ISIS-II
8080/8085 MACRO ASSEMBLER
OPERATOR'S MANUAL

Manual Order Number: 9800292-04 Rev. D

Intel Corporation, 3065 Bowers Avenue, Santa Clara, California 95051

Additional copies of this manual or other Intel literature may be obtained from:

Literature Department
Intel Corporation
3065 Bowers Avenue
Santa Clara, CA 95051

PREFACE

This manual describes operating procedures for the ISIS-II 8080/8085 Macro Assembler. The assembler translates 8080/8085 assembly language source code into object code executable on the 8080/8085 microprocessors. No discussion of assembly language or assembler directives is provided here, as these topics are covered in the document

8080/8085 Assembly Language Programming Manual 9800301

The ISIS-II assembler is invoked using the diskette operating system facility of Intel's Intellec Microcomputer Development System. Use of this facility is described in the document

ISIS-II System User's Guide 9800306

CONTENTS

CHAPTER 1
ASSEMBLER OVERVIEW

An assembler performs the clerical function of converting your assembly language program into machine-executable form. It accepts your source file and, depending on the output options selected, can produce an executable object file, a listing of the source and assembled code, and a symbol cross-reference listing.

The ISIS-II 8080/8085 Macro Assembler (ASM80) runs under the Intel Systems Implementation Supervisor (ISIS-II) and resides on the ISIS-II system diskette. Operation of the ISIS-II assember is the subject of this manual.

Details for loading and controlling the ISIS-II assembler are given in Chapters 2 and 3. List formats are shown in Chapter 4. Error messages issued by the assembler are listed in Chapter 7. Chapter 5 describes the linkage conventions for passing parameters from an assembly language program module to a PL/M module. The hardware/software environment requirements are summarized below.

ISIS-II Assembler Environment

The ISIS-II assembler uses the following hardware:

- Intellec system with 32K RAM memory (48K if source contains macros)
- Console device (TTY or CRT)
- Diskette unit
- Line printer (if available)

This assembler runs under ISIS-II, which includes the Locater, Linker, and Library Manager.

Overlay and Nonoverlay Operating Modes

The ISIS-II assembler can be run in either an overlay or nonoverlay mode. Overlay mode requires only 32K of Intellec memory and is selected by specifying the NOMACROFILE control (described in Chapter 2). This control can be specified only if your program does not include macros.

If your program has macros, or if you want faster assemblies, specify the MACROFILE control. Your Intellec system must have a 48K memory in this case, but the assembler can run in nonoverlay mode. Specifying MACROFILE also adds the macro reserved symbols (MACRO, ENDM, LOCAL, REPT, IRP, IRPC, EXITM) to the list of reserved words.

No matter how much Intellec memory you have, the assembler runs in overlay (slower) mode unless you specify MACROFILE. (NOMACROFILE is the default control.) Both modes use an optional overlay to create cross-reference listings.

Symbol Table Size

In overlay mode with 32K of Intellec memory, you can generate slightly more than 200 symbols. In nonoverlay mode with 48K of memory, you can have slightly more than 800 symbols. Each 16K memory increment adds 2000 symbols.

Macro processing uses a small amount of memory (for storing actual parameters) that could otherwise be used for symbols. The size or number of macro definitions does not affect symbol table space, however, since macro definitions are stored in a diskette file.

If you need more symbol table space, you can either add more Intellec memory or divide your program into smaller modules.

Input/Output Files

Source File

The input to the ISIS-II assembler is a source file, which can contain three elements:

- An 8080/8085 assembly language program, composed of instructions described in the *8080/8085 Assembly Language Programming Manual* (9800301);
- Assembler directives, described in the same manual;
- Assembler control lines, described in the next chapter of this manual.

Only the instructions in your assembly language program are translated into executable object code.

Object File

The ISIS-II assembler creates its object file on a flexible diskette.

The object file contains machine language instructions and data that can be loaded into memory for execution or interpretation. In addition, it contains control information governing the loading process (such as the starting address for program execution).

The assembler can produce object files in relocatable object code format. The object modules produced by the ISIS-II 8080/8085 assembler can be loaded and executed anywhere in memory if they don't interfere with other programs they need for proper operation. The modules can also be linked to form a larger program. See the *ISIS-II Systems User's Guide* (9800306) for details.

List File

The list file is a formatted file designed to be output to a line printer or terminal. It includes listings of:

- Your assembled object code;
- Your source program;
- A table of symbols and their values;
- A summary of assembly errors.

The formats of these listings are described in Chapter 4.

Symbol-Cross-Reference File

During the first pass of the assembler, a file of symbol-cross-reference records is created, if requested. This diskette file is named ASXREF.TMP.

In general, the assembler generates two types of symbol-cross-reference records: *symbol-definition records* and *symbol-reference records.* If a symbol appears as a name in a label field and the symbol is being defined (by SET, EQU, MACRO, or as a label), a symbol-definition record is produced. If the symbol is being redefined by SET or MACRO, it is considered a symbol definition. All other symbol occurrences are considered references and cause the assembler to generate a symbol-reference record each time the symbol appears. Symbol definition records are terminated by a pound sign (#) in the cross-reference listing.

All symbols are cross referenced except dummy parameters and local labels appearing in macro definitions (that is, all global user-defined symbols, macro names, and actual symbols replacing dummy parameters or local labels are cross referenced). When certain controls that suppress source listing are specified (NOLIST, NOGEN, NOCOND), symbols not listed are not cross referenced.

The assembler calls on ISIS-II to load its cross-reference-generator program and ASXREF.TMP from the diskette. From the programmer's point of view, the operations required to produce the cross-reference listing are automatic (once the cross reference file has been requested). The format of the cross-reference listing is shown in Chapter 4.

Assembler Files

The ISIS-II assembler uses several files of its own, such as the intermediate cross-reference file just mentioned. While you don't need to remember the names of these files, you must know where they reside to avoid diskette space conflict.

The assembler root program (ASM80) and its overlays (ASM80.0Vn, where n=0,1,2...) must reside on the same diskette, but this diskette can be on any drive. The cross-reference generator (ASXREF) must reside on this diskette also.

The intermediate cross-reference file (ASXREF.TMP) is written to the drive containing your source file. The MACROFILE control determines where the intermediate macro file (ASMAC.TMP) is written; the default is the source file drive.

The list and object files can be directed to any drive via assembler controls. The default sends these files to the source file drive.

CHAPTER 2
ASSEMBLER CONTROLS

Introduction to Assembler Controls

Assembler controls allow you to specify the input/output files or devices to be used by the assembler and whether list or object files (or portions of these files) are to be generated by the assembler. Controls can be specified at two levels:

- In commands specified at assembly time
- As control lines embedded throughout your source file

The latter allows selective control over sections of your program. For example, you might want to suppress the assembly listing for certain sections of your program, or to cause page ejects at specific places.

Primary and General Controls

Controls are classified as primary and general. Both classes of controls can be set when the assembler is run or in source file control lines. However, source file control lines containing primary controls must be inserted before the first line of comments or source code. General controls can be respecified at any time.

The ISIS-II assembler allows primary controls to be specified only once. This applies to controls specified in assembly-time command lines, to control lines embedded in your source code, or combinations of the two.

Specifying Controls

Controls can be specified using either upper-case or lower-case characters.

If a control is specified incorrectly in an assembly-time command, the entire command is ignored and must be reentered.

If a control is specified incorrectly in a source code control line, the incorrect control and all controls following it in the line are ignored.

Summary of Controls

The following list shows the controls available, their basic functions, and whether they are primary or general (P/G). Default controls are *italicized*. The remainder of this chapter describes each control in greater detail.

Control	P/G	Function Area
OBJECT/NOOBJECT	P	Object File
DEBUG/*NODEBUG*	P	Object File
PRINT/NOPRINT	P	Assembly Listing
COND/NOCOND	G	Assembly Listing
LIST/NOLIST	G	Assembly Listing
SYMBOLS/NOSYMBOLS	P	Assembly Listing
XREF/*NOXREF*	P	Cross-Reference Listing
PAGING/NOPAGING	P	Listing Format

Control	P/G	Function Area
PAGELENGTH (*66*)	P	Listing Format
PAGEWIDTH (*120*)	P	Listing Format
TITLE	G	Listing Format
EJECT	G	Listing Format
GEN/NOGEN	G	Macro List
MACRODEBUG/*NOMACRODEBUG*	P	Macro List
MACROFILE/*NOMACROFILE*	P	Macro Temporary File
MOD85	P	8085 Switch
SAVE	G	Stack Controls
RESTORE	G	Fetch Controls
INCLUDE	G	Library Function
TTY/*NOTTY*	P	Teletypewriter Function

In general, controls go into effect at the end of the control line. Exceptions occur for control lines that cause a transition between 'listing' and 'not listing' states.

If the transition is caused by a change in the LIST/NOLIST controls, the control line containing the LIST/NOLIST is always printed. If a change in GEN/NOGEN causes the transition, the control line is not printed. If the control line contains both LIST/NOLIST and GEN/NOGEN changes, the control line is printed.

ISIS-II Assembler Controls

ISIS-II Assembly-Time Command

The ISIS-II Assembler for the 8080 or 8085 is invoked by calling the ISIS-II file ASM80. This call includes the name of your source file and any assembler controls you wish to specify. Items in the control list are separated by spaces. The call is terminated by a carriage return.

```
-ASM80 file control-list
```

The "file" in this format is your source file. This file (and files enclosed in parentheses as part of a control) can be a 1-6 character file name, a file name followed by a period and 1-3 character extension, an ISIS-II device name, or an ISIS-II device name followed by a file name and extension. (See the *ISIS-II System User's Guide* for details.)

Examples:

```
FILE20          (filename)
PROG.SRC        (filename.extension)
:HR:            (:ISIS-II device name:)
:F1:ASSMB.SRC   (:ISIS-II dev name:filename.ext)
```

All control items specified must be spelled out in their entirety.

Example:

```
-ASM80 PROG.SRC DEBUG SYMBOLS XREF
```

Primary Controls

Control	Effect
OBJECT(file)	An object code file is generated and is output to the specified diskette file. If this control is omitted, "OBJECT(file.OBJ)" is assumed, where "file" is the name of your source file; the file is directed to the same drive used for the source file.
NOOBJECT	Object code generation is suppressed.
MOD85	The assembler assumes that 8080 code is being assembled unless the 8085 instruction set is specified by this control. This assembler issues a warning if an 8085 instruction is encountered in the 8080 mode.
DEBUG	If an object file is requested, the symbol table is output to that file. DEBUG has no effect otherwise.
NODEBUG	The symbol table is not included in the object file.
PRINT(file)	An assembly list file is generated and is output to the specified file. If this control is omitted, "PRINT(file.LST)" is assumed, where "file" is the name of your source code file; the file is directed to the name drive used for the source file. See general control LIST.
NOPRINT	The assembly output listing is suppressed. No file is specified for listing; therefore, no listing output is possible.
SYMBOLS	If a list file is opened by PRINT, the symbol table is output to the list file. SYMBOLS has no effect otherwise.
NOSYMBOLS	The symbol table is not included in the list file created by PRINT.
XREF	A symbol-cross-reference file is requested. An intermediate file is output to ASXREF.TMP and the cross-reference listing to the file created by PRINT.
NOXREF	Symbol-cross-reference file generation is suppressed.
MACROFILE(drive)	The macro definition file is directed to the specified drive. If no drive is specified, the drive where the source file resides is used. Your Intellec system must have a 48K memory if MACROFILE is specified as the assembler runs in nonoverlay mode. MACROFILE also adds the macro reserved words (MACRO, LOCAL, etc.) to the reserved word list.
NOMACROFILE	No macro temporary files are created. If your source file contains macros, all definitions and calls cause errors. This control lets the assembler run on a 32K-memory Intellec system in overlay mode.
PAGELENGTH(n)	Each list file page is "n" lines long, where "n" must be at least 12 and includes 3 blank lines at the top of the page, 3 blank lines at the bottom of the page, and any page headings specified. If "n" is ≤11, PAGELENGTH is set to 12. The default value is 66. Note that 3 blank lines are issued to reach the next "top-of- page" as opposed to issuing form feeds to reach the physical "top-of-form."
PAGEWIDTH(n)	Each list file line can be up to "n" characters long, where "n" must be in the range 72≤n≤132. Lines exceeding the page width are continued in column 25 of the following line (but lines >132 characters are truncated to 132). The default page width is 120.

Control	Effect
PAGING	The assembler separates listing into pages with headers at each page break.
NOPAGING	The listing is not separated into pages. Headers are printed only once at the beginning of the listing.
MACRODEBUG	Assembler-generated macro symbols are output to the list and object files when the symbol table is output.
NOMACRODEBUG	Assembler-generated macro symbols are not output to the list and object files.
TTY	Simulates form-feed for teletypewriter output.
NOTTY	No teletypewriter output (form-feed simulation).

General Controls

Control	Effect
INCLUDE(file)	Subsequent source lines are input from a specified file until an end-of-file or nested INCLUDE is found. (Nesting may be four deep.) Following the end-of-file, input resumes from the file being processed when the INCLUDE was encountered. The primary use of INCLUDE is to provide a macro library capability. All include lines have an "=" printed in column 19. After the initial INCLUDE, any nested INCLUDEs have their nesting level (1-4) printed in column 18.
LIST	An assembly output listing is generated and sent to the file specified by the PRINT control.
NOLIST	Assembly listing is suppressed, except lines containing errors.
COND	Conditionally-skipped source code is included in the assembly listing if LIST is selected. The conditional-assembly directives are also listed.
NOCOND	Listing of conditionally-skipped source code and conditional-assembly directives is suppressed. Listing of the EXITM directive is suppressed also.
GEN	Macro expansion source text generated by macro calls is listed if LIST is selected.
NOGEN	Macro expansion source text listing is suppressed.
TITLE('string')	The specified string is printed in character positions 1-64 of the second line of page headings. Strings longer than 64 characters are truncated. 'String' must not be null. TITLE remains in effect until another TITLE is encountered. The assembler inserts a blank line if TITLE is not specified.
EJECT	Spaces are skipped to the next top-of-form. The position of the next top-of-form is determined by PAGELENGTH, not by the physical top-of-form.
SAVE	The current settings of the LIST, COND, and GEN controls are stacked (but remain valid until explicitly changed). These controls can be stacked up to eight levels deep.
RESTORE	The LIST, COND, and GEN control settings at the top of the stack are restored.

Defaults

The following defaults are assumed by the ISIS-II assembler if the corresponding controls are not selected:

```
OBJECT(file.OBJ)
NODEBUG
PRINT(file.LST)
LIST
SYMBOLS
COND
GEN
NOXREF
NOMACRODEBUG
NOMACROFILE
PAGING
PAGELENGTH(66)
PAGEWIDTH(120)
```

ISIS-II Assembler Control Lines

The format for control lines embedded in source files to be processed by the ISIS-II assembler is

```
$control list
```

where "$" must appear in column 1 and items in the control list are separated by spaces.

Example:

```
$LIST DEBUG XREF MOD85
```

Control lines containing primary controls must appear before the first statement in the source file, including comments. General control lines can be interspersed throughout the source file.

A control line containing more than one control is scanned from left to right. If a control is specified incorrectly, it is ignored. All remaining controls on that line are also ignored.

CHAPTER 3
ASSEMBLER OPERATION

The ISIS-II 8080/8085 Assembler resides on the ISIS-II system diskette. The assembler is loaded by calling ASM80 at the ISIS-II command level and specifying your source file along with any desired assembler controls (Chapter 2). All requested assembler operations are performed without further intervention once the assembly begins.

Activation Sequence

The following command activates and completes an ISIS-II assembly:

```
-ASM80 PROG.SRC SYMBOLS NODEBUG
```

Following the ISIS-II command prompt (-), you issue a command to assemble the file PROG.SRC. Be default, an assembly listing and object code file are requested and are output to PROG.LST and PROG.OBJ respectively. In addition, a symbol table listing is performed, but symbol table output to the object file is suppressed. Note that the same effect can be achieved with no control specification, since the controls specified are both defaults.

The assembler sends out its sign-on message to the console device:

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V2.0
```

After execution, the assembler issues a sign-off message and error summary:

```
ASSEMBLY COMPLETE, NO ERRORS
```

If XREF is selected, the sign-on message

```
ISIS-II ASSEMBLER SYMBOL CROSS REFERENCE, V2.0
```

is then issued on the console.

Sample Assembly

The following example illustrates a typical use of the ISIS-II assembler. A short program (MBADD.SRC) is taken through all the steps needed to activate the assembler, obtain an object code file, and generate a symbol-cross-reference listing. The source program to be assembled is shown first, followed by the assembler activation sequence. The resulting assembly and symbol-cross-reference listings are also shown.

The source code for program MBADD.SRC is as follows:

```
$       TITLE('MULTIBYTE ADDITION PROGRAM')
        NAME   MBADD
LDAD    MACRO  OPD1,OPD2,COUNT
L1:     LXI    B,OPD1   ;;ADDRESS OF 1ST NO. TO BC
        LXI    H,OPD2   ;;ADDRESS OF 2ND NO. TO HL
        LXI    D,COUNT  ;;LOOP CONTROL COUNT TO D
        ENDM
        CSEG
        PUBLIC NUM1,NUM2,NUM3
        EXTRN  MAIN
```

```
CBASE  EQU    0100H
       ORG    CBASE
START: LDAD   NUM1,NUM2,NUM3
       XRA    A          ;CLEAR ACCUMULATOR
LOOP:  LDAX   B          ;LOAD BYTE OF 1ST NO.
       ADC    M          ;ADD BYTE OF 2ND NO.
       STAX   B          ;STORE RESULT
       DCR    D          ;DONE IF REG D = 0
       JZ     MAIN       ;RETURN TO MAIN PROGRAM
       INX    B          ;INCR BC TO NEXT BYTE
       INX    H          ;INCR HL TO NEXT BYTE
       JMP    LOOP       ;ADD NEXT TWO BYTES
NUM1:  DB     090H       ;1ST NUMBER
       DB     0BAH
       DB     084H
NUM2:  DB     08AH       ;2ND NUMBER
       DB     0AFH
       DB     032H
NUM3:  DB     003H
       END    START
```

Once loaded, the ISIS-II assembler performs its operations without further user intervention. In this example, both assembly listing and object output are requested by default. The activation sequence is as follows:

```
-ASM80 MBADD.SRC SYMBOLS XREF MACROFILE
```

The source file is specified as MBADD.SRC. The PRINT control file defaults to MBADD.LST. The OBJECT control file defaults to MBADD.OBJ. A symbol-cross-reference listing is also requested and macros are present.

The assembly and cross-reference listings are shown below. For a detailed explanation of each item in these listings, see Chapter 4.

```
-ASM80 MADD.SRC SYMBOLS XREF MACROFILE
ISIS-II 8080/8085 MACRO ASSEMBLER, V2.0          MBADD          PAGE 1
MULTIBYTE ADDITION PROGRAM

LOC   OBJ        LINE     SOURCE STATEMENT

                   1$     TITLE('MULTIBYTE ADDITION PROGRAM')
                   2      NAME   MBADD
                   3 LDAD MACRO  OPD1,OPD2,COUNT
                   4 L1:  LXI    B,OPD1   ;;ADDRESS OF 1ST NO. TO BC
                   5      LXI    H,OPD2   ;;ADDRESS OF 2ND NO. TO HL
                   6      LXI    D,COUNT  ;;LOOP CONTROL COUNT TO D
                   7      ENDM
                   8      CSEG
                   9      PUBLIC NUM1,NUM2,NUM3
                  10      EXTRN  MAIN
0100              11 CBASE EQU    0100H
0100              12       ORG    CBASE
                  13 START: LDAD  NUM1,NUM2,NUM3
0100 011601    C  14+L1:   LXI    B,NUM1
0103 211901    C  15+      LXI    H,NUM2
0106 111C01    C  16+      LXI    D,NUM3
0109 AF           17       XRA    A        ;CLEAR ACCUMULATOR
100A 0A           18 LOOP: LDAX   B        ;LOAD BYTE OF 1ST NO.
010B 8E           19       ADC    M        ;ADD BYTE OF 2ND NO.
010C 02           20       STAX   B        ;STORE RESULT
010D 15           21       DCR    D        ;DONE IF REG D = 0
010E CA0000    E  22       JZ     MAIN     ;RETURN TO MAIN PROGRAM
0111 03           23       INX    B        ;INCR BC TO NEXT BYTE
0112 23           24       INX    H        ;INCR HL TO NEXT BYTE
0113 C30A01    C  25       JMP    LOOP     ;ADD NEXT TWO BYTES
0116 90           26 NUM1: DB     090H     ;1ST NUMBER
```

```
0117 BA          27          DB      0BAH
0118 84          28          DB      084H
0119 8A          29 NUM2:    DB      08AH      ;2ND NUMBER
011A AF          30          DB      0AFH
011B 32          31          DB      032H
011C 03          32          NUM3:   DB      003H
0100             33          END     START


PUBLIC SYMBOLS
NUM1 C 0116 NUM2 C 0119 NUM3 C 011C

EXTERNAL SYMBOLS
MAIN E 0000

USER SYMBOLS
CBASE  A  0100  L1     A  0100  LOOP  C  010A  MAIN   E  0000
NUM1   C  0116  NUM2   C  0119  NUM3  C  011C  START  A  0100

ASSEMBLY COMPLETE, NO ERRORS
```

Note that the "ASSEMBLY COMPLETE" message is also issued on the console, followed by the cross-reference sign-on message if a cross-reference listing has been requested.

```
ISIS-II ASSEMBLER SYMBOL CROSS REFERENCE, V2.0                    PAGE 1

SYMBOL          CROSS REFERENCE

CBASE           11#     12
L1               4#
LDAD             3#     13
LOOP            18#     25
MAIN            10      22
MBADD            2
NUM1             9      13      14      26#
NUM2             9      13      15      29#
NUM3             9      13      16      32#
START           13#     33

CROSS REFERENCE COMPLETE
```

Control then returns to ISIS-II, which prompts for the next command with a hyphen (-).

Reducing Assembly Time

In the program debugging stage of project development, you can save an assembly pass by specifying NOOBJECT. If you want an error count, specifying NOPRINT with NOOBJECT yields an error count on the console as the only output. If you want just a listing of the lines in error, specify NOLIST and NOSYMBOLS.

Assembly time can be decreased by minimizing operations requiring a lot of diskette activity. INCLUDEs always increase diskette activity. Control lines embedded in your source code cause increased overlay activity in overlay mode. Macro definitions are stored on a diskette; heavy use of macro calls and nested macro definitions cause increased diskette activity.

CHAPTER 4
LIST FILE FORMATS

Assembly Listing Format

The assembly list file is designed for output to a line printer or terminal. Unless otherwise specified, an output page consists of 66 lines, 120 characters wide, and includes three leading and three trailing blank lines, a page header, title line, column headings, and assembly output lines. If a listing line exceeds the right margin setting, it is continued in column 25 of the following line (unless the line exceeds 132 characters, in which case those >132 are truncated).

The first line of the first page of a listing is an echo of the ISIS-II call to the assembler followed by the page header.

If the NOPAGING assembler control is selected, the page header is followed by the title line and column headings, and finally the complete assembly listing with no additional headers.

Page Header

Columns	Description
1-33	The string "ISIS-II 8080/8085 MACRO ASSEMBLER"
34	Blank.
35-38	The string "Vx.y" where "x" is the version and "y" is the release number.
39-46	Blanks.
47-52	Program module name from the NAME directive.
53-64	Blanks.
65-68	The string "PAGE"
69	Blank.
70-72	Three character positions containing the page number in decimal.

Title Line

Columns	Description
1-64	Program title as specified in TITLE assembler control.

Column Heading

Columns	Description
1-2	Blanks.
3-5	The string "LOC"
6-7	Blanks.
8-10	The string "OBJ"
11-16	Blanks.
17-19	The string "SEQ"
31-46	The string "SOURCE STATEMENT"

Assembly Output Line

Columns	Description
1	Assembler error code. If the assembler encountered a syntax error in this source line, the appropriate error code appears in this column. Otherwise, this column is blank. If an error occurs in the present line, the following line will be blank except for a decimal sequence number in columns 3-6 enclosed by parentheses. This sequence number is a pointer to the previous line containing an error. The first error encountered in a program will be followed by a line with a pointer equal to zero. See Chapter 7 for error codes.
2	Minus sign (–) indicates macro definition; otherwise blank.
3-6	The address assigned to the first byte of the object code shown in columns 8-9 of this line is printed in hexadecimal. In addition, the result of the value-generating assembler directives ORG, EQU, SET, and END will appear in this field. For END, the program start address value will appear in this field if specified; otherwise blank.
7	Blank.
8-9	The first byte of object code produced by the assembler for this source line is printed here in hexadecimal. If the source statement produces no object code (comments and assembler directives), this field is blank.
10-11	Second byte of object code in hexadecimal. This field will be blank if the source statement generates only one byte of object code or no object code.
12-13	Third byte of object code in hexadecimal, if generated; otherwise, blank.
14-15	Fourth byte of object code in hexadecimal, if generated; otherwise, blank.
16	Blank.
17	Object code type. The relocatability of an operand is determined by the segment in which the operand lies, not by the segment that references the operand. Values are as follows: C = CODE - segment relocatable D = DATA - segment relocatable S = STACK - segment relocatable M = MEMORY - segment relocatable E = EXTERNAL For a discussion of relocatability, see the *8080/8085 Assembly Language Programming Manual.*
18	Blank if no nested source INCLUDE files; otherwise, the number 1-4 indicating the level of nesting.
19	Blank if not listing a source INCLUDE file; otherwise an "=" sign.
20-23	Four character positions containing the source line number in decimal, right-justified and left blank-filled.
24	Macro expansion flag. A "+" in this column indicates that the source line was produced as a result of a macro expansion. Otherwise, this column will be blank.
25-...	Listing of assembler source text. This field terminates at column 72 for most output devices other than the line printer. For a line printer, this field terminates at column 132.

For DB and DW assembler directives containing a *list* of operands, the generated code for each operand is listed on a separate line.

If a list line exceeds the specified page width, the source line continues starting at column 25 of the next line.

Symbol Table Listing

The listing of the assembled source code is optionally followed by the symbol table listing. If the NOSYMBOLS control is specified, the symbol table listing is suppressed.

The assembler lists the symbol table in three sections: PUBLIC symbols, EXTERNAL symbols, and symbols defined by the user in the current assembly (including PUBLIC and EXTERNAL symbols). Each section of the symbol table is preceded by a title in columns 1-16 of the listing, as follows:

```
PUBLIC SYMBOLS      (Public Symbol Table)
EXTERNAL SYMBOLS    (External Symbol Table)
USER SYMBOLS        (User-Defined Symbol Table)
```

The format of a symbol table output line is as follows:

Columns	Description
1-6	Symbol names up to six characters, left-justified.
7	Blank.
8	Relocatability flag: absolute (A), relocatable CODE-segment (C), relocatable DATA-segment (D), external (E).
9	Blank.
10-13	Symbol value in hexadecimal.
14-17	Blank.
18-n	Repeat pattern of columns 1-17 for additional symbols, where "n" is the page width.

Error Summary

After listing the last line of the symbol table and spacing one line, the assembler lists an error summary line in the following format:

Columns	Description
1-19	The string "ASSEMBLY COMPLETE"
20-23	Number of errors. Four character positions containing the number of errors in the source encountered during assembly. This decimal number is right-justified and left blank-filled. If there are no errors, the string "NO" is output instead.
24	Blank.
25-30	The string "ERRORS"
31	Blank.
32	If the number of errors is not zero, the character "("; otherwise, blank.
33-36	If the number of errors is not zero, these four character positions contain the sequence number (in decimal) of the last source line with an error; otherwise, blank.
37	If the number of errors is not zero, the character ")"; otherwise, blank.

Symbol-Cross-Reference Listing

The assembler generates a file of symbol-cross-reference records during assembly pass 1 if the XREF assembler control is selected. This control is described in Chapter 2. The actual symbol-cross-reference listing is generated by running the XREF utility program, using the ASXREF.TMP file created during pass 1 as input. The utility sorts symbols alphabetically before producing its listing.

Page Header

Columns	Description
1-40	The string "ISIS-II ASSEMBLER SYMBOL CROSS REFERENCE"
41	Blank.
42-45	The string "Vx.y" where "x" is the version and "y" is the release number.
46-64	Blanks.
65-68	The string "PAGE"
69	Blank.
70-72	Three character positions containing the page number in decimal.

Cross-Reference Output Line

Columns	Description
1-6	For lines that start a new entry, this field contains the symbol itself; otherwise, blank.
7	Blank.
8-11	Sequence number of source line containing a reference to or definition of the current symbol entry.
12	Blank if the source line contains a reference; "#" if the source line contains a definition.
13-14	Blanks.
15-68	Repetitions of format in columns 1-14.

If no errors are found during the symbol-cross-reference listing, the message

```
CROSS REFERENCE COMPLETE
```

is issued following the listing. If an error is found, the listing terminates immediately.

CHAPTER 5
PL/M LINKAGE CONVENTIONS

With the relocation feature, it is possible for an assembly language program module to call a procedure originally coded in PL/M. (PL/M procedures can also call assembly language modules.)

Linkage between assembly language modules and PL/M procedures should follow the linkage conventions of PL/M.

Formal parameters declared in a procedure definition are treated as locally-defined variables for the PL/M procedure. That is, each parameter is allocated storage sequentially in memory as if it were a variable local to the procedure. During procedure invocation, actual parameters are evaluated, and the results assigned to the corresponding formal parameters. Parameters are passed as follows:

- A single BYTE parameter is passed in register C. A single ADDRESS parameter is passed in register B (high-order byte) and C (low-order byte).
- If there are two parameters, the first is passed as described above; the second is passed in registers D (high-order byte, if any) and E (low-order byte).
- When there are more than two parameters, the last two are passed as described above, and the remainder are passed via the stack.

All actual parameters are copied into the locations of the formal parameters upon entry to the procedure.

CPU registers are used to hold results returned by procedures which have the BYTE or ADDRESS attribute. In the case of a BYTE procedure, the value is returned in the accumulator, while an ADDRESS procedure returns the low-order byte in the L register and the high-order byte in the H register.

CHAPTER 6
RUNNING YOUR PROGRAM

Absolute Programs

If your program was assembled using the ASEG location counter, you may be able to load and test your program on the Intellec development system immediately after it is assembled. Notice, however, that your program must include an ORG directive that will cause the program to load into a memory location that will not conflict with ISIS-II. (ORG 4000H loads your program well above ISIS-II.) If your program does not include such an ORG directive, it must be reassembled before you attempt to execute it.

To execute your program, you need only enter its object file name in response to an ISIS-II prompt. For example, assume that your program's name is MYPROG. By default, its object file name is MYPROG.OBJ. Entering MYPROG.OBJ after an ISIS-II prompt loads the program and starts its execution.

Program execution begins at the address specified as the operand of the END directive in the source program. If this address is missing or conflicts with ISIS-II, the program cannot be executed.

Relocatable Programs

Some portion of your program has been assembled in the relocatable mode if the program contains a CSEG or DSEG directive. Such programs are not directly executable since some addresses are assembled relative to zero rather than any actual address. To resolve these addresses, you must submit your object file to the LOCATE program. LOCATE supplies absolute addresses for the relocatable addresses. Notice that you can supply absolute load addresses through the LOCATE program. These addresses override any ORG directives in your program. Also, you must be certain to locate your program so that is does not conflict with ISIS-II.

If your program comprises a number of separately assembled modules, you must submit the object file to the LINK program before using LOCATE. The LINK program binds together the separate modules and resolves any external addresses specifed by EXTRN directives.

The final product of LINK (if needed) and LOCATE is an absolute object program file. You can run this program as though it were assembled as an absolute program. Program execution begins at the address specified as the operand of an END directive in the source program. When the program comprises separately assembled modules, only one module may specify a start address in its END directive. If this address is missing or conflicts with ISIS-II, the program cannot be executed.

CHAPTER 7
ERROR MESSAGES

Error Detection and Reporting

The assemblers detect and report three classes of errors: source-file errors (including control line errors), run-time errors, and assembler control syntax errors.

Source-file errors are indicated in the assembly listing by single-letter codes listed in column 1 of the erroneous source statement. If multiple errors occur in the same statement, only the first error is reported. Each error is followed by a pointer to the previous erroneous line to ease finding errors. A summary of source-file errors is sent to the console and list devices.

Run-time errors cause the assembly to terminate abnormally. An error message of the form:

```
error type ERROR
```

is sent to the console and list device (if listing is in progress) and control returns to ISIS-II.

Assembler control errors in the assembler command are reported on the console device with the message:

```
COMMAND ERROR
```

ISIS-II errors are shown as numerical codes. These are listed at the end of this chapter and explained in more detail in the *ISIS-II System User's Guide.*

Error Codes

Source-File Errors

Code	Source
B	Balance error. Parentheses or quote marks are unbalanced.
C	Control line error. An illegal control has been specified in a control line or a primary control appears in illegal context. The erroneous control and following controls on the same line are ignored.
E	Expression error. An expression has been constructed erroneously; usually a missing operator or delimiter. Also caused by adjacent operands with no separating operator.
I	Illegal character. A statement contains an invalid ASCII character, or a specified number is illegal in the context of the number base in which it occurs. Also issued if a carriage return character is not followed by a line-feed character.
L	Location counter error. The symbol being defined has been illegally forward referenced. The definition is made in all cases except macro definitions. This condition can be corrected by moving the definition to precede all references.

Code	Source
M	Multiple definition. A symbol is illegally defined because of prior permanent definition. Only symbols defined by SET and MACRO are redefinable. All occurrences of the multiply-defined item are flagged.
N	Nesting error. Conditional assembly statements of macro body delimiters are improperly nested.
O	Opcode or operand illegal. An opcode or operand illegal in this particular device's instruction set causes a warning.
P	Phase error. Value of symbol being defined has changed between passes 1 and 2 of assembly. Caused by a forward reference of an operand in an ORG, IF, or DS directive.
Q	Questionable syntax. Invalid syntax, usually due to a missing opcode.
R	NAME directive was preceded by an instruction or another directive.
U	Undefined symbol. Symbol used has not been defined.
V	Value illegal. Value exceeds permissible range for this operation or is null.
X	Illegal operand. Specified operand is illegal for this operation. Possible use of register-type symbol in illegal field or use of nonregister type in a field requiring register type.

Run-Time Errors

Message	Explanation
EOF ERROR	End-of-file has been encountered before END directive or END was not terminated by a carriage-return, line-feed.
FILE ERROR	An ISIS file name used in an assembly-time command or control line is illegal or missing. Following this message, ISIS-II will report its own error number (see below).
MEMORY ERROR	System has insufficient memory to execute assembler.
STACK ERROR	Assembler internal stack has overflowed. Possible cause of error: 1. Operators nested more than 16 deep; 2. More than 8 operands in DB or DW list; 3. More than 128 characters in an operand field (probably string too long); 4. Macro or conditional assembly nesting greater than 8 deep. 5. INCLUDEs nested more than 4 deep.
TABLE ERROR	Assembler symbol or macro table has overflowed. More memory needed, or reduce number of symbols or macro definitions/calls.

Assembler Control Error

Message	Explanation
COMMAND ERROR	Assembler console command line syntax is illegal, usually due to missing or illegal delimiter or missing parameter. The entire command line is ignored.

ISIS-II Error Messages

By convention, error numbers 1-99 are reserved for errors that originate in or are detected by the resident routines of ISIS; error numbers 101-199 are reserved for user programs; numbers 200-255 are used for errors encountered by nonresident system routines. In the following list an asterisk precedes fatal errors. The other errors are generally nonfatal unless they are issued by the CONSOLE system call.

0 No error detected.
*1 Insufficient space in buffer area for a required buffer.
2 AFTN does not specify an open file.
3 Attempt to open more than six files simultaneously.
4 Illegal filename specification.
5 Illegal or unrecognized device specification in filename.
6 Attempt to write to a file open for input.
*7 Operation aborted; insufficient diskette space.
8 Attempt to read from a file open for output.
9 No more room in diskette directory.
10 Filenames do not specify the same diskette.
11 Cannot rename file; name already in use.
12 Attempt to open a file already open.
13 No such file.
14 Attempt to open for writing (output or update) or to delete or rename a write-protected file.
*15 Attempt to load into ISIS area or buffer area.
*16 Incorrect ISIS binary format.
17 Attempt to rename or delete a file not on diskette.
18 Unrecognized system call.
19 Attempt to seek in a file not on diskette.
20 Attempt to seek backward past beginning of file.
21 Attempt to rescan a file not line edited.
22 Illegal ACCESS parameter to OPEN or access mode impossible for file specified (input mode for :LP:, for example).
23 No filename specified for a diskette file.
24 Input/output error on diskette (see below).
25 Incorrect specification of echo file to OPEN.
26 Incorrect SWID parameter in ATTRIB system call.
27 Incorrect MODE parameter in SEEK system call.
28 Null file extension.
*29 End of file on console input.
*30 Drive not ready.
31 Attempted seek on file open for output.
32 Can't delete an open file.
33 Illegal system call parameter.
34 Bad RETSW parameter to LOAD.
35 Attempt to extend a file opened for input by seeking past end-of-file.

When error number 24 occurs, an additional message is output to the console:

```
FDCC = 00nn
```

where nn has the following meanings:

01 Deleted record.
02 CRC error (data field).
03 Invalid address mark.
04 Seek error.
08 Address error.
0A CRC error (ID field).
0E No address mark.
0F Incorrect data address mark.
10 Data overrun or data underrun.
20 Write protect.
40 Write error.
80 Not ready.

REQUEST FOR READER'S COMMENTS

The Microcomputer Division Technical Publications Department attempts to provide documents that meet the needs of all Intel product users. This form lets you participate directly in the documentation process.

Please restrict your comments to the usability, accuracy, readability, organization, and completeness of this document.

1. Please specify by page any errors you found in this manual.

2. Does the document cover the information you expected or required? Please make suggestions for improvement.

3. Is this the right type of document for your needs? Is it at the right level? What other types of documents are needed?

4. Did you have any difficulty understanding descriptions or wording? Where?

5. Please rate this document on a scale of 1 to 10 with 10 being the best rating. _____

NAME _____ DATE _____

TITLE _____

COMPANY NAME/DEPARTMENT _____

ADDRESS _____

CITY _____ STATE _____ ZIP CODE _____

Please check here if you require a written reply. ☐

WE'D LIKE YOUR COMMENTS ...

This document is one of a series describing Intel products. Your comments on the back of this form will help us produce better manuals. Each reply will be carefully reviewed by the responsible person. All comments and suggestions become the property of Intel Corporation.

NO POSTAGE
NECESSARY
IF MAILED
IN U.S.A.

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 1040 SANTA CLARA, CA

POSTAGE WILL BE PAID BY ADDRESSEE

Intel Corporation
Attn: Technical Publications M/S 6-2000
3065 Bowers Avenue
Santa Clara, CA 95051

INTEL CORPORATION, 3065 Bowers Avenue, Santa Clara, California 95051 (408) **987-8080**

Printed in U.S.A.