: 12

IET105 EOD ON SYSTEM MACRO LIBRARY

Explanation: MEND statement not in macro definition.

Severity Code: 12

IET106 MESSAGE NOT DEFINED FOR THIS ERROR CODE

IET107 INVALID OPERAND

Explanation: Unrecognizable operand in PRINT statement.

Severity Code: 4

IET108 PREMATURE EOD

Explanation: Indicates an internal assembler error; should not occur.

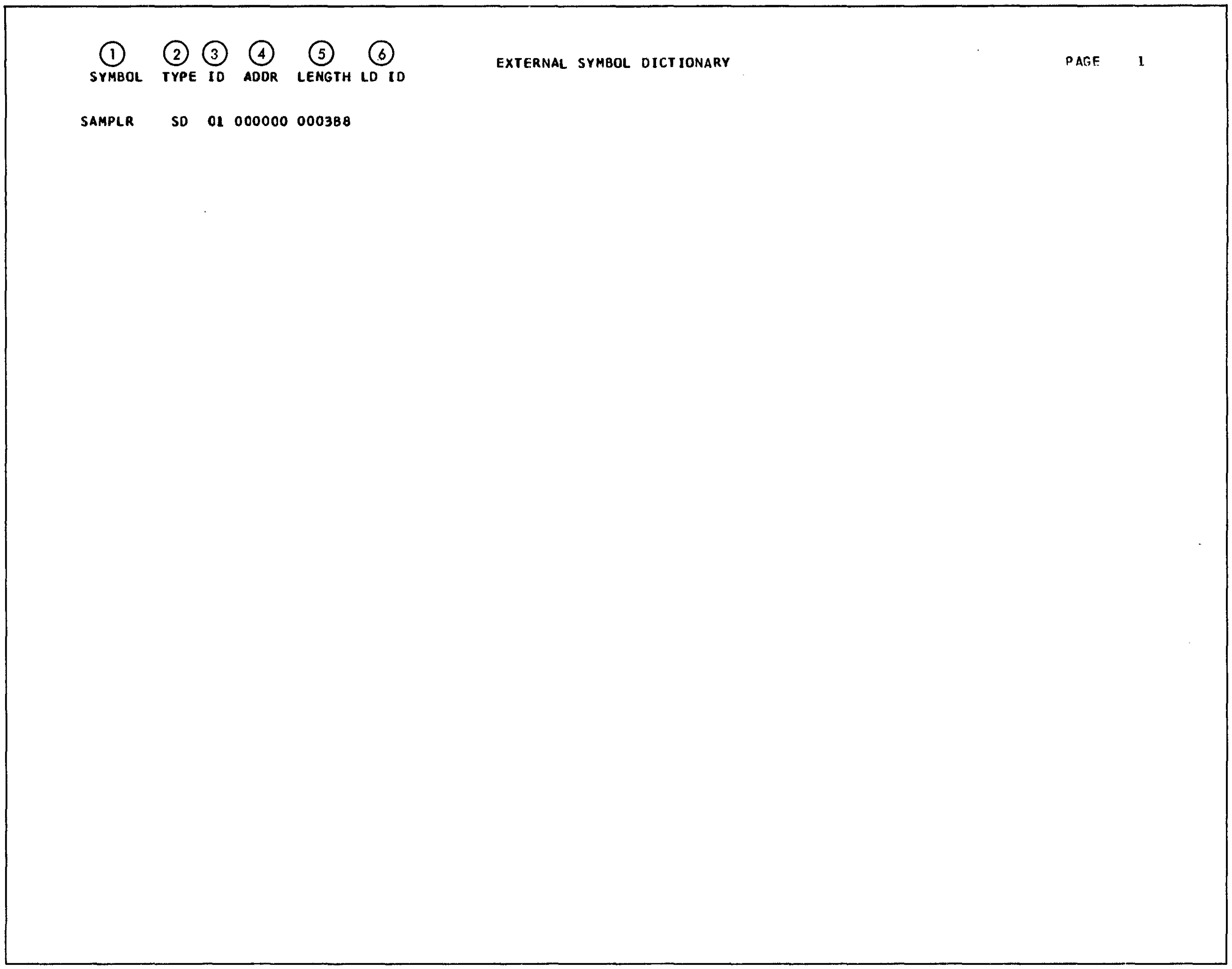
Severity Code: 16

IET109 PRECISION LOST

Severity Code: 8

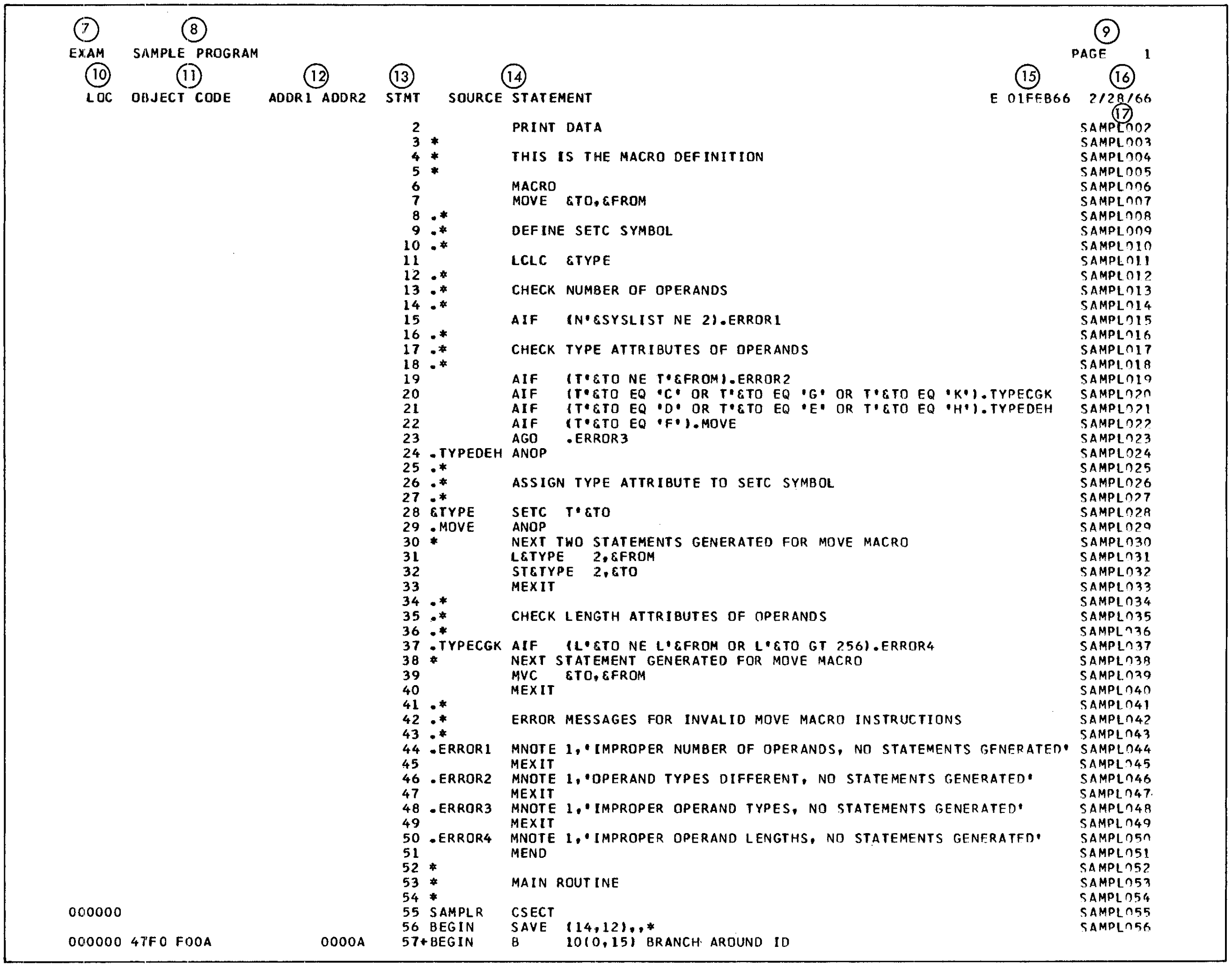
The listing shown in this appendix results from assembling the source program documented in Appendix H of the Assembler Language publication. For easy reference to the explanations that appear in the section "The Assembler Listing," the headings on the listing are numbered.

Since there were no errors in the assembly, a diagnostic list was not produced. Each of the following pages represents one printer-produced listing page.

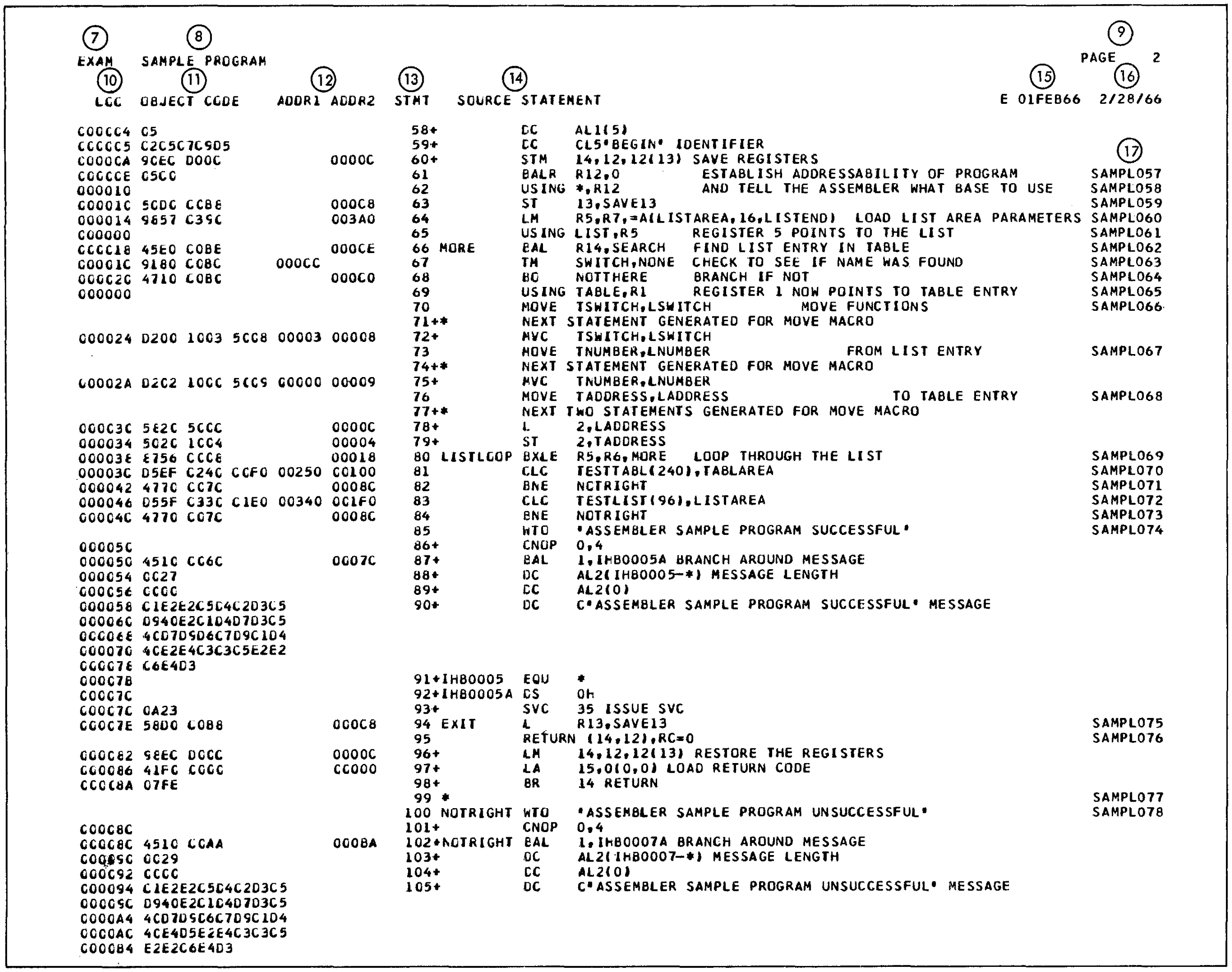


```
    ①       ②   ③    ④      ⑤      ⑥
  SYMBOL   TYPE ID  ADDR  LENGTH LD ID            EXTERNAL SYMBOL DICTIONARY                              PAGE    1

 SAMPLR     SD  01 000000 000388
```

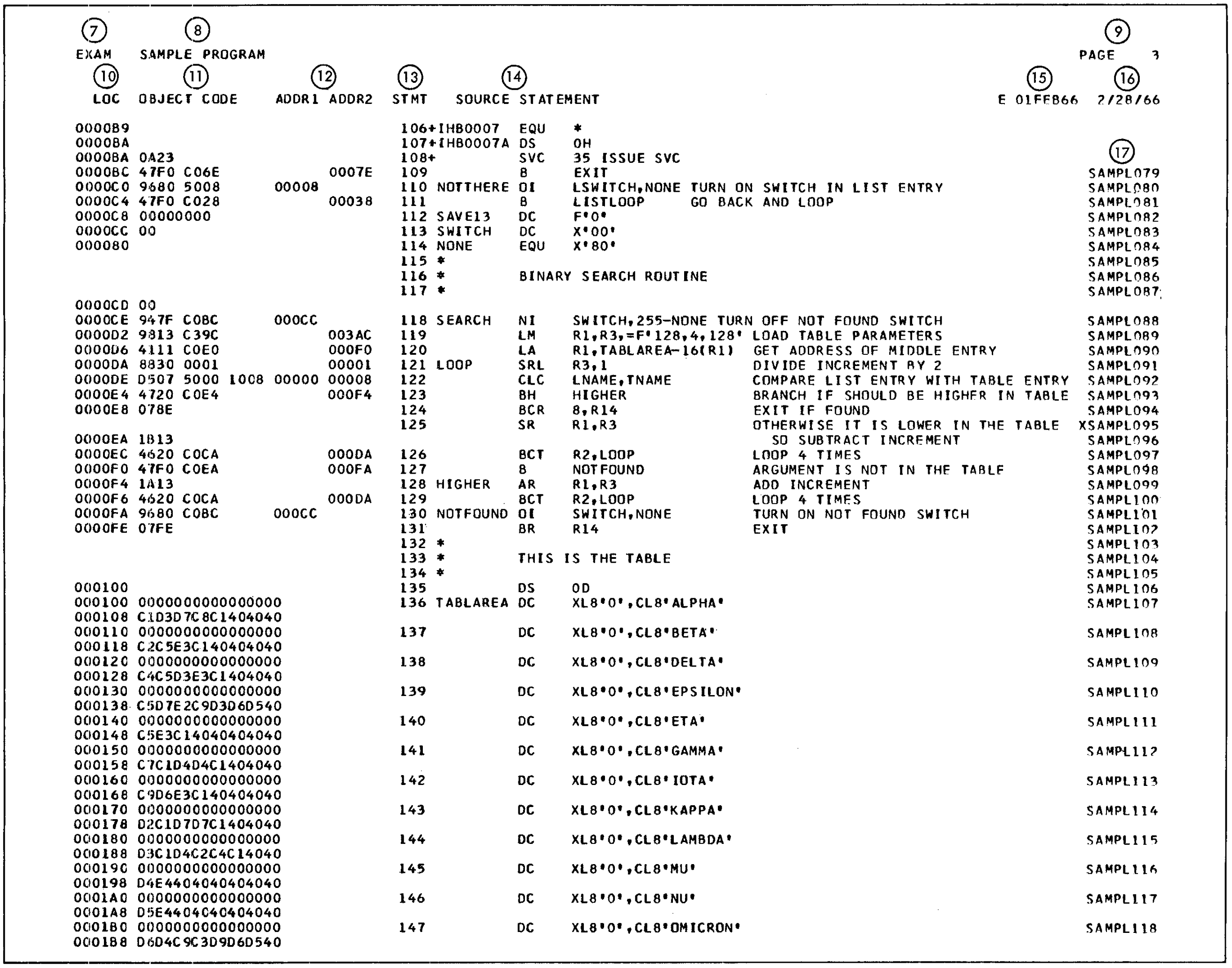

7
8
9
10
11
12
13
14
15
16
17

```
EXAM   SAMPLE PROGRAM                                                                          PAGE    1
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                            E 01FEB66  2/28/66
                                      2          PRINT DATA                                                     SAMPL002
                                      3 *                                                                       SAMPL003
                                      4 *        THIS IS THE MACRO DEFINITION                                   SAMPL004
                                      5 *                                                                       SAMPL005
                                      6          MACRO                                                          SAMPL006
                                      7          MOVE  &TO,&FROM                                                SAMPL007
                                      8 .*                                                                      SAMPL008
                                      9 .*       DEFINE SETC SYMBOL                                             SAMPL009
                                     10 .*                                                                      SAMPL010
                                     11          LCLC  &TYPE                                                    SAMPL011
                                     12 .*                                                                      SAMPL012
                                     13 .*       CHECK NUMBER OF OPERANDS                                       SAMPL013
                                     14 .*                                                                      SAMPL014
                                     15          AIF   (N'&SYSLIST NE 2).ERROR1                                 SAMPL015
                                     16 .*                                                                      SAMPL016
                                     17 .*       CHECK TYPE ATTRIBUTES OF OPERANDS                              SAMPL017
                                     18 .*                                                                      SAMPL018
                                     19          AIF   (T'&TO NE T'&FROM).ERROR2                                SAMPL019
                                     20          AIF   (T'&TO EQ 'C' OR T'&TO EQ 'G' OR T'&TO EQ 'K').TYPECGK   SAMPL020
                                     21          AIF   (T'&TO EQ 'D' OR T'&TO EQ 'E' OR T'&TO EQ 'H').TYPEDEH   SAMPL021
                                     22          AIF   (T'&TO EQ 'F').MOVE                                      SAMPL022
                                     23          AGO   .ERROR3                                                  SAMPL023
                                     24 .TYPEDEH ANOP                                                           SAMPL024
                                     25 .*                                                                      SAMPL025
                                     26 .*       ASSIGN TYPE ATTRIBUTE TO SETC SYMBOL                           SAMPL026
                                     27 .*                                                                      SAMPL027
                                     28 &TYPE    SETC  T'&TO                                                    SAMPL028
                                     29 .MOVE    ANOP                                                           SAMPL029
                                     30 *        NEXT TWO STATEMENTS GENERATED FOR MOVE MACRO                   SAMPL030
                                     31          L&TYPE   2,&FROM                                               SAMPL031
                                     32          ST&TYPE  2,&TO                                                 SAMPL032
                                     33          MEXIT                                                          SAMPL033
                                     34 .*                                                                      SAMPL034
                                     35 .*       CHECK LENGTH ATTRIBUTES OF OPERANDS                            SAMPL035
                                     36 .*                                                                      SAMPL036
                                     37 .TYPECGK AIF   (L'&TO NE L'&FROM OR L'&TO GT 256).ERROR4                SAMPL037
                                     38 *        NEXT STATEMENT GENERATED FOR MOVE MACRO                        SAMPL038
                                     39          MVC   &TO,&FROM                                                SAMPL039
                                     40          MEXIT                                                          SAMPL040
                                     41 .*                                                                      SAMPL041
                                     42 .*       ERROR MESSAGES FOR INVALID MOVE MACRO INSTRUCTIONS             SAMPL042
                                     43 .*                                                                      SAMPL043
                                     44 .ERROR1  MNOTE 1,'IMPROPER NUMBER OF OPERANDS, NO STATEMENTS GENERATED' SAMPL044
                                     45          MEXIT                                                          SAMPL045
                                     46 .ERROR2  MNOTE 1,'OPERAND TYPES DIFFERENT, NO STATEMENTS GENERATED'     SAMPL046
                                     47          MEXIT                                                          SAMPL047
                                     48 .ERROR3  MNOTE 1,'IMPROPER OPERAND TYPES, NO STATEMENTS GENERATED'      SAMPL048
                                     49          MEXIT                                                          SAMPL049
                                     50 .ERROR4  MNOTE 1,'IMPROPER OPERAND LENGTHS, NO STATEMENTS GENERATED'    SAMPL050
                                     51          MEND                                                           SAMPL051
                                     52 *                                                                       SAMPL052
                                     53 *        MAIN ROUTINE                                                   SAMPL053
                                     54 *                                                                       SAMPL054
000000                               55 SAMPLR   CSECT                                                          SAMPL055
                                     56 BEGIN    SAVE  (14,12),,*                                               SAMPL056
000000 47F0 F00A            0000A    57+BEGIN    B     10(0,15) BRANCH AROUND ID
```


7
8
9
10
11
12
13
14
15
16
17

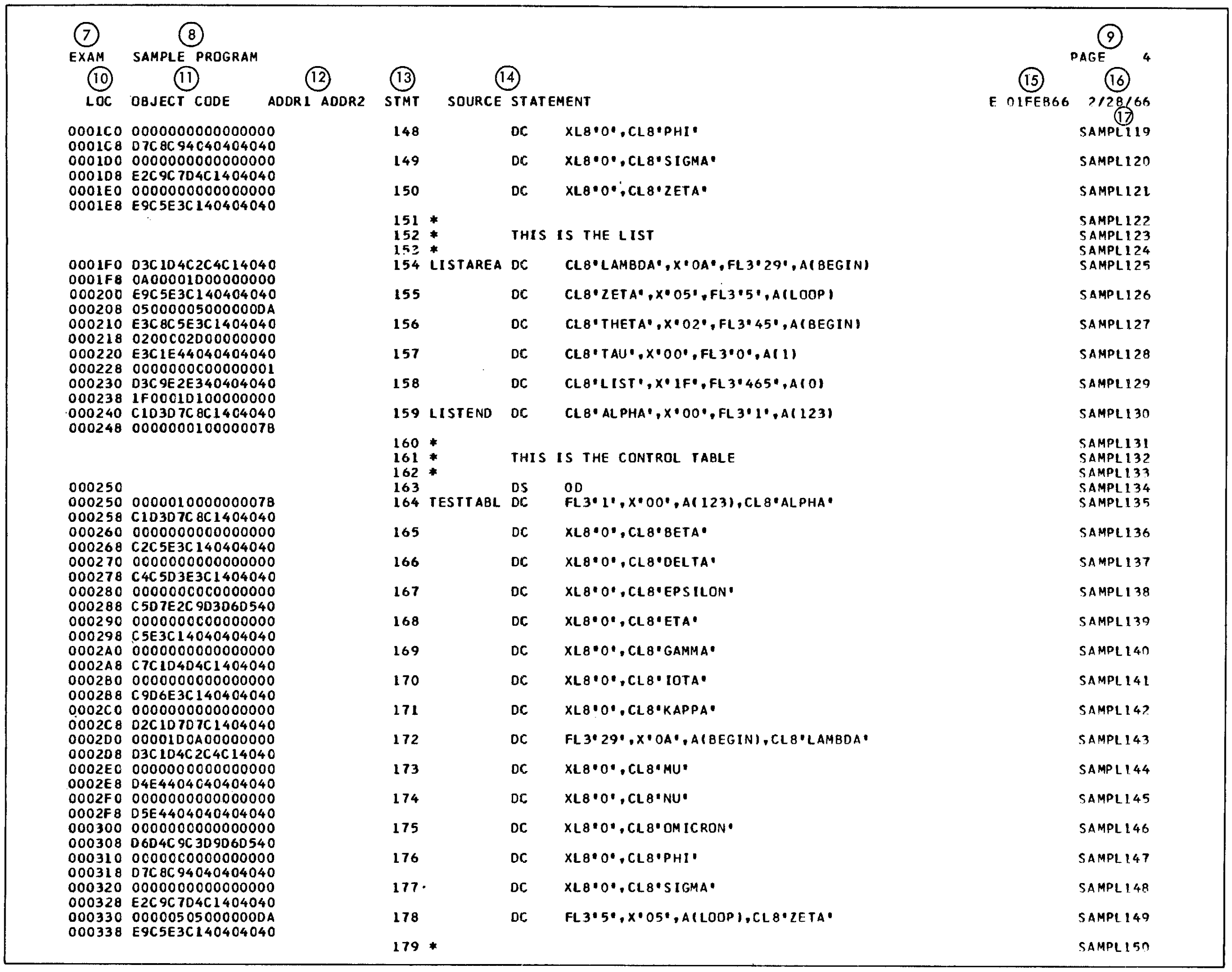
```
EXAM    SAMPLE PROGRAM                                                                                PAGE    2
  LCC   OBJECT CODE      ADDR1 ADDR2  STMT    SOURCE STATEMENT                               E 01FEB66  2/28/66

C000C4 C5                             58+          DC     AL1(5)
CCCCC5 C2C5C7C9D5                     59+          DC     CL5'BEGIN' IDENTIFIER
C000CA 9CEC D00C             0000C    60+          STM    14,12,12(13) SAVE REGISTERS
CCCCCE C5CC                           61           BALR   R12,0          ESTABLISH ADDRESSABILITY OF PROGRAM      SAMPL057
000010                                62           USING *,R12           AND TELL THE ASSEMBLER WHAT BASE TO USE  SAMPL058
C0001C 5CDC CCB8             000C8    63           ST     13,SAVE13                                                SAMPL059
000014 9657 C39C             003A0    64           LM     R5,R7,=A(LISTAREA,16,LISTEND)  LOAD LIST AREA PARAMETERS SAMPL060
C00000                                65           USING LIST,R5       REGISTER 5 POINTS TO THE LIST               SAMPL061
CCCC18 45E0 C0BE             000CE    66 MORE      BAL    R14,SEARCH   FIND LIST ENTRY IN TABLE                    SAMPL062
C0001C 9180 C0BC       000CC          67           TM     SWITCH,NONE  CHECK TO SEE IF NAME WAS FOUND              SAMPL063
000C20 4710 C0BC             000C0    68           BO     NOTTHERE     BRANCH IF NOT                               SAMPL064
000000                                69           USING TABLE,R1      REGISTER 1 NOW POINTS TO TABLE ENTRY        SAMPL065
                                      70           MOVE   TSWITCH,LSWITCH           MOVE FUNCTIONS                     SAMPL066
                                      71+*         NEXT STATEMENT GENERATED FOR MOVE MACRO
000024 D200 1003 5C08 00003 00008     72+          MVC    TSWITCH,LSWITCH
                                      73           MOVE   TNUMBER,LNUMBER                FROM LIST ENTRY               SAMPL067
                                      74+*         NEXT STATEMENT GENERATED FOR MOVE MACRO
C0002A D2C2 10CC 5CC9 C0000 00009     75+          MVC    TNUMBER,LNUMBER
                                      76           MOVE   TADDRESS,LADDRESS                   TO TABLE ENTRY           SAMPL068
                                      77+*         NEXT TWO STATEMENTS GENERATED FOR MOVE MACRO
000C3C 5E2C 5CCC             0000C    78+          L      2,LADDRESS
000034 502C 1CC4             00004    79+          ST     2,TADDRESS
00003E E756 CCC8             00018    80 LISTLCOP  BXLE   R5,R6,MORE   LOOP THROUGH THE LIST                       SAMPL069
00003C D5EF C24C CCF0 00250 C0100     81           CLC    TESTTABL(240),TABLAREA                                   SAMPL070
000042 477C CC7C             0008C    82           BNE    NCTRIGHT                                                 SAMPL071
000046 D55F C33C C1E0 00340 0C1F0     83           CLC    TESTLIST(96),LISTAREA                                    SAMPL072
C0004C 4770 C07C             0008C    84           BNE    NOTRIGHT                                                 SAMPL073
                                      85           WTO    'ASSEMBLER SAMPLE PROGRAM SUCCESSFUL'                    SAMPL074
00005C                                86+          CNOP   0,4
000050 4510 CC6C             0G07C    87+          BAL    1,IHB0005A BRANCH AROUND MESSAGE
000054 CC27                           88+          DC     AL2(IHB0005-*) MESSAGE LENGTH
C000C56 CCCC                          89+          CC     AL2(0)
000058 C1E2E2C5D4C2D3C5               90+          DC     C'ASSEMBLER SAMPLE PROGRAM SUCCESSFUL' MESSAGE
00006C D940E2C1D4D7D3C5
0CC06E 4CD7D9D6C7D9C1D4
000070 4CE2E4C3C3C5E2E2
CGC07E C6E4D3
00007B                                91+IHB0005   EQU    *
C00C7C                                92+IHB0005A  CS     0H
C00C7C 0A23                           93+          SVC    35 ISSUE SVC
CG0C7E 58D0 C0B8             000C8    94 EXIT      L      R13,SAVE13                                               SAMPL075
                                      95           RETURN (14,12),RC=0                                             SAMPL076
CG0C82 98EC D0CC             0000C    96+          LM     14,12,12(13) RESTORE THE REGISTERS
CG0086 41FC C0CC             CC000    97+          LA     15,0(0,0) LOAD RETURN CODE
CCCC8A 07FE                           98+          BR     14 RETURN
                                      99 *                                                                         SAMPL077
                                     100 NOTRIGHT  WTO    'ASSEMBLER SAMPLE PROGRAM UNSUCCESSFUL'                  SAMPL078
C00C8C                               101+          CNOP   0,4
CCCC8C 4510 CCAA             0008A   102+NOTRIGHT  BAL    1,IHB0007A BRANCH AROUND MESSAGE
C00C9C 0C29                          103+          DC     AL2(IHB0007-*) MESSAGE LENGTH
000C92 CCCC                          104+          CC     AL2(0)
C00094 C1E2E2C5D4C2D3C5              105+          DC     C'ASSEMBLER SAMPLE PROGRAM UNSUCCESSFUL' MESSAGE
0000C9C D940E2C1D4D7D3C5
C000A4 4CD7D9C6C7D9C1D4
C000AC 4CE4D5E2E4C3C3C5
C00084 E2E2C6E4D3
```


7
8
9
10
11
12
13
14
15
16
17

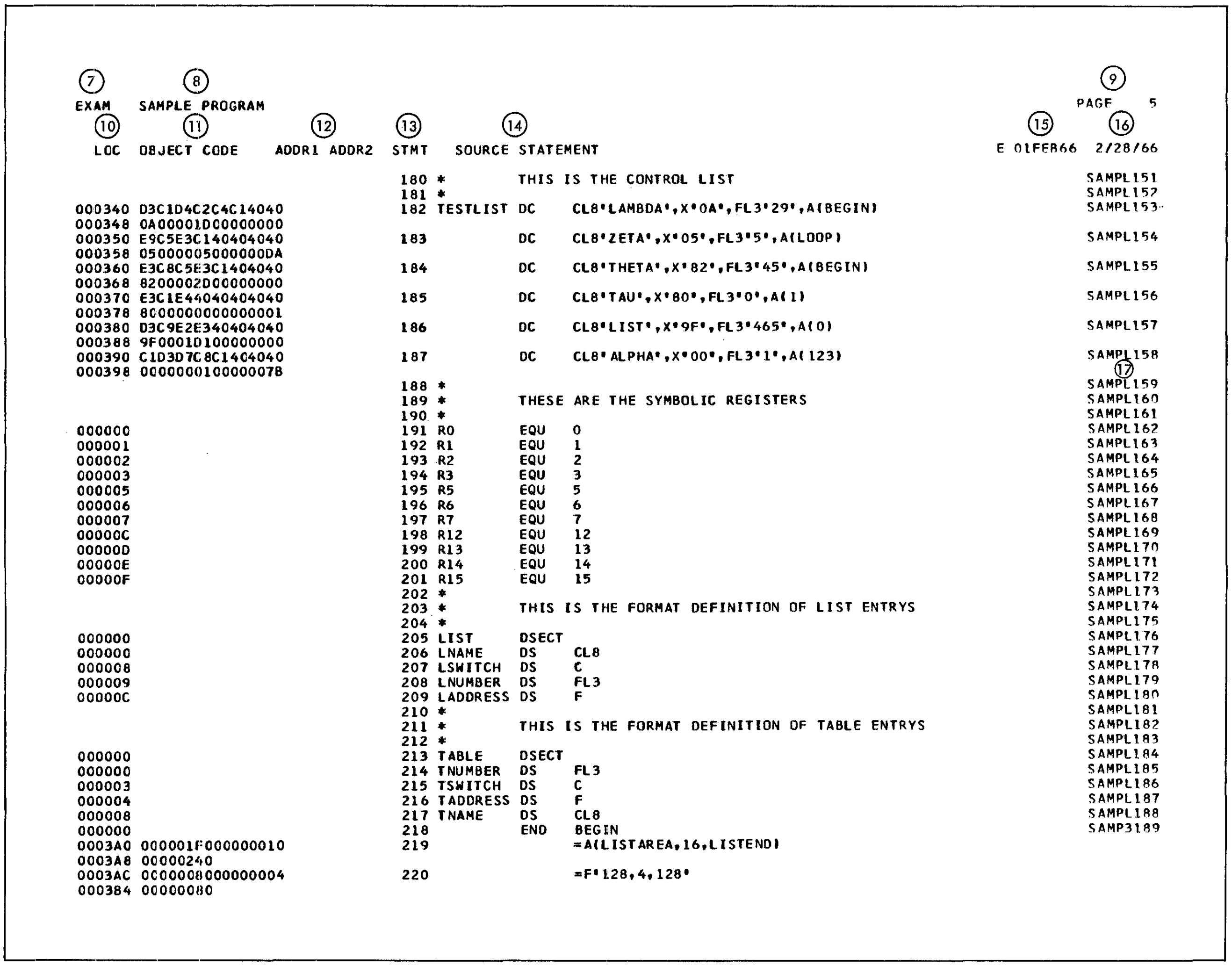
```
EXAM   SAMPLE PROGRAM                                                                                                 PAGE    3
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                                     E 01FEB66  2/28/66

0000B9                             106+IHB0007  EQU   *
0000BA                             107+IHB0007A DS    0H
0000BA 0A23                        108+         SVC   35 ISSUE SVC
0000BC 47F0 C06E            0007E  109          B     EXIT                                                     SAMPL079
0000C0 9680 5008      00008        110 NOTTHERE OI    LSWITCH,NONE TURN ON SWITCH IN LIST ENTRY                SAMPL080
0000C4 47F0 C028            00038  111          B     LISTLOOP      GO BACK AND LOOP                           SAMPL081
0000C8 00000000                    112 SAVE13   DC    F'0'                                                     SAMPL082
0000CC 00                          113 SWITCH   DC    X'00'                                                    SAMPL083
000080                             114 NONE     EQU   X'80'                                                    SAMPL084
                                   115 *                                                                       SAMPL085
                                   116 *        BINARY SEARCH ROUTINE                                          SAMPL086
                                   117 *                                                                       SAMPL087
0000CD 00
0000CE 947F C0BC      000CC        118 SEARCH   NI    SWITCH,255-NONE TURN OFF NOT FOUND SWITCH                SAMPL088
0000D2 9813 C39C            003AC  119          LM    R1,R3,=F'128,4,128' LOAD TABLE PARAMETERS                SAMPL089
0000D6 4111 C0E0            000F0  120          LA    R1,TABLAREA-16(R1)  GET ADDRESS OF MIDDLE ENTRY          SAMPL090
0000DA 8830 0001            00001  121 LOOP     SRL   R3,1                DIVIDE INCREMENT BY 2                SAMPL091
0000DE D507 5000 1008 00000 00008  122          CLC   LNAME,TNAME         COMPARE LIST ENTRY WITH TABLE ENTRY  SAMPL092
0000E4 4720 C0E4            000F4  123          BH    HIGHER              BRANCH IF SHOULD BE HIGHER IN TABLE  SAMPL093
0000E8 078E                        124          BCR   8,R14               EXIT IF FOUND                        SAMPL094
                                   125          SR    R1,R3               OTHERWISE IT IS LOWER IN THE TABLE  XSAMPL095
0000EA 1B13                                                                 SO SUBTRACT INCREMENT              SAMPL096
0000EC 4620 C0CA            000DA  126          BCT   R2,LOOP             LOOP 4 TIMES                         SAMPL097
0000F0 47F0 C0EA            000FA  127          B     NOTFOUND            ARGUMENT IS NOT IN THE TABLE         SAMPL098
0000F4 1A13                        128 HIGHER   AR    R1,R3               ADD INCREMENT                        SAMPL099
0000F6 4620 C0CA            000DA  129          BCT   R2,LOOP             LOOP 4 TIMES                         SAMPL100
0000FA 9680 C0BC      000CC        130 NOTFOUND OI    SWITCH,NONE         TURN ON NOT FOUND SWITCH             SAMPL101
0000FE 07FE                        131          BR    R14                 EXIT                                 SAMPL102
                                   132 *                                                                       SAMPL103
                                   133 *        THIS IS THE TABLE                                              SAMPL104
                                   134 *                                                                       SAMPL105
000100                             135          DS    0D                                                       SAMPL106
000100 0000000000000000            136 TABLAREA DC    XL8'0',CL8'ALPHA'                                        SAMPL107
000108 C1D3D7C8C1404040
000110 0000000000000000            137          DC    XL8'0',CL8'BETA'                                         SAMPL108
000118 C2C5E3C140404040
000120 0000000000000000            138          DC    XL8'0',CL8'DELTA'                                        SAMPL109
000128 C4C5D3E3C1404040
000130 0000000000000000            139          DC    XL8'0',CL8'EPSILON'                                      SAMPL110
000138 C5D7E2C9D3D6D540
000140 0000000000000000            140          DC    XL8'0',CL8'ETA'                                          SAMPL111
000148 C5E3C14040404040
000150 0000000000000000            141          DC    XL8'0',CL8'GAMMA'                                        SAMPL112
000158 C7C1D4D4C1404040
000160 0000000000000000            142          DC    XL8'0',CL8'IOTA'                                         SAMPL113
000168 C9D6E3C140404040
000170 0000000000000000            143          DC    XL8'0',CL8'KAPPA'                                        SAMPL114
000178 D2C1D7D7C1404040
000180 0000000000000000            144          DC    XL8'0',CL8'LAMBDA'                                       SAMPL115
000188 D3C1D4C2C4C14040
000190 0000000000000000            145          DC    XL8'0',CL8'MU'                                           SAMPL116
000198 D4E4404040404040
0001A0 0000000000000000            146          DC    XL8'0',CL8'NU'                                           SAMPL117
0001A8 D5E4404040404040
0001B0 0000000000000000            147          DC    XL8'0',CL8'OMICRON'                                      SAMPL118
0001B8 D6D4C9C3D9D6D540
```


7
8
9
10
11
12
13
14
15
16
17

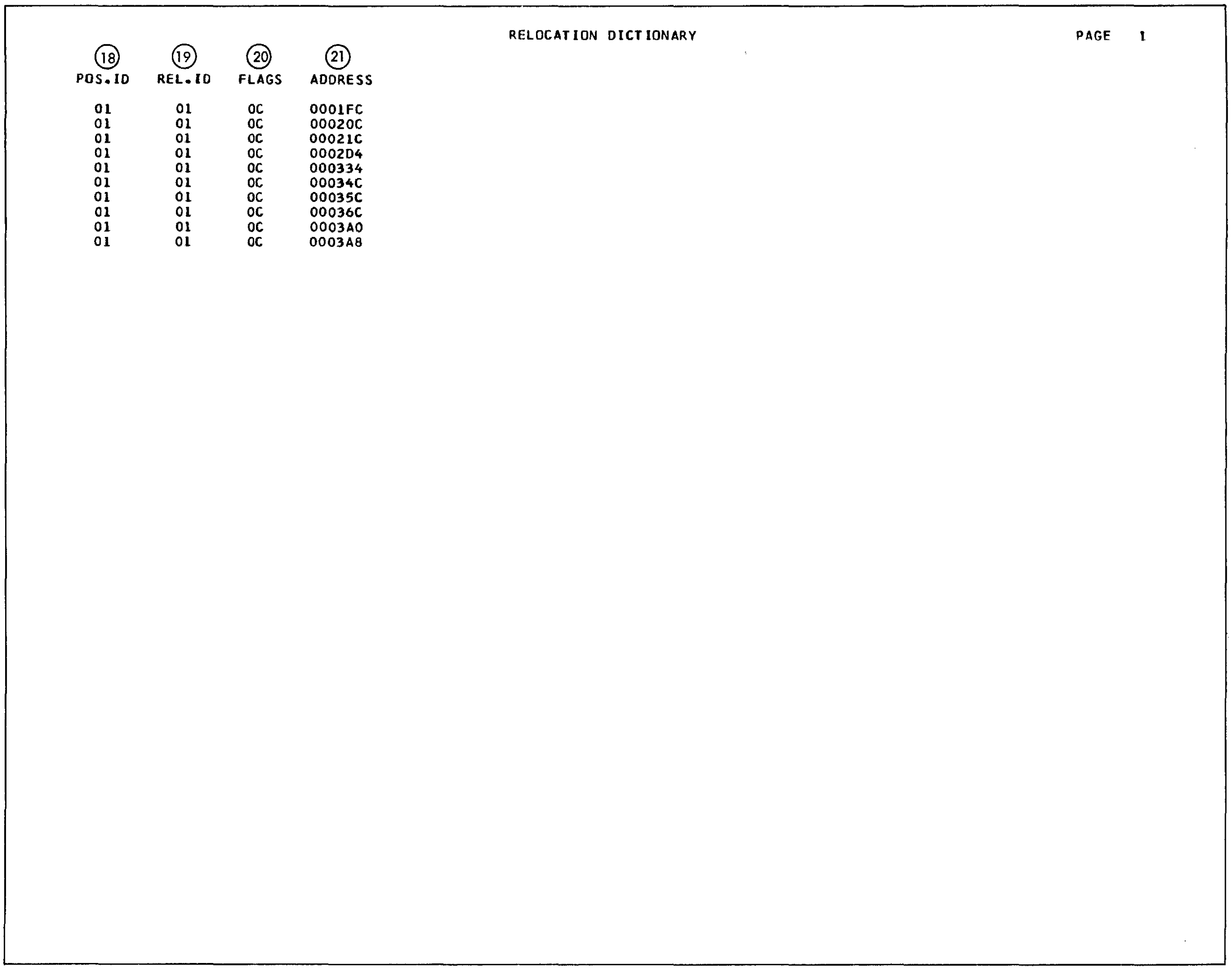
```
EXAM   SAMPLE PROGRAM                                                                                  PAGE    4
  LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT                                E 01FEB66  2/28/66

0001C0 0000000000000000             148           DC    XL8'0',CL8'PHI'                                SAMPL119
0001C8 D7C8C94040404040
0001D0 0000000000000000             149           DC    XL8'0',CL8'SIGMA'                              SAMPL120
0001D8 E2C9C7D4C1404040
0001E0 0000000000000000             150           DC    XL8'0',CL8'ZETA'                               SAMPL121
0001E8 E9C5E3C140404040
                                    151 *                                                          SAMPL122
                                    152 *         THIS IS THE LIST                                 SAMPL123
                                    153 *                                                          SAMPL124
0001F0 D3C1D4C2C4C14040             154 LISTAREA DC    CL8'LAMBDA',X'0A',FL3'29',A(BEGIN)           SAMPL125
0001F8 0A00001D00000000
000200 E9C5E3C140404040             155          DC    CL8'ZETA',X'05',FL3'5',A(LOOP)                SAMPL126
000208 05000005000000DA
000210 E3C8C5E3C1404040             156          DC    CL8'THETA',X'02',FL3'45',A(BEGIN)             SAMPL127
000218 0200C02D00000000
000220 E3C1E44040404040             157          DC    CL8'TAU',X'00',FL3'0',A(1)                    SAMPL128
000228 0000000C00000001
000230 D3C9E2E340404040             158          DC    CL8'LIST',X'1F',FL3'465',A(0)                 SAMPL129
000238 1F0001D100000000
000240 C1D3D7C8C1404040             159 LISTEND  DC    CL8'ALPHA',X'00',FL3'1',A(123)                SAMPL130
000248 000000010000007B
                                    160 *                                                          SAMPL131
                                    161 *         THIS IS THE CONTROL TABLE                        SAMPL132
                                    162 *                                                          SAMPL133
000250                              163          DS    0D                                          SAMPL134
000250 000001000000007B             164 TESTTABL DC    FL3'1',X'00',A(123),CL8'ALPHA'                SAMPL135
000258 C1D3D7C8C1404040
000260 0000000000000000             165          DC    XL8'0',CL8'BETA'                              SAMPL136
000268 C2C5E3C140404040
000270 0000000000000000             166          DC    XL8'0',CL8'DELTA'                             SAMPL137
000278 C4C5D3E3C1404040
000280 000000C0C0000000             167          DC    XL8'0',CL8'EPSILON'                           SAMPL138
000288 C5D7E2C9D3D6D540
000290 0000000C00000000             168          DC    XL8'0',CL8'ETA'                               SAMPL139
000298 C5E3C14040404040
0002A0 0000000000000000             169          DC    XL8'0',CL8'GAMMA'                             SAMPL140
0002A8 C7C1D4D4C1404040
0002B0 0000000000000000             170          DC    XL8'0',CL8'IOTA'                              SAMPL141
0002B8 C9D6E3C140404040
0002C0 0000000000000000             171          DC    XL8'0',CL8'KAPPA'                             SAMPL142
0002C8 D2C1D7D7C1404040
0002D0 00001D0A00000000             172          DC    FL3'29',X'0A',A(BEGIN),CL8'LAMBDA'            SAMPL143
0002D8 D3C1D4C2C4C14040
0002E0 0000000000000000             173          DC    XL8'0',CL8'MU'                                SAMPL144
0002E8 D4E4404040404040
0002F0 0000000000000000             174          DC    XL8'0',CL8'NU'                                SAMPL145
0002F8 D5E4404040404040
000300 0000000000000000             175          DC    XL8'0',CL8'OMICRON'                           SAMPL146
000308 D6D4C9C3D9D6D540
000310 00000C0000000000             176          DC    XL8'0',CL8'PHI'                               SAMPL147
000318 D7C8C94040404040
000320 0000000000000000             177          DC    XL8'0',CL8'SIGMA'                             SAMPL148
000328 E2C9C7D4C1404040
000330 00000505000000DA             178          DC    FL3'5',X'05',A(LOOP),CL8'ZETA'                SAMPL149
000338 E9C5E3C140404040
                                    179 *                                                          SAMPL150
```


7
8
9
10
11
12
13
14
15
16
17

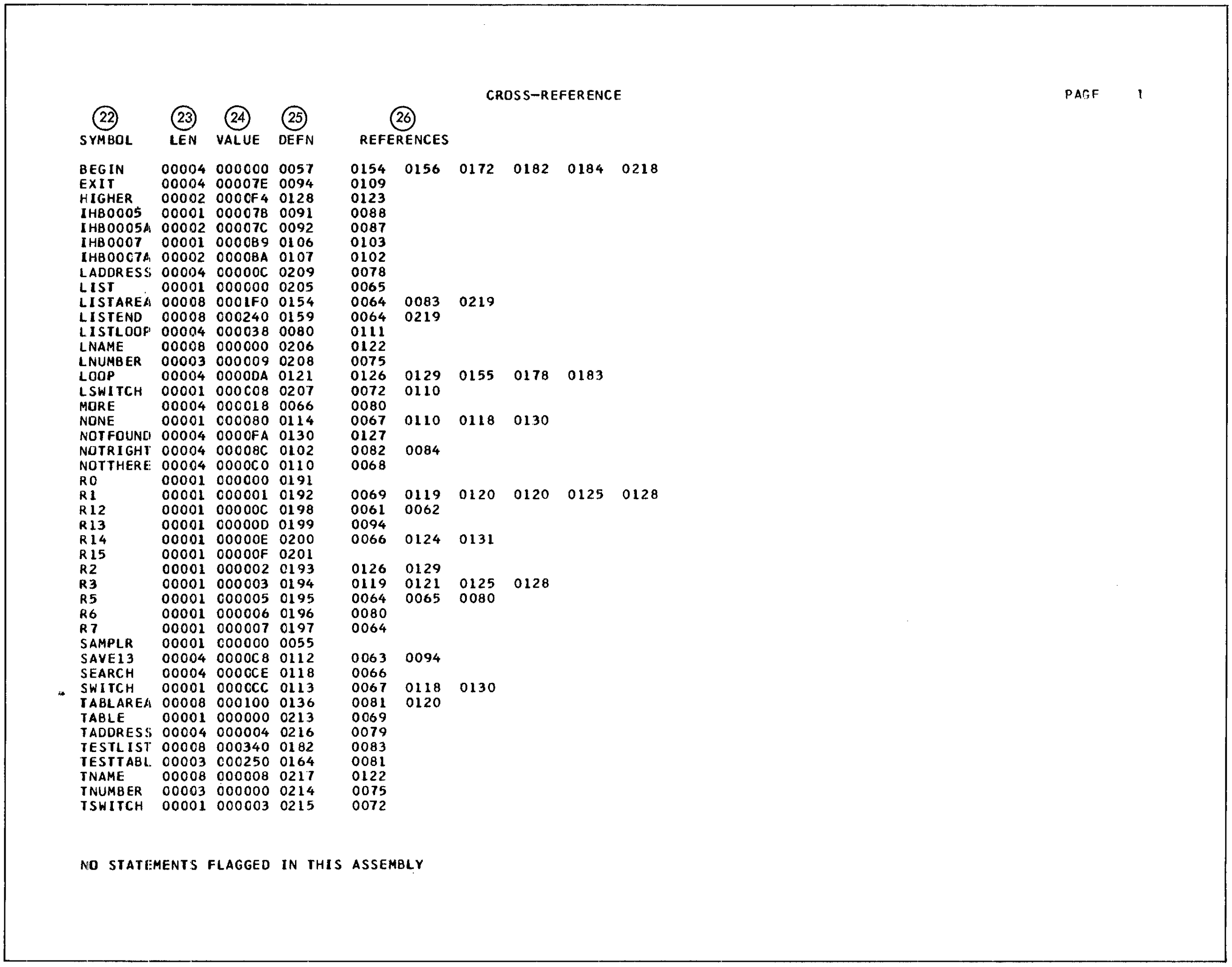
```
EXAM   SAMPLE PROGRAM                                                                                        PAGE    5
  LOC  OBJECT CODE      ADDR1 ADDR2  STMT   SOURCE STATEMENT                                        E 01FEB66  2/28/66

                                    180 *        THIS IS THE CONTROL LIST                                       SAMPL151
                                    181 *                                                                       SAMPL152
000340 D3C1D4C2C4C14040             182 TESTLIST DC    CL8'LAMBDA',X'0A',FL3'29',A(BEGIN)                       SAMPL153
000348 0A00001D00000000
000350 E9C5E3C140404040             183          DC    CL8'ZETA',X'05',FL3'5',A(LOOP)                           SAMPL154
000358 050000050000000DA
000360 E3C8C5E3C1404040             184          DC    CL8'THETA',X'82',FL3'45',A(BEGIN)                        SAMPL155
000368 8200002D00000000
000370 E3C1E44040404040             185          DC    CL8'TAU',X'80',FL3'0',A(1)                               SAMPL156
000378 8000000000000001
000380 D3C9E2E340404040             186          DC    CL8'LIST',X'9F',FL3'465',A(0)                            SAMPL157
000388 9F0001D100000000
000390 C1D3D7C8C1404040             187          DC    CL8'ALPHA',X'00',FL3'1',A(123)                           SAMPL158
000398 000000010000007B
                                    188 *                                                                       SAMPL159
                                    189 *        THESE ARE THE SYMBOLIC REGISTERS                               SAMPL160
                                    190 *                                                                       SAMPL161
000000                              191 R0       EQU   0                                                        SAMPL162
000001                              192 R1       EQU   1                                                        SAMPL163
000002                              193 R2       EQU   2                                                        SAMPL164
000003                              194 R3       EQU   3                                                        SAMPL165
000005                              195 R5       EQU   5                                                        SAMPL166
000006                              196 R6       EQU   6                                                        SAMPL167
000007                              197 R7       EQU   7                                                        SAMPL168
00000C                              198 R12      EQU   12                                                       SAMPL169
00000D                              199 R13      EQU   13                                                       SAMPL170
00000E                              200 R14      EQU   14                                                       SAMPL171
00000F                              201 R15      EQU   15                                                       SAMPL172
                                    202 *                                                                       SAMPL173
                                    203 *        THIS IS THE FORMAT DEFINITION OF LIST ENTRYS                   SAMPL174
                                    204 *                                                                       SAMPL175
000000                              205 LIST     DSECT                                                          SAMPL176
000000                              206 LNAME    DS    CL8                                                      SAMPL177
000008                              207 LSWITCH  DS    C                                                        SAMPL178
000009                              208 LNUMBER  DS    FL3                                                      SAMPL179
00000C                              209 LADDRESS DS    F                                                        SAMPL180
                                    210 *                                                                       SAMPL181
                                    211 *        THIS IS THE FORMAT DEFINITION OF TABLE ENTRYS                  SAMPL182
                                    212 *                                                                       SAMPL183
000000                              213 TABLE    DSECT                                                          SAMPL184
000000                              214 TNUMBER  DS    FL3                                                      SAMPL185
000003                              215 TSWITCH  DS    C                                                        SAMPL186
000004                              216 TADDRESS DS    F                                                        SAMPL187
000008                              217 TNAME    DS    CL8                                                      SAMPL188
000000                              218          END   BEGIN                                                    SAMP3189
0003A0 000001F000000010             219                =A(LISTAREA,16,LISTEND)
0003A8 00000240
0003AC 0000008000000004             220                =F'128,4,128'
0003B4 00000080
```



```
                                              RELOCATION DICTIONARY                                        PAGE   1
   POS.ID     REL.ID    FLAGS    ADDRESS

     01         01        0C     0001FC
     01         01        0C     00020C
     01         01        0C     00021C
     01         01        0C     0002D4
     01         01        0C     000334
     01         01        0C     00034C
     01         01        0C     00035C
     01         01        0C     00036C
     01         01        0C     0003A0
     01         01        0C     0003A8
```



```
                                              CROSS-REFERENCE                                        PAGE    1
  ㉒         ㉓    ㉔     ㉕          ㉖
SYMBOL      LEN   VALUE  DEFN     REFERENCES

BEGIN     00004 000000 0057     0154  0156  0172  0182  0184  0218
EXIT      00004 00007E 0094     0109
HIGHER    00002 0000F4 0128     0123
IHB0005   00001 00007B 0091     0088
IHB0005A  00002 00007C 0092     0087
IHB0007   00001 0000B9 0106     0103
IHB0007A  00002 0000BA 0107     0102
LADDRESS  00004 00000C 0209     0078
LIST      00001 000000 0205     0065
LISTAREA  00008 0001F0 0154     0064  0083  0219
LISTEND   00008 000240 0159     0064  0219
LISTLOOP  00004 000038 0080     0111
LNAME     00008 000000 0206     0122
LNUMBER   00003 000009 0208     0075
LOOP      00004 0000DA 0121     0126  0129  0155  0178  0183
LSWITCH   00001 000C08 0207     0072  0110
MORE      00004 000018 0066     0080
NONE      00001 000080 0114     0067  0110  0118  0130
NOTFOUND  00004 0000FA 0130     0127
NOTRIGHT  00004 00008C 0102     0082  0084
NOTTHERE  00004 0000C0 0110     0068
R0        00001 000000 0191
R1        00001 000001 0192     0069  0119  0120  0120  0125  0128
R12       00001 00000C 0198     0061  0062
R13       00001 00000D 0199     0094
R14       00001 00000E 0200     0066  0124  0131
R15       00001 00000F 0201
R2        00001 000002 0193     0126  0129
R3        00001 000003 0194     0119  0121  0125  0128
R5        00001 000005 0195     0064  0065  0080
R6        00001 000006 0196     0080
R7        00001 000007 0197     0064
SAMPLR    00001 000000 0055
SAVE13    00004 0000C8 0112     0063  0094
SEARCH    00004 0000CE 0118     0066
SWITCH    00001 0000CC 0113     0067  0118  0130
TABLAREA  00008 000100 0136     0081  0120
TABLE     00001 000000 0213     0069
TADDRESS  00004 000004 0216     0079
TESTLIST  00008 000340 0182     0083
TESTTABL  00003 000250 0164     0081
TNAME     00008 000008 0217     0122
TNUMBER   00003 000000 0214     0075
TSWITCH   00001 000003 0215     0072


NO STATEMENTS FLAGGED IN THIS ASSEMBLY
```

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

READER'S COMMENT FORM

IBM System/360 Operating System
Assembler (E) Programmer's Guide

Form C28-6595-1

- Your comments, accompanied by answers to the following questions, help us produce better publications for your use. If your answer to a question is "No" or requires qualification, please explain in the space provided below. Comments and suggestions become the property of IBM.

	Yes	No
• Does this publication meet your needs?	□	□
• Did you find the material:		
Easy to read and understand?	□	□
Organized for convenient use?	□	□
Complete?	□	□
Well illustrated?	□	□
Written for your technical level?	□	□

- What is your occupation? ______________________________
- How do you use this publication?

  As an introduction to the subject? □

  For advanced knowledge of the subject? □

  For information about operating procedures? □

  As an instructor in a class? □

  As a student in a class? □

  As a reference manual? □

  Other ______________________________

- Please give specific page and line references with your comments when appropriate. If you wish a reply, be sure to include your name and address.

COMMENTS

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

C28-6595-1

fold fold

FIRST CLASS
PERMIT NO. 2078
SAN JOSE, CALIF.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

POSTAGE WILL BE PAID BY . . .

IBM Corporation
Monterey & Cottle Rds.
San Jose, California
95114

Attention: Programming Publications, Dept. 452

fold fold

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]