NO STOCK

162
9-30-76

LA-6440-MS

Informal Report

UC-32
Reporting Date: July 1976
Issued: July 1976

Software for the Intel 8Ø8Ø Microprocessor Resident on Machine M (Ø)

by

William M. Seifert

los alamos
scientific laboratory
of the University of California
LOS ALAMOS, NEW MEXICO 87545

An Affirmative Action/Equal Opportunity Employer

UNITED STATES
ENERGY RESEARCH AND DEVELOPMENT ADMINISTRATION
CONTRACT W-7405-ENG. 36

Printed in the United States of America. Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161
Price: Printed Copy $4.00 Microfiche $2.25

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

CONTENTS

NOTICE

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

SOFTWARE FOR THE INTEL 8Ø8Ø MICROPROCESSOR

RESIDENT ON MACHINE M (Ø)

by

William M. Seifert

ABSTRACT

Access to the Intel 8Ø8Ø software which resides in the library of Machine M (Ø) is described. File definitions, control statements, user options and examples are included. All reference manuals required are cited in the introduction.

I. INTRODUCTION

This manual is intended to give the reader a working knowledge of the required control statements necessary for the execution of the Intel 8Ø8Ø software which resides in the library of Machine M (Ø). The following Intel manuals will be of invaluable assistance:

A. "8ØØ8 and 8Ø8Ø PL/M Programming Manual"

B. "8Ø8Ø PL/M Compiler Operators Manual"

C. "8Ø8Ø Assembly Language Programming Manual"

D. "MAC8Ø Reference Specification 8Ø8Ø Macro-Assembler"

E. "INTERP/8Ø User's Manual"

Additionally, familiarity with the NOS Time-Sharing System is assumed. If the reader is not familiar with NOS, the C-Division Consulting Services can offer numerous suggestions in learning NOS, the text editor, and in the writing of "Procedure Files." The following Control Data publications are also extremely valuable:

F. "KRONOS 2.1 Reference Manual, Volume 1"

G. "KRONOS Time-Sharing User's Reference Manual"

H. "Text Editor Reference Manual"

I. "KRONOS Terminal User's Instant Manual"

The LASL publication, "LASL Guide to NOS" (LA-5525-M, Vol. A), outlines the differences between KRONOS and NOS, the new operating system on Machine M (Ø). It also contains a command summary, useful for guide reference, but not for explaining the detailed operation of any control statement.

No explanations on the effects of the various control statements are provided herein. This guide's purpose is to provide the user with a "cookbook" procedure for using the Intel 8Ø8Ø software, along with several realistic examples of the following:

1. Use of the Text Editor.
2. Use of the TEXT mode for permfile creation.
3. Procedure File for using the PL/M compiler.
4. Procedure File for using the MAC8Ø assembler.
5. Obtaining an assembled listing at a 200 terminal (CBT).
6. Reading the object file over NOS.

All of these examples are contained in Section VII, "Sample Programs."

Any comments or suggestions concerning this document or the Intel software packages should be directed to W. M. Seifert, E-5, MS/447.

II. FILE DEFINITIONS

A. User permfiles for using the PL/M compiler:

SOURCE: User's source program with Pass 1 switches inserted at very beginning (Ref. "8Ø8Ø PL/M Compiler Operator Manual," Section 4.1, p. 9).

LIST1: PL/M Pass 1 output listing.

LIST2: PL/M Pass 2 output listing.

SWITCH: PL/M Pass 2 switch settings (Ref. "8Ø8Ø PL/M Compiler Operator's Manual," Sec. 4.2, p. 10).

OBJECT: Object file containing either BNPF or hexadecimal object code, selectable by Pass 2 switches.

B. User permfiles for using the MAC8Ø assembler:

SOURCE: User's source program (assembly language).

LIST: MAC8Ø assembled source listing.

SWITCHA: MAC8Ø switch settings.

OBJECT: Object file (same format feature as PL/M).

C. Machine M (Ø) Library Files: READ ONLY!

PLM81A: PL/M compiler Pass 1.

PLM81B: PL/M compiler Pass 2.

MAC8ØB: MAC8Ø assembler.

INTERPB: INTERP8Ø emulator.

D. Local input/output files - not permfiles:

TAPE20: Input file for MAC8Ø assembler.

TAPE21: Object file for PL/M compiler and MAC8Ø assembler.

TAPE22 &
TAPE23: Linkage files for connecting Pass 1 to Pass 2 of PL/M compiler.

III. CONTROL STATEMENTS FOR PL/M COMPILER

It is assumed that the user will access the PL/M compiler by means of a "Procedure File" (Sec. I., Ref. F., pp. 1-4-4 through 1-4-12). This may be executed from a terminal with the CALL statement (p. 1-4-5) or submitted to the batch stream with the SUBMIT statement (p. 1-6-16). Examples are to be found in Sec. VII.

The following set of control statements represents the minimum necessary for execution of the PL/M compiler:

```
GET(SOURCE)
GET(LGO=PLM81A/UN=LIBRARY)
LGO(SOURCE,LIST1)
REPLACE(LIST1)
REWIND(TAPE22,TAPE23)
GET(SWITCH)
GET(LGO=PLM81B/UN=LIBRARY)
LGO(SWITCH,LIST2)
REPLACE(TAPE21=OBJECT)
REPLACE(LIST2)
RETURN(procedurefilename)
```

NOTE: The recommended minimum memory requirement is 1ØØ5ØØ$_8$.

IV. CONTROL STATEMENTS FOR MAC8Ø ASSEMBLER

Again, the recommended method of access to the MAC8Ø assembler is through a Procedure File. The following set of statements represents the suggested minimum necessary for execution of the MAC8Ø assembler:

```
GET(TAPE2Ø=SOURCE,SWITCHA)
GET(LGO=MAC8ØB/UN=LIBRARY)
LGO(SWITCHA,LIST)
REPLACE(TAPE21=OBJECT)
REPLACE(LIST)
RETURN(procedurefilename)
```

NOTE: The recomended minimum memory requirement is 1ØØ5ØØ$_8$.

V. CONTROL STATEMENTS FOR 8Ø8Ø EMULATOR

Obviously, a Procedure File to execute the emulator is not really necessary, but if the object file name is strictly arbitrary, the following list of control statements represents a completely general method of execution:

```
GET(TAPE21=OBJECT)
GET(LGO=INTERPB/UN=LIBRARY)
LGO.
```

At this point, the emulator program assumes control and indicates this by typing:

INTERP/8Ø VERS x.x

? $F = 1 is typed by user if object file format is hexadecimal.

(see Ref. E., Sec. I.)

NOTE: The recommended minimum memory requirement is 1ØØ5ØØ$_8$.

VI. USER OPTIONS

All user options are outlined in the appropriate Intel manuals cited in Sec. I. These options are selected by the "software switches" and give the user the ability to cause the output listings to be printed at the teletype terminal or any other device, change the object file format to BNPF for PROM programming, etc. It is advisable that the user establish the file for these switch settings before attempting to use the Intel software packages in any way. This will insure that the proper I/O devices are used, the file formatting is correct, etc.

A further option the user may wish to consider is that of creating source files on cassette tape with a Texas Instrument's 733 ASR terminal set in the LOCAL mode (refer to "Model 733 ASR/KSR Operating Instructions"). This method is advantageous from several vantage points:

1. Cards are not wasted in creating or updating a source file, as file creation is via the TEXT mode.
2. An easily stored medium for the source program is realized.
3. The cassettes are, of course, erasable and thus reusable.
4. The cassettes offer a viable method of copying the object files from permfile storage via NOS.
5. Once the application program is debugged, a permfile is no longer needed and both the source program and the object program may be written to the cassettes for long-term storage.

In creating the source file locally, valuable computer time is not consumed and it is easy to edit mistakes from the source program as it is typed by utilizing the local edit capabilities of the TI 733 ASR.

Once the source program has been compiled/assembled, it is not always necessary for the user to generate a printed output listing. Any errors may be detected by means of the text editor, with corrections then implemented on the source file. The process is then iterated until an errorless compilation/assembly results. At that point, a printed listing is then highly desirable and may be obtained by several means:

1. Listed on the TTY terminal with the command LIST,F=localfilename.
2. Listed on the TTY terminal through the text editor.
3. Output to a 200 terminal to the CCF by means of the DISPOSE command (Ref. Sec. I., F., p. 1-7-14). Call the consulting office for the identification of the nearest 200 terminal (CBT).

Finally, it may be that the user will not desire to sit at a terminal until his compilation/assembly is completed. Thus, a job may be submitted to the batch stream by means of the SUBMIT command (Ref. Sec. I., F., p. 1-6-16).

VII. SAMPLE PROGRAMS

Following are a number of examples which illustrate the creation of source programs, the compilation or assembly of such programs, the correction of programming errors, obtaining a program listing, and the recording of the object program on tape cassette. The following list identifies the examples shown:

1. Creation of a PL/M source program using the TEXT mode.
2. Compilation of the example PL/M source program.
3. Correction of errors in the PL/M example source program.
4. Obtaining the compilation listing of the PL/M example.
5. Recording the object code from the PL/M example.
6. Creation of a MAC8Ø source program using the TEXT mode.
7. Assembly of the example MAC8Ø source program.
8. Correction of errors in the MAC8Ø example source program.
9. Obtaining the assembled listing of the MAC8Ø example.
10. Recording the object code from the MAC8Ø example.

Each example has explanations of various control statements interspersed throughout. Various actions required of the operator are also included to clarify any file manipulations, terminal control functions, and correct terminal responses to computer command requests.

All examples conform to the convention:

1. Lower case ASCII characters typed by user.
2. Upper case ASCII characters typed by machine M(Ø).

Where no distinction between upper and lower case characters exists, the context should indicate the character origin.

1. Creation of a PL/M source program using the TEXT mode

This example is taken from the sample PL/M program on page 15 of the "8Ø8Ø Compiler Operators Manual." The procedure named "PRINT$CHAR" has been modified to be compatible with the E-5 microcontroller's serial I/O interface for the TI 733 terminal.

Note that after "logging in" to NOS, a new file is created with the command NEW(permfilename). This command also causes any other files to be dropped from the work space and the file created by the NEW command then becomes the primary file. This means that all subsequent operations which omit reference to a specific file will cause such operations to be done to the primary file. After entering the TEXT mode, any characters entered (except BREAK and CNTRL-C) will be treated as text. In this particular case, the source program was originally created locally on a TI 733 ASR, recording it on a magnetic tape cassette. This meant that no computer time was expended in creating the original source. After the program has been completely debugged and checkout completed, the source program permfile may be copied back onto the same tape cassette for archival storage, and then the source program permfile may then be eliminated.

Although not illustrated here, it is imperative that the software switches for PASS 1 of the PL/M compiler be included in the source program before any program statements or comments. All PASS 1 switches are explained in the "8Ø8Ø PL/M Compiler Operators Manual," Appendix B.

```
 76/06/11. 09.56.00.
LASL 6600 -  0 KRONOS TIME SHARING.     K2.1.2-411J 760526
USER NUMBER:
TERMINAL:      12,TTY
RECOVER/ CHARGE:

READY.
new(exampl1)

READY.
text.
ENTER TEXT MODE.
```

At this point, the tape cassette was inserted, loaded, and the playback control switched to CONT START position. This action caused the contents of the tape to be read by machine M(Ø).

```
/*       SAMPLE PL/M PROGRAM          */
/*
       THIS PROGRAM PRINTS ALL
   NUMBERS BETWEEN 1 AND 1000.
                                      */
DECLARE CRLF DATA(0DH,0AH,00H,00H,
                   00H,00H,00H,00H)
DECLARE HEADING DATA('          ',
                                   ',
        'INTEGER TABLE',0DH,0AH,0AH,
        00H,00H,00H,00H,00H,00H);
DECLARE TTY$RDY LITERALLY '0AH';
DECLARE TTY$OUT LITERALLY '0BH';
DECLARE I ADDRESS;
/*                                    */
PRINT$CHAR: PROCEDURE(CHAR);
   DECLARE CHAR BYTE;
/*   WAIT FOR 733 TO COME READY   */
   DO WHILE(INPUT(TTY$RDY) AND 01H)=00H;
   END;
   OUTPUT(TTY$OUT)=CHAR;
END PRINT$CHAR;
```

```
/*                                *\/
PRINT$STRING: PROCEDURE(NAME,LENGTH);
   DECLARE NAME ADDRESS;
   DECLARE (LENGTH,I,CHAR BASED NAME) BYTE;
   DO I = 0 TO LENGTH-1;
      CALL PRINT$CHAR(CHAR(I));
   END;
END PRINT$STRING;
/*                                */
PRINT$NUMBER: PROCEDURE(NUMBER,BASE,
               CHARS,ZERO$SUPPRESS);
   DECLARE NUMBER ADDRESS, (BASE,CHARS,
        ZERO$SUPPRESS,I,J) BYTE;
   DECLARE TEMP(16) BYTE;
   IF CHARS > LAST(TEMP) THEN
      CHARS = LAST(TEMP);
   DO I = 1 TO CHARS;
      J = NUMBER MOD BASE + '0';
      IF J > '9' THEN J = J + 7;
      IF ZERO$SUPRESS AND I <> 1 AND
         NUMBER = 0 THEN J = ' ';
      TEMP(LENGTH(TEMP) - I) = J;
      NUMBER = NUMBER / BASE;
   END;
   CALL PRINT$STRING(.TEMP + LENGTH(TEMP)
                     - CHARS, CHARS);
END PRINT$NUMBER;
```

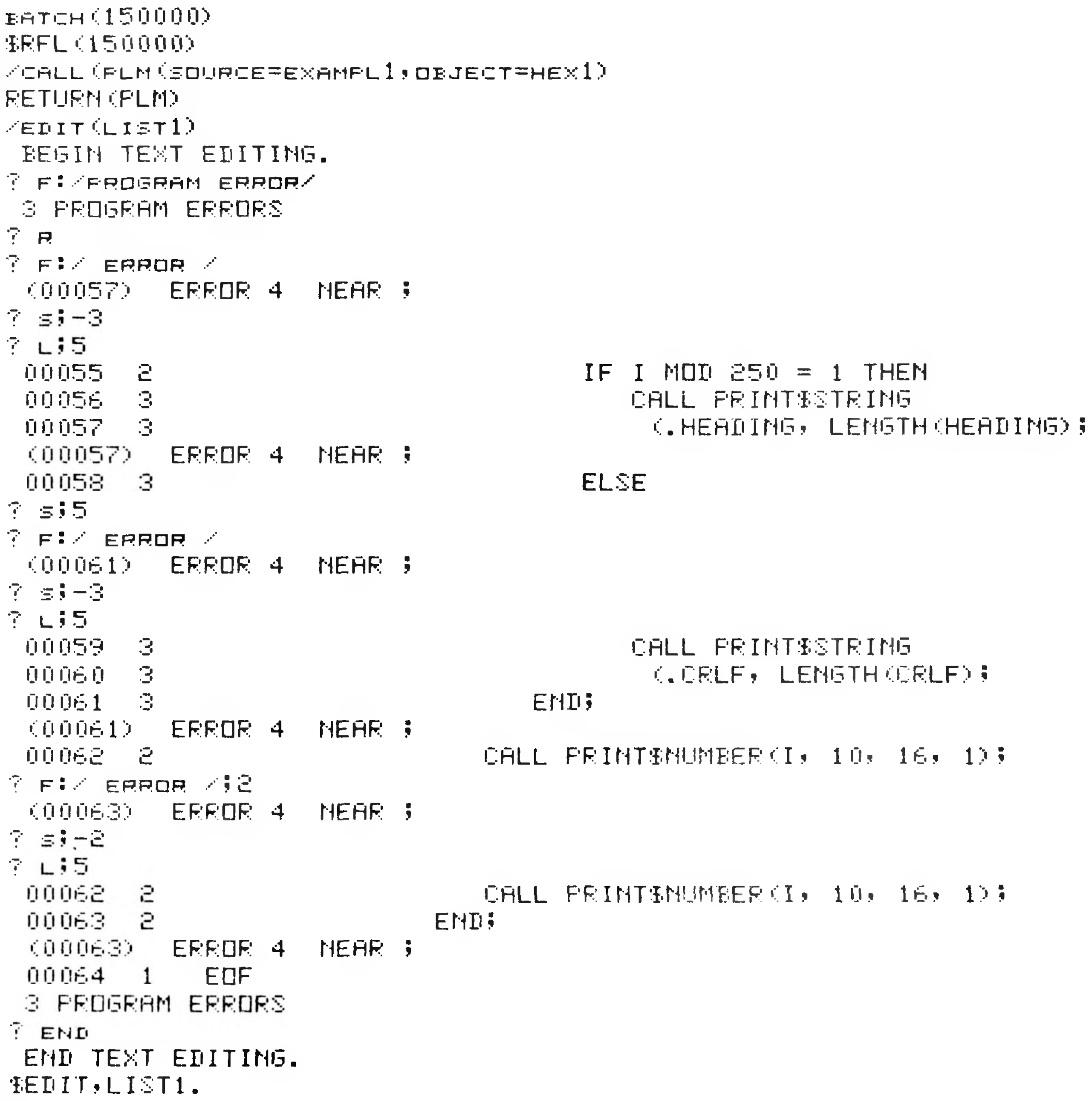
```
       /*                                */

       BEGIN: DISABLE;

              DO I = 1 TO 1000;

                 IF I MOD 5 = 1 THEN

                    DO;

                       IF I MOD 250 = 1 THEN

                          CALL PRINT$STRING

                           (.HEADING, LENGTH(HEADING));

                       ELSE

                          CALL PRINT$STRING

                           (.CRLF, LENGTH(CRLF));

                    END;

                 CALL PRINT$NUMBER(I, 10, 16, 1);

              END;

EOF
```

At this point, if the 733 is equipped with either the Automatic Device Control or Remote Device Control options and if the DC3 character is enabled while transmitting, the playback will be automatically turned off when the DC3 (TAPE OFF) character is read from the tape cassette. It may be preceded by the ETX character which will cause the following to take place:

```
EXIT TEXT MODE
NOSORT
READY.
PACK

READY.
REPLACE(EXAMPL1)
```

2. <u>Compilation of the example PL/M source program</u>

Following is an example terminal session to compile the sample PL/M program and to find any compilation errors. Note that both list files, LIST1 and LIST2, must be examined for errors, even though errors found in PASS 2 may be due to errors in PASS 1.



```
BATCH(150000)
$RFL(150000)
/CALL(PLM(SOURCE=EXAMPL1,OBJECT=HEX1)
RETURN(PLM)
/EDIT(LIST1)
 BEGIN TEXT EDITING.
? F:/PROGRAM ERROR/
 3 PROGRAM ERRORS
? R
? F:/ ERROR /
 (00057)   ERROR 4   NEAR ;
? S;-3
? L;5
 00055  2                          IF I MOD 250 = 1 THEN
 00056  3                             CALL PRINT$STRING
 00057  3                              (.HEADING, LENGTH(HEADING));
 (00057)   ERROR 4   NEAR ;
 00058  3                          ELSE
? S;5
? F:/ ERROR /
 (00061)   ERROR 4   NEAR ;
? S;-3
? L;5
 00059  3                             CALL PRINT$STRING
 00060  3                              (.CRLF, LENGTH(CRLF);
 00061  3                       END;
 (00061)   ERROR 4   NEAR ;
 00062  2                    CALL PRINT$NUMBER(I, 10, 16, 1);
? F:/ ERROR /;2
 (00063)   ERROR 4   NEAR ;
? S;-2
? L;5
 00062  2                    CALL PRINT$NUMBER(I, 10, 16, 1);
 00063  2                 END;
 (00063)   ERROR 4   NEAR ;
 00064   1    EOF
 3 PROGRAM ERRORS
? END
 END TEXT EDITING.
$EDIT,LIST1.
```

```
/EDIT(LIST2)
 BEGIN TEXT EDITING.
? F:/PROGRAM ERROR/
 1 PROGRAM ERROR
? R
? F:/ ERROR /
 (00064)  ERROR 144
? S;-2
? L;5
    45=0123H   46=0137H   47=0143H   48=0146H   49=015CH   50=015DH
    52=015EH   53=0175H   54=018AH   55=018DH   56=01A2H   57=01A5H
 (00064)  ERROR 144
 STACK SIZE = 4 BYTES
 MEMORY..........................4000H
? END
 END TEXT EDITING.
$EDIT,LIST2.
```

The actual compilation is done by the CALL(PLM(...) statement. The text editor is then used to examine both LIST1 and LIST2 for errors, which can then be related back to the original source program for correction.

3. Correction of errors in the PL/M example source program

To correct the errors in the source program, it is usually necessary to identify the error with a unique line or phrase. This is done in the following example by means of the FIND editor command. Corrections are accomplished with the REPLACE STRING command, and the result is shown with the LIST command.

```
/EDIT(EXAMPL1)
 BEGIN TEXT EDITING.
? F:/ LENGTH(HEADING);/
                                (.HEADING, LENGTH(HEADING);

? RS:/);/,/));/
? L
                                (.HEADING, LENGTH(HEADING));

? F:/ LENGTH(CRLF);/
                                (.CRLF, LENGTH(CRLF);

? RS:/);/,/));/
? L
                                (.CRLF, LENGTH(CRLF));

? END
 END TEXT EDITING.
$EDIT,EXAMPL1. '
/REPLACE(EXAMPL1)
```

4. Obtaining the compilation listing of the PL/M example

Correction of errors in the source program must be followed by another compilation and search of the list files for additional errors. In this example, the compilation is accomplished again with the CALL statement, and a subsequent search of both list files reveals no program errors have resulted from this compilation. At this point, a compilation listing of both passes of the PL/M compiler is desired so that the user may execute the program object code either by means of the 8Ø8Ø simulator, INTERP/8Ø, or on the user's actual 8Ø8Ø-based system in real time. Such a listing may be obtained in three ways:

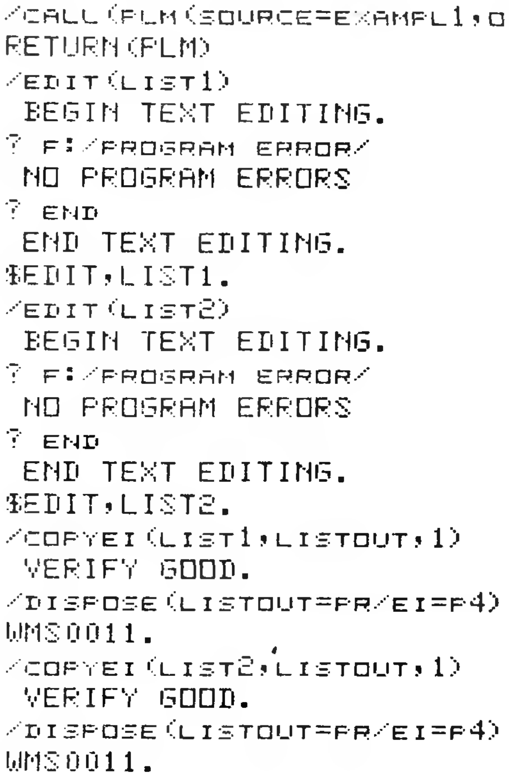
a. For short programs, the list files can be output to the timesharing terminal in two ways:

   1) Through the text editor;

   2) By use of the command LIST,F=localfilename.

b. For longer programs where the 300 baud (30 characters/s) rate is too slow to be practical the list files may be output to a 200 terminal by means of the DISPOSE command.

For this example, the last method has been selected. Note that the permfiles LIST1 and LIST2 have been copied to the local file LISTOUT before the DISPOSE command is invoked. This is done to insure the correct file type is being output to the 200 terminal.



```
/CALL(PLM(SOURCE=EXAMPL1,OBJECT=HEX1)
RETURN(PLM)
/EDIT(LIST1)
 BEGIN TEXT EDITING.
? F:/PROGRAM ERROR/
 NO PROGRAM ERRORS
? END
 END TEXT EDITING.
$EDIT,LIST1.
/EDIT(LIST2)
 BEGIN TEXT EDITING.
? F:/PROGRAM ERROR/
 NO PROGRAM ERRORS
? END
 END TEXT EDITING.
$EDIT,LIST2.
/COPYEI(LIST1,LISTOUT,1)
 VERIFY GOOD.
/DISPOSE(LISTOUT=PR/EI=P4)
WMS0011.
/COPYEI(LIST2,LISTOUT,1)
 VERIFY GOOD.
/DISPOSE(LISTOUT=PR/EI=P4)
WMS0011.
```

5. Recording the object code from the PL/M example

Since the object code for the 8Ø8Ø is output to a permfile, some recording medium must be found which is common to both the CCF and the user's 8Ø8Ø-based system. The possible alternatives are presently limited to two types of media: magnetic tape cassette and paper tape. Both media have advantages and disadvantages, so a discussion of the relative merits of each shall not be undertaken here. Group E-5 has relied on the magnetic tape cassettes simply because of the widespread availability of TI "Silent 7ØØ" terminals with cassette tape transports. This makes the desirable features of cassette tape available to any user within LASL who has access to such a terminal.

Intel provides two types of object file format: BNPF for PROM programming, and hexidecimal. Both have, of course, ASCII representations, but the obvious advantage of the hexadecimal format over that of BNPF is its brevity. Only two ASCII hexadecimal numbers are required to represent an 8-bit word, whereas ten ASCII characters in BNPF format are required. The type of object code format is determined by the value of the software switch, Q, so that either format may be easily specified. Either format is accepted by the 8Ø8Ø simulator INTERP/8Ø.

To execute the object code on the user's 8Ø8Ø system, two approaches are possible:

1. Program a set of PROMs which are then properly mapped in the user's memory for proper program addressing and execution;
2. Write a loader program which can reside in PROM memory, then read the application program into read/write memory (RAM) from which the program may then be executed, debugged, and altered to some limited extent.

The latter approach is commonly used to properly configure the application program to the user's system. Of course, this requires that the application program be written such that the program origin will correctly map the object code into the user's system RAM. The program origin may be easily altered by means of a numeric label which specifies the absolute address of the program at that point.

In this example, no explicit starting address appears in the source program so the starting address is implicitly assumed to be ØØØØH. Following is the listing of the hexadecimal object code for example 6:

```
READY.
GET(HEX1)

READY.
EDIT(HEX1)

 BEGIN TEXT EDITING.
? F:/$/
  $
? L;◆
  $
 ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
 :1000000031E03FC35D010D0A000000000000202028
 :1000100020202020202020202020202020202020E0
 :1000200020202020202020202020202020202020D0
 :10003000494E54454745522054414243450D0A0A09
 :1000400000000000000021E33F71DB0AE601D6005A
 :10005000CA4A0021E33F7ED30BC921E43F712370DC
 :100060002C732C360021E63F4E0D792C96DA830056
 :100070004E06002AE43F097E4FCD460021E73F347B
 :10008000C26500C921EC3F712C733E0F2D96D293AF
 :1000900000360F2EEE360121EC3F7E2EEE96DA432F
 :1000A0000121EB3F5E16002EE84E2C46C3DC007AA1
 :1000B0002F577B2F5F132100003E11E519D2C1009D
 :1000C000E3E1F579174F7817477D176F7C1767F1D4
 :1000D0003DC2BB00B77C1F577D1F5FC9CDAF00017C
 :1000E0003000EB09EB21EF3F733E3996D2F3007EEF
 :1000F000C607772D4E0D3EFFC2FC00AF2DA62EE8A1
 :100100004F7E2C56D6005F7ADE00B3D6019FA10F3A
 :10011000D217012EEF36203E102EEE964F06002EFF
 :10012000F009EB21EF3F4E791221EB3F5E16002ED6
 :10013000E84E2C46CDAF0021E83F7123702EEE34FF
 :10014000C2970001F03F111000696019EB7B21ECB0
 :100150003F965F7ADE004B475ECD5A00C9F321E03F
 :100160003F360123360 03EE8060321E03F962C4F40
 :10017000789EDADD011E05160021E03F4E2C46CDAB
 :10018000AF007BD6015F7ADE00B3C2B8011EFA165B
 :100190000021E03F4E2C46CDAF007BD6015F7ADEDA
 :1001A00000B3C2B001010E001E38CD5A00C3B80121
 :1001B0000106001E08CD5A0021E03F4E2C462EE8D5
 :1001C0007123702EEB360A0E101E01CD84002EE036
 :0F01D0004E2C462101000922E03FC36601FB7659
 :0000000000
 ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
  $
 -END OF FILE-
? END
 END TEXT EDITING.
```

6. Creation of a MAC8Ø source program using the TEXT mode

This example is taken from the sample assembly language program on p. 55 of the "8Ø8Ø Assembly Language Programming Manual." This program is modified to await the typing of a RETURN. It then outputs two numbers being added, then outputs the sum. The source permfile has been created in the same manner as the PL/M example.

```
READY.
NEW(EXAMPL2)

READY.
TEXT.
ENTER TEXT MODE.




;    THIS PROGRAM WILL, UPON RECEIPT OF A "RETURN"

; FROM A TI 733 ASR, OUTPUT TWO NUMBERS, ADD THE

; TWO TOGETHER, THEN OUTPUT THE SUM.

;

;

;          ADDRESS DEFINITION

;

  STACK       EQU       3F20H

  FIRST       EQU       3F40H

  SECND       EQU       3F60H

;

;          DEFINE PROGRAM ORIGIN

;

              ORG       4000H

              JMP       BEGIN
```

```
;
;         DATA DEFINITION
;
   CRLF:       DB        0DH,0AH,7FH,7FH,
               DB        7FH,7FH,7FH,00H
   HEADR:      DB        '                ',
               DB        'NO. 1           ',
               DB        '      NO. 2     ',
               DB        '             SUM',00H
   TAB:        DB        '                ',00H
;
;         SUBROUTINE DEFINITION
;
   TTY$IN:     IN        0AH
               ANI       02H
               SUI       00H
               JZ        TTYIN       ; TTY NOT RDY!
               IN        0BH         ; TTY RDY
               RET
;
   TTOUT:      IN        0AH         ; TTY RDY?
               ANI       01H
               SUI       00H
               JZ        TTOUT       ; TTY NOT RDY!
               MOV       A,C
               OUT       0BH         ; YES - OUTPUT
               RET                   ;  CHAR & RETURN!
```

```
;
    LINE:        MOV     C,M        ; FETCH CHAR
                 CALL    TTOUT
                 CPI     00H        ; LAST CHAR?
                 RZ                 ; IF SO, RETURN!
                 INX     H          ; NOT DONE!
                 JMP     LINE       ; ITERATE!
;
    OBYTE:       MVI     B,02H      ; SET UP COUNTER
                 MOV     D,M
    BYTLP:       MOV     A,D
                 RRC                ; ROTATE BYTE
                 RRC                ;  RIGHT 4 BITS
                 RRC
                 RRC
                 MOV     D,A        ; REPLACE IN D REG
                 ANI     0FH        ; STRIP L.S. HALF-BYTE
                 CPI     0AH        ; >= 10?
                 JC      BYTLO
                 ADI     07H
    BYTLO:       ADI     30H        ; CONVERT TO ASCII
                 MOV     C,A
                 CALL    TTOUT
                 MOV     A,B
                 SUI     01H
                 MOV     B,A
                 JNZ     BYTLP      ; NOT FINISHED!
                 RET
```

```
;
;          MAIN PROGRAM
;
   BEGIN:      LXI       SP,STACK    ; INITIALIZE SP!
               LXI       H,CRLF
               CALL      LINE        ; RETURN 733 CARR.
               LXI       H,HEADR
               CALL      LINE        ; PRINT HEADER
               LXI       H,CRLF
               CALL      LINE
               CALL      LINE
   ITR8:       CALL      TTYIN
               CPI       0DH         ; COMMAND TO START?
               JNZ       $-5
               LXI       H,TAB       ; TAB FO 1ST NUMBER
               CALL      LINE
               LXI       H,FIRST+2   ; SET ADDRESS PTR TO
               MVI       M,84H       ;  M.S. DIGIT
               CALL      OBYTE
               INX       H
               MVI       M,0BAH      ; STORE NEXT M.S.D.
               CALL      OBYTE
               INX       H
               MVI       M,90H       ; STORE L.S.D.
               CALL      OBYTE
               LXI       H,TAB       ; TAB FOR 2ND NUMBER
               CALL      LINE
               LXI       H,SECND+2   ; SET-UP ADDRESS FOR
               MVI       M,32H       ;  2ND NUMBER
               CALL      OBYTE
               INX       H
               MVI       M,0AFH
               CALL      OBYTE
               INX       H
               MVI       M,8AH
               CALL      OBYTE
               LXI       H,TAB
               CALL      LINE
               MVI       D,03H
;
;         ADDITION ROUTINE
;
   MADD:       LXI       B,FIRST
               LXI       H,SECND
               XRA       A           ; RESET CARRY FLAG
   LOOP:       LDAX      B           ; LOAD A INDIRECTLY
               ADC       M           ; ADD MEM TO A
               STAX      B           ; REPLACE AT FIRST+N
               DCR       D           ; ONE PASS COMPLETE
               JZ        DONE        ; IF DONE, EXIT
               INX       B           ; SET POINTER TO NEXT NO.
               INX       H           ;  FOR FIRST AND SECND
               JMP       LOOP        ; ITERATE
   DONE:       MOV       L,C         ; SET UP ADDRESS OF
```

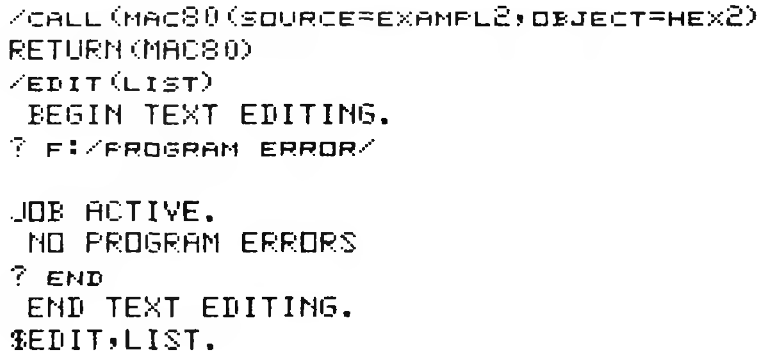
```
                    MOV       H,B           ;  RESULTANT
                    CALL      OBYTE         ; OUTPUT RESULT
                    DCX       H             ;  WITH M.S.D. FIRST
                    CALL      OBYTE
                    DCX       H             ; L.S.D. LAST
                    CALL      OBYTE
                    LXI       H,CRLF        ; RETURN CARR. & LF
                    CALL      LINE
                    JMP       ITR8          ; WAIT FOR COMMAND
                    END                     ;  TO REPEAT
$EOF


EXIT TEXT MODE
NOSORT
READY.
PACK

READY.
REPLACE(EXAMPL2)
```

Again, to exit the text mode, the ETX character is sent either from being locally recorded on the tape cassette or from the keyboard.

Note the terminating statement in the source file, '$EOF.' This serves as the end of file mark and it is essential that the dollar sign ($) appear in column 1 of the source file.

7. Assembly of the example MAC8Ø source program

Following is an example terminal session to assemble the sample MAC8Ø program and to detect assembly errors. Only the file LIST must be examined to find any such errors. In this case, after issuing the FIND command to the text editor, a second carriage return was issued to check whether the computer was still active. The reply "JOB ACTIVE" was in response to this second carriage return.



```
/CALL(MAC80(SOURCE=EXAMPL2,OBJECT=HEX2)
RETURN(MAC80)
/EDIT(LIST)
 BEGIN TEXT EDITING.
? F:/PROGRAM ERROR/

JOB ACTIVE.
 NO PROGRAM ERRORS
? END
 END TEXT EDITING.
$EDIT,LIST.
```

8. Correction of errors in the MAC8Ø example source program

Since there were no assembly errors in the above sample program assume that the simulator was then executed revealing some logical errors. It turns out that for the MAC8Ø example, the section of code which causes the two numbers to be written into read/write memory is incorrectly written. The instruction "INX H" should have been written "DCX H" where the data is being written into memory.

```
/OLD(EXAMPL2)
/EDIT(EXAMPL2)
 BEGIN TEXT EDITING.
? F:/FIRST+2/
                      LXI       H,FIRST+2    ; SET ADDRESS PTR TO
? L;8
                      LXI       H,FIRST+2    ; SET ADDRESS PTR TO
                      MVI       M,84H        ;  M.S. DIGIT
                      CALL      OBYTE
                      INX       H
                      MVI       M,0BAH       ; STORE NEXT M.S.D.
                      CALL      OBYTE
                      INX       H
                      MVI       M,90H        ; STORE L.S.D.
? RS:/INX/,/DCX/;2
? L;8
                      LXI       H,FIRST+2    ; SET ADDRESS PTR TO
                      MVI       M,84H        ;  M.S. DIGIT
                      CALL      OBYTE
                      DCX       H
                      MVI       M,0BAH       ; STORE NEXT M.S.D.
                      CALL      OBYTE
                      DCX       H
                      MVI       M,90H        ; STORE L.S.D.
? F:/SECND+2/
                      LXI       H,SECND+2    ; SET-UP ADDRESS FOR
? L;8
                      LXI       H,SECND+2    ; SET-UP ADDRESS FOR
                      MVI       M,32H        ;  2ND NUMBER
                      CALL      OBYTE
                      INX       H
                      MVI       M,0AFH
                      CALL      OBYTE
                      INX       H
                      MVI       M,8AH
? RS:/INX/,/DCX/;2
? L;12
                      LXI       H,SECND+2    ; SET-UP ADDRESS FOR
                      MVI       M,32H        ;  2ND NUMBER
                      CALL      OBYTE
                      DCX       H
                      MVI       M,0AFH
                      CALL      OBYTE
                      DCX       H
                      MVI       M,8AH
                      CALL      OBYTE
                      LXI       H,TAB
                      CALL      LINE
                      MVI       D,03H
? END
 END TEXT EDITING.
$EDIT,EXAMPL2.
/REPLACE(EXAMPL2)
```

9. <u>Obtaining the assembled listing of the MAC8Ø example</u>

Correction of errors in the source program must be followed by another assembly and search of the file LIST to find subsequent errors. After all such errors have been corrected, an assembly listing is desirable for debugging and checkout. This can be done in the same ways as the PL/M example (item 4).

Following are the control statements necessary for printing the assembly listing at the P-Division 200 terminal:

```
/COPYEI(LIST,LISTOUT,1)
 VERIFY GOOD.
/DISPOSE(LISTOUT=PR/EI=P4)
WMS0011.
```

10. <u>Recording the object code from the MAC8Ø example</u>

The object file shown below has been recorded on magnetic tape cassette using the command LIST,F=localfilename. The object file format is the same as that of PL/M (see example 5).

```
READY.
GET(HEX2)

READY.
LIST,F=HEX2

 :10400000C3A0400D0A7F7F7F7F7F002020202020DB
 :1040100020202020202020202020204E4F2E203124
 :10402000202020202020202020202020202020202090
 :10403000204E4F2E2032202020202020202020202003
 :104040002020202020205355 4D00202020202020FB
 :104050002020202020202020202020 00DB0AE60233
 :10406000D600CA5C40DB0BC9DB0AE601D600CA6891
 :104070004079D30BC94ECD6840FE00C823C37540BC
 :104080000602567A0F0F0F0F57E60FFE0ADA92401C
 :10409000C607C6304FCD684078D60147C28340C9B5
 :1040A00031203F21034 0CD7540210B40CD7540218B
 :1040B0000340CD7540CD7540CD5C40FE0DC2B8408B
 :1040C000214B40CD754021423F3684CD80402B3678
 :1040D000BACD80402B3690CD8040214B40CD7540ED
 :1040E00021623F3632CD80402B36AFCD80402B361B
 :1040F0008ACD8040214B40CD7540160301403F21C1
 :10410000603FAF0A8E0215CA0F410323C303416902
 :1041100060CD80402BCD80402BCD8040210340CD11
 :054120007540C3B8402A
 :0000000000
 $
READY.
```

★ US GOVERNMENT PRINTING OFFICE: 1976—677-343/180