Title
LBLHEX - A HEX MONITOR PROGRAM FOR 8080 CODE-COMPATIBLE MICROPROCESSORS

Author
Katz, Joseph E.

Publication Date
1978-03-01

Submitted to BYTE Magazine

LBL-7531 c. 1
Preprint

LBLHEX - A HEX MONITOR PROGRAM FOR 8080 CODE-COMPATIBLE MICROPROCESSORS

Joseph E. Katz

March 1978

Prepared for the U. S. Department of Energy
under Contract W-7405-ENG-48

DISCLAIMER

LBLHEX - A HEX MONITOR PROGRAM FOR 8Ø8Ø CODE-COMPATIBLE MICROPROCESSORS

LBL-7531

CONTENTS

LBL-7531

LBLHEX - A HEX MONITOR PROGRAM FOR 8080 CODE-COMPATIBLE MICROPROCESSORS

Joseph E. Katz

Lawrence Berkeley Laboratory
University of California
Berkeley, California

March 1978

ABSTRACT

LBLHEX is a hexadecimal monitor program for 8080 code-compatible microprocessors. The program allows the user, via simple keyboard commands, to examine and modify all of memory and to transfer program control to any memory location. It has a multiple pass breakpoint feature, a unique feature of this program, which is of great value in the development and debugging of microprocessor programs. Routines to check and program EPROMs by either remote or resident programmers are provided. The ability to read or write object code to peripherals or other computer systems is also included.

This work was supported by the Division of Chemical Sciences, Office of Basic Energy Research, U.S. Department of Energy.

INTRODUCTION

This hexadecimal monitor program was written to provide in one program some new features and options that have appeared in one monitor program or another. The language is meant to be forgiving, easy to remember, and convenient to use.

The program occupies slightly less than $2\emptyset48_{1\emptyset}$ words and therefore may be located on one 2716 EPROM. No effort has been made to minimize the code since the cost of EPROM memory continues to rapidly decrease.

This write-up and description is meant to briefly describe the code and its features. The program is written so it may easily be reassembled at any memory location and may be used with a wide variety of peripheral devices. It is assumed that the reader is somewhat familiar with 8Ø8Ø code-compatible microprocessors and its jargon. Therefore terms like EPROM, USART, RAM, HEX, RESTART, ETC. will not be defined as they are introduced.

If the reader is in doubt, there are innumerable microprocessor introductory tutorials available with extensive glossaries, Ref. 1, for example.

LBLHEX, a linear descendant of ODT[2,3], has proved to be an effective debugging and system development tool. The tradition of soft front panels is well served by LBLHEX. This monitor program has on many occasions been the only tool required to commission and maintain a microprocessor system.

SYSTEM REQUIREMENTS

The minimum system requirements for using LBLHEX, as it is assembled, are as follows:

1) Any 8Ø8Ø code-compatible microprocessor
2) $2Ø48_{1Ø}$ of read only memory at location ØØØØH.
3) RAM memory for use as stack pointer and scratch pad from 83ØØH to 83FFH.
4) A terminal interface with input and output via I/O port 2. Status and control word, for a USART or UART at I/O port 3. Status bit Ø indicates that the terminal is free to send another character and status bit 1 indicates that a keyboard character has been received.

A program listing of LBLHEX is provided in Appendix A. The program may be reassembled with the scratch memory (RAM) at any location by changing the address of the STACK variable at the top of the assembly listing page 2. The main body of the program may be assembled at any memory location (EPROM) by changing the ORG statement shown in the assembly listing page 3.

LBLHEX uses 3 of the 8 RESTART/INTERRUPT locations of the 8Ø8Ø microprocessor. It is very convenient to have your monitor program come up when the system is turned on or restarted. Therefore location ØØØØH is dedicated to initializing the console device and transferring control to the monitor program.

RESTART/INTERRUPT locations 3ØH and 38H are dedicated to the Breakpoint and Trap feature of the monitor. Breakpoint is defined as a deliberate interruption of a program, installed by the user, to examine the contents

of the registers, the stack pointer, the stack contents, and the flag word, to evaluate the operation of the program under development. A trap is an interruption of program caused by the execution of a memory location containing the code FFH. This may be the result of addressing non-existant memory. In the event of a trap, it is certainly helpful to have the contents of the registers, the address of the trap, the stack contents, etc. printed out on the console device.

The other RESTART/INTERRUPT locations shown in the assembly listing are programmed to allow the user to transfer execution to other routines. Microprocessor controlled pieces of hardware that we have built at LBL* have a front panel RESTART button and an octal switch which allows the user to choose which program mode the hardware should respond in. The assembly listing, Appendix A, pages 2 and 3, show the options provided. We will not describe these other programs at this time but refer the reader to a fine description[4] of a microprocessor system philosphy.

INPUT/OUTPUT

All input/output for LBLHEX is performed through calls to subroutines CI (any console input), RI (read in, paper tape, cassette, etc.), CO (any console ouput), and PO (punch output, paper tape, cassette, etc.). The assembly listing, page 3, shows jumps to teletype routines. LBLHEX is designed to allow the user to communicate easily with a large variety of I/O devices. For a microprocessor system with other peripheral devices, one may simply replace the teletype routines with the appropriate drivers.

The I/O routines may be reconfigured to satisfy the more elegant system with several different console devices. The I/O jump table, may be

*LBL - Lawrence Berkeley Laboratory of the University of California in Berkeley

changed, to point to a routine to check a (software or hardware) switch which determines the I/O driver and device in use.

The I/O routines and subroutines, of LBLHEX, may be used by other system programs. The calling protocol and affected registers are described in the Useful Subroutines section of this write-up.

KEYBOARD COMMANDS

A brief description of each element of the command structure is given on page 1 of the assembly listing. This short listing is intended to be used as a handy reminder which may be posted on or about your console device.

Commands, generally consist of a single character followed by one or more hexadecimal numbers. In the absence of a number, a reasonable default value is assumed. The default value for each command element is described below and in the short form command menu, see page 1 of the assembly listing. Commands are terminated by a carriage return, (CR), or the send key on some terminals. This mode of command character, data or address options and an execute, (CR), allows the user to verify his/her input before execution.

A number in an address field consists of one to four hexadecimal characters,(Ø to 9,A to F). Since only the last four characters before the terminator are considered, one may correct any mistakes by continuing on until the correct address has been typed. Of course, the number in a memory byte or number-of-times field consists of a maximum of two hexadecimal characters. In this case, only the last two characters are considered. Leading zeros are assumed, address (ØØA5) may be entered as A5 and data byte (Ø3) may be entered as 3.

Upon a RESTART/INTERRUPT(front panel push-button, option Ø) or when power first comes on, program control is transferred to LBLHEX. The response is the asterisk character (*), to signify that LBLHEX is ready to accept

input commands. Upon completion of any assigned task, the program response is the asterisk (*).

For a non-hexadecimal character in the significant characters of a numeric field or a non-existent command character, the response is a question mark (?). This error return message is followed by a carriage return, line feed and the reinitialization of LBLHEX signified by the asterisk (*).

GO

This command allows the user to transfer program control to any address by typing *GNNNN(CR). Where NNNN is the hexadecimal address to which you wish to transfer program control. (CR) indicates a carriage return or the send key on some terminals, therefore to transfer control to some program located at a address of 1ØØØH, one would type:

*G1ØØØ(CR)

An output to indicate the response of LBLHEX to a non-numeric character in an address field to shown below:

*G78A9K(CR)

?

*

The non-hex K in the significant portion of the address field caused the question mark response (?) and the return to LBLHEX (*).

BREAKPOINT

This feature allows the user to cause an interrupt of program execution at any address. Upon a break, all of the register pairs, the address of the breakpoint, the stack pointer, the contents of the stack, and the flag word will be printed on the console device.

At the end of a break, the contents of the breakpoint memory location are automatically restored with their original contents, and program control is

transferred to LBLHEX. The user may then use any of its features to examine, modify, dump, or install another breakpoint before making another pass of the program in question.

A novel feature of LBLHEX, as far as I know, is the provision of allowing a specified number of passes (1 to 255(FFH) are the range permitted) of the code in question through the breakpoint before allowing the break.

The command structure for the breakpoint feature is BNNNN,MM,LL(CR) NNNN is the address of the breakpoint and MM is the number of bytes in the instruction at that address. Since multiple passes are allowed, it is necessary for the monitor program to know the number of bytes in the breakpoint instruction. The interested reader may pursue this point in the assembly listing, e.g., see subroutine BKPNT, shown on page 5 and 6 of the listing, Appendix A. The break will occur on the LLth time that program control passes through address NNNN.

A typical output listing showing the use of the breakpoint feature of the monitor is shown below:

```
*B5A49,3,1Ø(CR)
*G4713(CR)
READY
RUN
INPUT STARTING ENERGY = :59.5EV
A F  B C  D E  H L  ADD  SP  STACK  F=S Z 0 AC 0 P 1 CY
8Ø56 ØØ38 ØØØ1 8Ø49 5A49 83DØ ØAB5    Ø 1 Ø 1  Ø 1 1 Ø
*
```

A breakpoint was installed at address 5A49H. The 3 byte instruction starting at that address is to be interrupted on the 16th (1ØH) pass. The user then transferred program control to location 4713H (In this example, a BASIC

Interpreter with a stored program). The next 3 lines are a BASIC reply, a command, and a programed input request. After 16 iterations through address 5A49H in an assembly language subroutine, a break was initiated and the dump, shown above, was printed. The original 3 byte instruction starting at address 5A49H was restored and program control was transferred to LBLHEX (*).

If a break is to occur on the first pass, a default command of BNNNN(CR) may be used. An output showing this appears below:

```
*B59Ø3(CR)
*G59ØØ(CR)
A F  B C  D E  H L  ADD  SP  STACK  F=S Z 0 AC 0 P 1 CY
1553 ØlAD ØlAD 59Ø3 59Ø3 8912 59ØF    Ø 1 Ø 1  Ø Ø 1 1
*
```

TRAP

Execution of the code FFH will cause a Trap as previously described. Upon a trap the message, T = , will be printed on the console device by LBLHEX. A response of, T, will generate a listing of all of the register pairs, the address of the trap, the stack pointer, and the contents of the stack. LBLHEX is then reinitialized and an asterisk (*) is printed. Note, the printout of the TRAP parameters will be initiated by the response, T, a carriage return (CR) is not required.

Any other valid keyboard command may be used, after a trap, to examine, modify, dump, or transfer program control to any other memory address.

DUMP

Dump allows the user to print the contents of any region of memory on the console device. The command format is of the form DMMMM,NNNN(CR), where MMMM is the starting address of the portion of memory to be displayed and NNNN is the end address. If NNNN is less than MMMM, the contents of only

one address (MMMM) will be printed and control will return to the monitor (*). A dump in progress may be interrupted at any time by striking any console keyboard character. An example of the use of this command is given below:

```
*D8ØØØ,8Ø12(CR)
8ØØØ 84 13 12 AB CD 13 Ø5 59 C3 A1 DD Ø1 31 33 3A FA
8Ø1Ø 3B C3 5C
*
```

LOAD

Load is used to sequentially insert code into memory, starting at any location. The command format is, LNNNN AA,BB CC(CR). NNNN is the first address in memory where code AA will be stored. BB will be stored at NNNN+1 and so on. Either a space or a comma may be used as separators between code words. A carriage return (CR) terminates the loading operation. A line feed (LF) will cause the monitor to print the next address and its contents. Then the user, may modify that address by inserting new code and continue the loading operation. If the contents of a location are not to be modified, one need only type a space or a comma and continue on to the next location. Some examples of Load and Dump are shown below:

```
*D8ØØØ,8ØØ9(CR)
8ØØØ AB CD Ø1 32 4Ø 56 8A 34 12 34
*L8ØØØ 2,2B34,,Ø(LF)
8ØØ4 4Ø ,Ø(CR)
*D8ØØØ,8ØØ9(CR)
8ØØØ Ø2 34 Ø1 ØØ 4Ø ØØ 8A 34 12 34
*
```

EXAMINE

Examine provides the user with the ability to display the contents of any memory location. After displaying the contents of a memory location, LBLHEX will enter the LOAD mode and all of the options, as previously described, for the Load command are available to the user.

The command format is ENNNN(CR) where NNNN may be any hex address. For example, a typical use of the Examine command to open a location to examine and load is shown below:

```
*E8B8ØØØ(CR)
8ØØØ Ø2(LF)
8ØØ1 34 AA,1234(CR)
*D8ØØØ,8ØØ2(CR)
8ØØØ Ø2 AA 34
*
```

A convenient use of the examine feature is to display several sequential locations and their contents for searching, checking, or listing purposes. An example is shown below:

```
*E4AC(CR)
Ø4AC Ø4(LF)
Ø4AD DC(LF)
Ø4AE A2(LF)
Ø4AF Ø4(LF)
Ø4BØ CD(CR)
*
```

READ

By means of the Read command, object code may be loaded into memory from any read-in device. The read routines shown in the assembly listing,

Appendix 1, pages 5,19-21 have been used to read object code from various paper tape readers or from telephone lines connected to the LBL computer center.

The command format is RHAAAA(CR), where H signifies that the format of object code is HEX(INTEL**). This format is described in Appendix B. The character, B, in place of the H would indicate that the format is BINARY (LLL*), See Appendix C. AAAA is on offset address, which is added to the load address contained in the input file. If no address is given, a default value of ØØØØH is assumed.

Usually the object code load address is specified on the source file and the offset address is not needed. When object code is to be loaded on EPROMs, see Move and Punch discussions, the offset address is found to be a very useful feature of LBLHEX.

The ability to read BINARY (LLL) formatted tapes is included for the convenience of users of early microprocessor systems which still use this older format.

The preferred format for both input and output object files is the HEX(INTEL) format. It has, in fact, became the defacto standard for industrial, laboratory, and home user systems.

HEX

This command allows the user to write a HEX(INTEL) formatted tape. The PO driver routine, see page 3 of the assembly listing, may be changed to write any other output device.

**Reference to a company or product name does not imply approval or recommendation of the product by the University of California or the U.S. Department of Energy to the exclusion of others that may be suitable.

*LLL - Lawrence Livermore Laboratory of the University of California in Livermore

The command format is HAAAA,BBBB(CR), where AAAA is the lower boundary memory address of the code to be transmitted and BBBB is the upper boundary memory address. The object code from the region specified will be transmitted in a HEX(INTEL) format and an EOF record will terminate the transmission, see Appendix B.

PUNCH

Until recently, most microprocessor systems did not have resident EPROM programmers. Therefore, a means of down-loading object code from a microprocessor system to a remote EPROM programming device is necessary. The Punch command is capable of producing paper tape formatted for the PRO-LOG** EPROM programmer. The command is of the format PNNN,AAA(CR), where NNN is the size of the EPROM. The user is allowed to specify either, 17Ø2 EPROMs with NNN=FF, 27Ø8 EPROMs with NNN=3FF, or 2716 EPROMs with NNN=7FF. AAA is an initial EPROM address, the user may, for example, want to write on only the upper half of a 2716 EPROM. After the Punch command (CR), LBLHEX will return to the command decoder and an asterisk will be printed on the console. The user is then required to specify, by means of the Dump command, the memory locations that are to be loaded on to the EPROM. An example is shown below:

```
*P3FF,3FE
*D43FE,47FF
PUNCH ON?
3FE Ø9
3FF 44
/
ØØØ CD
ØØ1 A3
```

ØØ2 44

ØØ3 C6

ØØ4 Ø1

ØØ5 77

ØØ6 C3

(and so on until)

3FF Ø5

/

The user specified that 27Ø8 EPROMs were to be loaded starting at address 3FEH on the first EPROM. The code from memory address, 43FEH, to memory address, 47FFH, was transferred to paper tape, formatted to load two 27Ø8 EPROMs.

The first two memory locations are to be put on a first 27Ø8 EPROM at locations 3FEH and 3FFH. The remainder of the code specified, was formatted to be loaded onto a second 27Ø8 EPROM, starting address = ØØØH. Note, remember that a Punch output, in progress, may be terminated by striking any console character.

CHECK

Check allows one to load any 8-bit byte of code into any region of memory. After the 8-bit byte is loaded into each memory location, it is read out by LBLHEX and verified. If the memory contents are not equal to the input byte, the address and the contents are printed on the console device. Program control may be transferred to LBLHEX at any time by striking any keyboard character.

The command is of the format CMMMM,NNNN,AA(CR). AA is the 8-bit byte of code to be loaded into memory from address MMMM to address NNNN.

There are many uses of this command. It may be used to clear an area of memory. For example, one may load all ØØ's in RAM memory from 84ØØH to 9ØØØH, as shown below:

```
*C84ØØ,9ØØØ,Ø(CR)
*
```

One may use this command to find the boundaries of a microprocessor system RAM memory, as shown below:

```
*C84ØØ,FFFF,11(CR)
BØØØ FF
BØØ1 FF
BØØ2 FF
BØØ3 FF
BØØ4 FF
BØØ5 FF
*
?
*
```

The microprocessor RAM memory lower boundary address is 84ØØH while the RAM memory upper boundary has been located at address, AFFFH, in the above example. Note that, the printout was terminated by striking any console character. This caused the question mark (?) response and the re-initialization of LBLHEX.

Lastly, this command may be used to check (hence the name) that EPROMs are completely erased.

```
*CDØØØ,D7FF,FF(CR)
*CD8ØØ,DFFF,FF(CR)
```

D9BA EF

*

In the above example, the EPROM memory from DØØØH to D7FFH is completely erased (it contains FF at every location). The 2716 EPROM at D8ØØH has one bad bit at memory location D9BAH. (Put it back under the UV source.)

MOVE

The move command provides a means of moving a block of code from one region of memory to another. This is most useful when loading EPROM's by means of a resident prom programmer.

The format of the command is MAAAA,BBBB,CCCC,(CR) where AAAA to BBBB define the addresses of the section of memory that is to be moved. CCCC is the starting address of section of memory that is to be loaded with the code from address AAAA to address BBBB. NN is the number of times that this move should be repeated. For no NN, the default value is one.

After NN moves, the memory bounded by address AAAA to address BBBB is compared, bit for bit, with the contents of memory starting at address CCCC. Any differences are reported on the console device. The contents of the original memory location and address are printed as well as the new address and its' contents.

As in other LBLHEX commands, an output listing in progress may be terminated at any time by entering any console keyboard character.

USEFUL SUBROUTINES

There are a number of useful subroutines contained in LBLHEX. These may be useful as utility routines for other programs. The list below contains a name (sometimes descriptive), an entry address, and a brief description including affected registers. For further details, please refer to the assembly listing in Appendix A.

NAME	CALL ADDRESS	DESCRIPTION
CI (console input)	4ØH	Waits for a character to be received from the teletype and returns the ASCII character in the A register. Affects only the A register.
CO (console output)	46H	Waits for teletype printer ready and then prints the contents of the C register. No registers are modified.
PRMSG	4CH	Prints a message stored in ASCII code and terminated by a ØØH character. The HL registers points to the first byte of the message string. No registers are modified.
CRLF	4FH	Prints a CR and a LF on the console device. No registers are modified.
SPACE	52H	Prints one space on the console device. C register is used.
CECHO	55H	Get a character from the console device and echoes (prints) it. A register contains character, no other register modified.
HEXNB	58H	Converts a Hex character in C register to 4-bit BINARY nibble in A register. No other registers are modified.
BTHEX	5BH	Converts an 8-bit Byte in C into 2 Hex characters in B and A. A, B, and C registers are used.

NAME	CALL ADDRESS	DESCRIPTION
NBHEX	5EH	Converts a nibble in C to a Hex character in A. No other registers are used.
HEXCO	61H	8-bit Byte in C is transferred as two Hex characters to the console. No registers are modified.
HEX1CO	64H	A space and 2 Hex characters are printed on the console. The Byte in the C register is printed. No registers are modified.
HEX2CO	67H	A space and 4 Hex characters are printed on the console. The two Bytes in BC are printed. Only the A register is modified.
ENDOP	6DH	Will return a carry if a character has been received on the teletype. Uses A only.
CHKAD	3Ø6H	Compare DE to HL. Return with Zero set if DE=HL. Returns with a carry set if DE if greater than HL. No carry and no Zero are returned if HL is less than DE. A is the only register modified.

ACKNOWLEDGEMENTS

LBLHEX has developed from earlier experiences with both versions of ODT (PDP-8[2] and LLL[3]). The LLL version of ODT was an invaluable tool in developing microprocessor systems at LBL for some time. It soon became apparent that any extended assembly code development or debugging would be very tedious using a page-oriented octal code.

In 1976, Gary Smith of LBL authored a very useful Hex monitor program called HEXMON. HEXMON was similar to ODT with the addition of a breakpoint and dump features.

The development of LBLHEX started in late 1976 as an effort to add some more features to HEXMON. A new command language has been evolved for LBLHEX as new features were added in an attempt to keep up with continued developments in microprocessor systems. This process will continue in the foreseeable future.

It should also be acknowledged that many useful features of the code were influenced by the parallel development of a monitor called HDT at LLL by J. M. Spann and B. P. Douros. Another useful monitor program that has been referred to in the course of the development of LBLHEX is the INTEL** 80-10 system monitor program.

In the course of this development, we have been aware of a U.C. Berkeley monitor program called HDT. The most recent version of this page-oriented monitor also has a rudimentary calculator function[5].

Lastly, I want to thank and acknowledge the comments, criticisms, suggestions, and general all-around support of Richard Strudwick and the rest of my associates at LBL.

REFERENCES

1. An Introduction to Microprocessors, 1975 by Adam Osborne and Associates, 2950 Seventh Street, Berkeley, California

2. "Octal Debugging Technique" (ODT), Digital Equipment Corporation**, DEC-08-COCO-D, 1968

3. E.R. Fisher, Octal Debugging Program (ODT-80) for the MSC-80 computer, Lawrence Livermore Laboratory, Rept. UCRL-51694 Rev. 1, 1975

4. M.D. Maples and E.R. Fisher, Real-Time Microcomputer Applications Using LLL BASIC, Computer, Sept. 1977, pp. 14-21

5. RUTS-8Ø A Research, Utility and Training System, Kim Rubin, Physics Department, U.C. Berkeley

APPENDIX A: SOURCE LISTING OF LBLHEX

```
8080 MACRO ASSEMBLER, VER 2.0  ERRORS = 0 PAGE 1

;.....................................................;
;                                                     ;
;       LBLHEX   2/27/78   J.KATZ X5636              ;
;       A HEX MONITOR PROGRAM FOR 8080 CODE-          ;
;       COMPATIBLE MICROPROCESSOR SYSTEMS             ;
;                                                     ;
;     COMMAND          TTY INPUT FORMAT               ;
;                                                     ;
;     BREAKPOINT       BNNNN,MM,LL (CR)               ;
;        NNNN= BREAKPOINT ADDRESS,MM= NUMBER OF BYTES ;
;        IN THIS INSTRUCTION,LL=NUMBER OF PASSES      ;
;        BEFORE BREAK. DEFAULT IS A BREAK ON 1ST PASS.;
;     TRAP     EXECUTION OF THE CODE OFFH,WILL PRINT  ;
;        (T=) , A TTY INPUT OF T WILL CAUSE A         ;
;        PRINT OF ALL THE REGISTER PAIRS              ;
;     CHECK MEMORY     CMMMM,NNNN,AA (CR)             ;
;        LOADS AA INTO MEMORY(MMMM TO NNNN) AND VERIFIES;
;     DUMP MEMORY      DMMMM,NNNN (CR)                ;
;     EXAMINE          ENNNN (CR)                     ;
;        AFTER EXAMINE , ADDRESS IS OPEN TO LOAD      ;
;     GO (EXECUTE)     GNNNN (CR)                     ;
;     LOAD MEMORY      LNNNN,AA BB CC (CR)            ;
;        , = OPEN NEXT ADDRESS FOR LOAD               ;
;        SPACE=OPEN NEXT ADDRESS FOR LOAD             ;
;        CR = TERMINATE LOAD                          ;
;        LF = NEW LINE,PRINT ADD,CONTENTS,CONTINUE    ;
;     PUNCH            PNNN,AAA (CR)                  ;
;        PUNCH A HEX ASCII TAPE TO LOAD A PROM ON     ;
;        THE PRO-LOG PROM PROGRAMMER,  NNN=SIZE OF    ;
;        THE PROM(FF,3FF,7FF=256,1024,2048)           ;
;        AAA=INITIAL PROM ADDRESS(MODULO NNN),USE DUMP;
;        TO SPECIFY CODE FOR PROM LOAD TAPE.          ;
;     MOVE             MAAAA,BBBB,CCCC,NN (CR)        ;
;        MOVE CODE FROM ADDRESSES AAAA-BBBB TO        ;
;        CCCC(NN TIMES). A CHECK IS MADE TO VERIFY    ;
;        THE MOVE. DEFAULT(NO NN) IS ONE MOVE.        ;
;     READ             RHAAAA (CR)                    ;
;        READ A HEX(INTEL) FORMAT PAPER TAPE.R IN     ;
;        PLACE OF H SPECIFIES A BINARY(LLL) FORMAT.   ;
;        AAAA=OFFSET ADDRESS WHICH IS ADDED TO PAPER  ;
;        TAPE ADDRESS.DEFAULT OFFSET=0.               ;
;     HEX ASCII        HAAAA,BBBB (CR)                ;
;        PUNCH A HEX ASCII LOAD TAPE,AAAA TO BBBB     ;
;                                                     ;
;    FOR AN INPUT ERROR, THE RESPONSE IS ? (TRY AGAIN);
;                                                     ;
;.....................................................;
;
;WHEN ENTERING A BYTE,ONLY INFORMATION IMMEDIATELY PROCEEDING A
;TERMINATOR IS HELD. ANY TYPING ERROR MAY BE CORRECTED BY CONTINUED
;TYPING. A NON-HEX CHAR. IN A NUMERIC FIELD WILL CAUSE AN
```

```
8080 MACRO ASSEMBLER, VER 2.0  ERRORS = 0 PAGE 2

                        ;ERROR RETURN.
                        ;THE SOURCE CODE IS STORED IN LIBRARY=MICRO,SUBSET=LBLHEX, ON LBL PSS
                        ;
000002                  TTY     EQU  2          ;TTY KEYBOARD IN AND OUT PORT
000003                  FLAG    EQU  3          ;UART AND USART STATUS AND CONTROL PORT
000002                  TTYDA   EQU  2          ;STATUS BIT 1,TTY HAS A CHAR.
000001                  TTYTR   EQU  1          ;STATUS BIT 0,TTY READY TO SEND AGAIN
000025                  CMD     EQU 025H        ;USART COMMAND INPUT
0000CE                  MODE    EQU 0CEH        ;USART INITIALIZATION OF MODE
                                                ;FOR 80-10 SYSTEMS USE MODE=CFH
0083D0                  STACK   EQU 83D0H       ;INITIAL STACK POINTER
0083D2                  PFLAG   EQU STACK+2     ;PUNCH,MOVE,BREAK  FLAG
0083D3                  PROM    EQU STACK+3     ;PROM SIZE
0083D5                  PADDR   EQU STACK+5     ;PROM ADDRESS
0083D7                  MOVAD   EQU STACK+7     ;BREAK, MOVE ADDRESS STORAGE
0083D9                  SAVE1   EQU STACK+9     ;BREAKPOINT BYTE REFUGE, 1ST BYTE
0083DA                  SAVE2   EQU SAVE1+1     ;      2ND BYTE
0083DB                  SAVE3   EQU SAVE1+2     ;      3RD BYTE
0083DC                  SAVE4   EQU SAVE1+3     ; RET CODE
0083DD                  NB      EQU SAVE1+4     ; NO. OF BYTES
0083DE                  BCTR    EQU SAVE1+5     ;BREAKPOINT TIMES COUNTER
0083DF                  RAMJP   EQU SAVE1+6     ;RESTART 5,ENTRY JUMP COMMAND
0083E2                  TEMP1   EQU SAVE1+9     ;TEMP. ADD. STORE 1
0083E4                  TEMP2   EQU SAVE1+11    ;TEMP. ADD. STORE 2
0083E6                  TEMP3   EQU SAVE1+13    ;TEMP. ADD. STORE 3
0083E8                  EXTRA   EQU SAVE1+15    ;EXTRA SCRATCH RAM
                        ;
000000                          ORG  0H
000000  0FB             S000→    EI             ;ENABLE INTERRUPT
                        ;INTERRUPT TO 0=LBLHEX, 1=BASIC(CLEAR MEMORY), 2=BASIC(SAVE MEMORY),
                        ;             3=BASIC(RUN), 4=BASIC(I/O INITIALIZE), 5=JUMP TO RAM,
                        ;             6=LBLHEX(BREAKPOINT), 7=TRAP(LBLHEX)
                        ;
000001  000                     NOP             ;ON RESET/INTERRUPT FRONT PANEL SWITCH DIRECTS
000002  031 0D0 083             LXI  SP,STACK   ;SP TO END OF 1KSCRATCH RAM AT 8000H.
000005  0C3 070 000             JMP  RES0       ;RESET = 0
000008  0F3                     DI
000009  031 0D0 083             LXI  SP,STACK
00000C  0C3 076 000             JMP RES1        ;RESET = 1
00000F  000                     NOP
000010  0F3                     DI
000011  031 0D0 083             LXI  SP,STACK
000014  0C3 07C 000             JMP RES2    ;RESET = 2
000017  000                     NOP
000018  0F3                     DI
000019  031 0D0 083             LXI  SP,STACK
00001C  0C3 082 000             JMP RES3        ;RESET =3
00001F  000                     NOP
000020  0F3                     DI      ;RESET=4
000021  031 0D0 083             LXI SP,STACK
000024  0C3 088 000             JMP RES4
000027  000                     NOP
```

```
8080 MACRO ASSEMBLER, VER 2.0  ERRORS = 0 PAGE 3

000028  0F3                     DI          ;RESET=5
000029  0C3 0DF 083             JMP RAMJP       JUMP TO RAM INTERRUPT ROUTINE
00002C  000                     NOP
00002D  000                     NOP
00002E  000                     NOP
00002F  000                     NOP
000030  0F3                     DI          ;RESET=6
000031  0C3 013 004             JMP HEXBR   ;LBLHEX BREAKPOINT ROUTINE
000034  000                     NOP
000035  000                     NOP
000036  000                     NOP
000037  000                     NOP
000038  0F3                     DI          ;RESET=7
000039  0C3 0B8 004             JMP TRAP        ;BREAKPOINT TRAP ROUTINE
000040                      ORG 5000+40H
                        ;JUMP TABLE
                        ;USE THIS TABLE AS AN ENTRY TO DESIRED ROUTINES
000040  0C3 023 007 CI→     JMP TTYIN       ;CONSOLE INPUT
000043  0C3 023 007 RI→     JMP TTYIN       ;USE TTY FOR READER IN
000046  0C3 02D 007 CO→     JMP TTYOT       ;CONSOLE OUTPUT
000049  0C3 02D 007 PO→     JMP TTYOT       ;PUNCH OUT
00004C  0C3 0D0 004         JMP PRMSG       ;PRINT MESSAGE
00004F  0C3 0E6 004         JMP CRLF        ;PRINT CR + LF
000052  0C3 097 004         JMP SPACE       ;PRINT A SPACE
000055  0C3 0F3 004         JMP CECHO       ;CHAR WITH ECHO FROM CONSOLE
000058  0C3 003 005         JMP HEXNB       ;CONV HEX TO NIBBLE
00005B  0C3 016 005         JMP BTHEX       ;BYTE TO 2 HEX
00005E  0C3 036 005         JMP NBHEX       ;NIBBLE TO HEX
000061  0C3 041 005         JMP HEXCO       ;HEX TO CONSOLE DEVICE
000064  0C3 051 005         JMP HX1CO       ;SPACE,HEX TO CONS
000067  0C3 05B 005         JMP HX2CO       ;SPACE,2 HEX TO CONS
00006A  0C3 0FC 005         JMP HEXTO       ;HEX RECORD TO MEMORY
00006D  0C3 00F 003         JMP ENDOP       ;RETURN WITH CARRY IF TTY
                        ;RESET ROUTINES TO RESET USART AND JUMP TO SELECTED CODE
000070  0CD 08E 000 RES0→   CALL INIT
000073  0C3 0A6 000         JMP START       ;RESET=0, GO MONITOR
000076  0CD 08E 000 RES1→   CALL INIT
000079  0C3 000 047         JMP 4700H       ;RESET=1,CLEAR BASIC MEMORY
00007C  0CD 08E 000 RES2→   CALL INIT
00007F  0C3 013 047         JMP 4713H       ;RESET=2,SAVE BASIC MEMORY
000082  0CD 08E 000 RES3→   CALL INIT
000085  0C3 0DC 057         JMP 57DCH       ;RESET=3,RUN BASIC
000088  0CD 08E 000 RES4→   CALL INIT    ;RESET=4,INIIT. I/O,CLEAR BASIC
00008B  0C3 0D9 057         JMP 57D9H
                        ;INIT. USART-2 STOP BITS,PARITY DISABLED,8 BIT CHAR,BAUD RATE
                        ;   OF 16X(110BAUD), COMMAND-NO HUNT,SEND
                        ;   RECEIVE ENABLED,DATA TERM. READY LOW,TRANS ENABLED.
00008E  0AF             INIT→   XRA A
00008F  0D3 003             OUT FLAG    ; OUTPUT 3 ZEROS TO
000091  0D3 003             OUT FLAG    ; ENABLE USART
000093  0D3 003             OUT FLAG    ;   RESET
```

```
8080 MACRO ASSEMBLER, VER 2.0  ERRORS = 0 PAGE 4

000095   03E 0FF            MVI A,0FFH  ;  INTERNAL RESET OF
000097   0D3 003            OUT FLAG    ;  THE USART
000099   03E 0CE            MVI A,MODE     ;GET USART INIT. MODE
00009B   0D3 003            OUT FLAG       ;LOAD MODE TO USART
00009D   03E 025            MVI A,CMD      ;USART COMMAND WORD
00009F   0D3 003            OUT FLAG       ;SEND IT TO USART
0000A1   03E 0FF            MVI A,377Q    ;RUBOUT
0000A3   0D3 002            OUT TTY        ; SEND IT TO RESYNC TTY
0000A5   0C9                RET
0000A6   0AF          START→ XRA A                  ;SET A=0
0000A7   032 0D2 083        STA PFLAG                 ;CLEAR PFLAG
0000AA   0F3                DI
0000AB   031 0D0 083        LXI  SP,STACK
0000AE   0CD 0E6 004 PUN1→  CALL CRLF
0000B1   00E 02A            MVI  C,'*'  ; PRINT AN ASTERISK
0000B3   0CD 046 000        CALL  CO
0000B6   0CD 0FD 004 STRD→  CALL NECHO     ;READ COMMAND.
0000B9   0FE 042            CPI  'B'      ;SENSE B.
0000BB   0CA 054 001        JZ   BKPNT
0000BE   0FE 043            CPI  'C'       ; SENSE C
0000C0   0CA 0FB 000        JZ  CKMEM
0000C3   0FE 044            CPI  'D'      ;SENSE D
0000C5   0CA 0EE 001        JZ   DUMP
0000C8   0FE 045            CPI  'E'      ;SENSE E.
0000CA   0CA 0A6 001        JZ   EXMNE
0000CD   0FE 047            CPI  'G'      ;SENSE G.
0000CF   0CA 0AF 001        JZ   GO
0000D2   0FE 048            CPI 'H'       ;SENSE H
0000D4   0CA 042 001        JZ  HWRITE     ;GO PUNCH HEX PAPER TAPE
0000D7   0FE 04C            CPI  'L'      ;SENSE L.
0000D9   0CA 0B3 001        JZ   LOAD
0000DC   0FE 04D            CPI 'M'        ; SENSE M
0000DE   0CA 079 002        JZ  MOVE
0000E1   0FE 052            CPI 'R'        ; SENSE R
0000E3   0CA 023 001        JZ  READER
0000E6   0FE 050            CPI 'P'     ; SENSE P
0000E8   0CA 02F 003        JZ  PUNCH
0000EB   0FE 054            CPI  'T'    ;SENSE T
0000ED   0CA 027 004        JZ  BRPRT  ;GO PRINT REGISTER PAIRS
0000F0   031 0D0 083 ERROR→ LXI  SP,STACK
0000F3   00E 0BF            MVI  C,277Q         ;PRINT A QUESTION MARK
0000F5   0CD 046 000        CALL  CO
0000F8   0C3 0A6 000        JMP  START     ;ERROR RETURN
                        ;
                        ; CKMEM WILL LOAD CODE  AA INTO MEMORY FROM MMMM TO NNNN. IT WILL
                        ; READ AFTER EACH LOAD TO CONFIRM LOAD. ON ERRORS A MESSAGE WILL BE
                        ; PRINTED. ANY CONSOLE INPUT CHARACTER WILL ABORT ROUTINE.
0000FB   0CD 0D9 001 CKMEM→ CALL GETAD    ;GET START AND END ADDRESSES
0000FE   0CD 049 003        CALL ADRIN    ;LOAD BYTE IN L
000101   07D                MOV A,L
000102   032 0D2 083        STA PFLAG    ;SAVE BYTE
```

```
8080 MACRO ASSEMBLER, VER 2.0  ERRORS = 0 PAGE 5

000105   0CD 0E6 001          CALL GETA1  ;DE=END ADD.,HL=START ADD.
000108   0CD 00F 003 CHKM1→   CALL ENDOP    ;ABORT?
00010B   0DA 0A6 000          JC START     ;YES
00010E   03A 0D2 083          LDA PFLAG    ;GET BYTE TO BE LOADED
000111   077                  MOV M,A      ;LOAD NEW ADD. WITH BYTE
000112   04F                  MOV C,A      ;BYTE IN C
000113   0CD 00C 003          CALL MVCMP   ;CHECK IF CODE IN ADD.=C
000116   0C4 0F5 002          CNZ MVEPR    ;GO REPORT ERROR
000119   0CD 006 003          CALL CHKAD   ;ARE WE DONE?
00011C   0CA 0A6 000          JZ  START    ;DONE
00011F   023                  INX H        ;+1 TO  ADDRESS
000120   0C3 008 001          JMP CHKM1    ; CONTINUE

                      ;
                      ;READER WILL READ EITHER BIN OR HEX PAPER TAPE PLUS AN OFFSET
                      ;ADDRESS TO MEMORY.DEFAULT OFFSET =0.
                      ;CHECKSUM ERRORS ARE REPORTED, COUNTED, AND STORED IN NB.
000123   0CD 0FD 004 READER→  CALL NECHO              ;GET B OR H
000126   032 0D2 083          STA PFLAG              ;STORE B OR H
000129   0CD 049 003          CALL ADRIN             ;H,L =OFFSET ADDRESS
00012C   022 0D7 083          SHLD MOVAD             ; SAVE OFFSET ADD.
00012F   0CD 0E6 004          CALL CRLF
000132   03A 0D2 083          LDA PFLAG
000135   0FE 042              CPI 'B'              ;IS IT A BINARY TAPE?
000137   0CA 0B6 006          JZ BINTP
00013A   0FE 048              CPI 'H'              ;IS IT A HEX TAPE?
00013C   0CA 0FC 005          JZ HEXTO
00013F   0C3 0F0 000          JMP ERROR        ;ERROR IF IT IS NOT B OR H

                      ;
                      ;HWRITE WILL PUNCH AN INTEL FORMATTED PAPER TAPE(HEX) WITH CODE
                      ;FROM ADDRESS AAAA TO BBBB.
000142   0CD 0D9 001 HWRITE→   CALL GETAD     ;GET AAAA AND BBBB(TEMP1,TEMP2)
000145   0CD 020 003          CALL PSTRT     ;TURN PUNCH ON,ETC.
000148   0CD 07C 005          CALL WRITE   ;GO PUNCH TAPE
00014B   0CD 0E3 005          CALL WTEOF     ;PUNCH EOF RECORD
00014E   0CD 0BC 003          CALL LDRTR     ;PUNCH TRAILER
000151   0C3 0A6 000          JMP START

                      ;
                      ;A RESTART INSTRUCTION IS INSERTED AT A BREAKPOINT WHEN EXECUTED,
                      ;FOR THE NTH TIME, WILL CALL A REGISTER DISPLAY ROUTINE. THE ORIGINAL
                      ;BREAKPOINT BYTES ARE RESTORED. DEFAULT IS A BREAK AT 1ST PASS.
000154   032 0D2 083 BKPNT→   STA PFLAG     ;SET BREAK SWITCH TO B
000157   0AF                  XRA A          ;INITIALIZE STORAGE
000158   032 0DA 083          STA SAVE2
00015B   032 0DB 083          STA SAVE3
00015E   03C                  INR A          ;+1 TO A
00015F   032 0DE 083          STA BCTR     ;SET FOR DEFAULT ONE PASS
000162   032 0DD 083          STA NB       ;DEFAULT=ONE BYTE
000165   03E 0C9              MVI A,0C9H     ; RETURN CODE
000167   032 0DC 083          STA SAVE4
00016A   0CD 049 003          CALL ADRIN     ;GET BREAKPOINT ADDRESS.
00016D   022 0D7 083          SHLD MOVAD     ;SAVE BRKPNT ADD.
```

```
8080 MACRO ASSEMBLER, VER 2.0  ERRORS = 0 PAGE 6

000170   07E                    MOV  A,M      ;SAVE BREAKPOINT BYTE.
000171   032 0D9 083            STA  SAVE1    ;SAVE 1ST BYTE
000174   03E 0F7                MVI A,367Q    ;RESTART INSTRUCTION
000176   077                    MOV M,A       ;PUT IT THERE
000177   01D                    DCR E         ;IS E=1(CR) ?
000178   0CA 0A6 000            JZ START    ;DONE, TAKE DEFAULT
00017B   0CD 049 003            CALL ADRIN      ;TO READ IN NB
00017E   07D                    MOV A,L         ;A=NO. OF BYTES
00017F   032 0DD 083            STA  NB         ;STORE NO. OF BYTES
000182   02A 0D7 083            LHLD MOVAD      ;GET BREAKPT ADD
000185   05F                    MOV E,A         ;INIT CTR
000186   0CD 09C 001            CALL BKP2       ;DEC. AND CHECK CTR.
000189   032 0DA 083            STA SAVE2       ;SAVE 2ND BYTE
00018C   0CD 09C 001            CALL BKP2       ;DEC. AND CHECK CTR.
00018F   032 0DB 083            STA SAVE3       ;SAVE 3RD BYTE
000192   0CD 049 003 BKP1→      CALL ADRIN      ;GET NO. OF TIMES
000195   07D                    MOV A,L         ;A=NO. OF TIMES
000196   032 0DE 083            STA BCTR        ;LOAD BREAK CTR
000199   0C3 0A6 000            JMP START          ;GO GET NEXT COMMAND
00019C   01D            BKP2→  DCR E
00019D   0CA 092 001            JZ BKP1       ;DONE
0001A0   023                    INX H
0001A1   046                    MOV B,M    ;CODE IN B
0001A2   0AF                    XRA A
0001A3   077                    MOV M,A  ;SET LOCATION TO NOP
0001A4   078                    MOV A,B  ;CODE TO BE SAVED IN A
0001A5   0C9                    RET

0001A6   0CD 049 003 EXMNE→     CALL ADRIN    ;EXAMINE ADDRESS.
0001A9   0CD 0F5 002            CALL MVEPR  ;PRINT CRLF,SP,ADD.,SP,CODE,SP
0001AC   0C3 0B6 001            JMP BYTE    ;READY TO LOAD

0001AF   0CD 049 003 GO→        CALL ADRIN    ;GET ADDRESS.
0001B2   0E9                    PCHL          ;GO THERE.

                        ;
                        ;LOAD MEMORY. EACH INPUT BYTE IS TERMINATED BY EITHER A SPACE OR A
                        ;COMMA. A LINE FEED WILL CAUSE A CRLF, PRINT ADDRESS AND CONTENTS
                        ;CR WILL CAUSE AN EXIT FROM THE LOAD ROUTINE.
0001B3   0CD 049 003 LOAD→      CALL ADRIN    ;GET INITIAL ADDRESS.
0001B6   022 0E2 083 BYTE→      SHLD TEMP1  ;SAVE ADDRESS
0001B9   0CD 049 003            CALL ADRIN  ;GET BYTE INTO L
0001BC   07D                    MOV  A,L    ;BYTE INTO A
0001BD   02A 0E2 083            LHLD TEMP1     ;GET ADDRESS INTO HL
0001C0   015                    DCR D         ;TEST NO. OF CHARS?
0001C1   0CA 0C5 001            JZ LOAD1    ;NO CHARACTER, CONTINUE
0001C4   077                    MOV M,A       ;BYTE INTO MEMORY
0001C5   023            LOAD1→  INX H         ;INC. LOAD ADDRESS
0001C6   01D                    DCR E       ;TEST FOR 1(CR)
0001C7   0CA 0A6 000            JZ START    ;EXIT LOAD
0001CA   01D                    DCR  E      ;TEST FOR 2(LF)
0001CB   0C2 0B6 001            JNZ BYTE    ;NOT A LF,CONTINUE LOAD
```

```
8080 MACRO ASSEMBLER, VER 2.0  ERRORS = 0 PAGE 7


0001CE   0CE 00D            MVI C,0DH
0001D0   0CD 046 000        CALL CO      ;PRINT A CR
0001D3   0CD 0F8 002        CALL MVEP1   ;PRINT A SP,ADD.,SP,CODE,SP
0001D6   0C3 0B6 001        JMP BYTE     ;GO GET NEXT BYTE

0001D9   0CD 049 003 GETAD- CALL ADRIN   ;GET START ADDRESS
0001DC   022 0E2 083        SHLD TEMP1   ;STORE IN TEMP1
0001DF   0CD 049 003        CALL ADRIN   ;GET END ADDRESS
0001E2   022 0E4 083        SHLD TEMP2   ;STORE IN TEMP2
0001E5   0C9                RET
0001E6   02A 0E4 083 GETA1- LHLD TEMP2
0001E9   0EB                XCHG
0001EA   02A 0E2 083        LHLD TEMP1   ;DE=END ADD ,HL=START ADD
0001ED   0C9                RET

                        ;DUMP MEMORY  BOUNDARY ADDRESSES ARE SPECIFIED BY
                        ;REGISTER PAIRS H AND D
0001EE   0CD 0D9 001 DUMP- CALL GETAD    ;GET START AND END ADDRESSFS
0001F1   0CD 0E6 001        CALL GETA1   ;DE=END ADD ,HL=START ADD
0001F4   03A 0D2 083        LDA PFLAG
0001F7   0FE 050            CPI 'P'      ;IS PUNCH ON?
0001F9   0C2 044 002        JNZ NEWLN    ;NO GO DUMP
                        ; PUNCH HEX ASCII TAPE FROM HERE
0001FC   0E5                PUSH H       ;STORE INITIAL ADDRESS ON STACK
0001FD   0CD 020 003        CALL PSTRT   ;TURN ON PUNCH,ETC
000200   02A 0D5 083 PUN4- LHLD PADDR    ;LOAD H+L WITH PROM ADDRESS
000203   03A 0D4 083        LDA PROM+1   ; GET MS PART OF PROM SIZE
000206   0FE 000            CPI 0        ;IS IT 0?
000208   0CA 013 002        JZ ADD2      ;YES, 2 CHAR ADDRESS
00020B   04C                MOV C,H      ; MS TO C
00020C   0CD 036 005        CALL NBHEX   ;CONVERT TO HEX ACSII
00020F   04F                MOV C,A
000210   0CD 049 000        CALL  PO     ; PUNCH IT
000213   04D          ADD2- MOV  C,L     ;LS PART OF PROM ADDRESS
000214   0CD 02A 005        CALL PTHEX   ; PUNCH 2 CHAR
000217   00E 020            MVI C,' '        ; SPACE
000219   0CD 049 000        CALL  PO
00021C   0E1                POP H        ; GET DATA ADDRESS
00021D   0E5                PUSH H       ; STORE ADDRESS
00021E   04E                MOV C,M      ; GET DATA BYTE
00021F   0CD 02A 005        CALL PTHEX   ;PUNCH 2 DATA CHAR.
000222   0CD 0D8 005        CALL PCRLF   ;PUNCH CR/LF
                        ; CHECK FOR END OF PROM
000225   02A 0D5 083        LHLD PADDR   ;LOAD H+L WITH PROM ADDRESS
000228   03A 0D3 083        LDA PROM     ;LS PROM SIZE
00022B   0BD                CMP L
00022C   0C2 03C 002        JNZ PUN3     ; NOT END OF PROM
00022F   03A 0D4 083        LDA PROM+1   ;MS PART OF PROM SIZE
000232   0BC                CMP H
000233   0C2 03C 002        JNZ PUN3     ;NOT END OF PROM
000236   0CD 014 003        CALL PELTR   ;END OF PROM, PUNCH LDR/TR
```

```
8080 MACRO ASSEMBLER, VER 2.0  ERRORS = 0 PAGE 8

000239  021 0FF 0FF          LXI H,0FFFFH    ;SET HL=-1
00023C  023            PUN3→ INX H           ;INCREMENT PROM ADDRESS
00023D  022 0D5 083          SHLD PADDR      ;STORE PROM ADDRESS
000240  0E1                  POP H           ;GET DATA ADDRESS
000241  0C3 04B 002          JMP DOUT2
000244  0CD 0AD 003 NEWLN→   CALL ADOUT       ;PRINT ADDRESS
000247  04E            DOUT→ MOV C,M         ;GET DATA BYTE
000248  0CD 041 005          CALL HEXCO      ;PRINT BYTE
00024B  0CD 00F 003 DOUT2→   CALL ENDOP      ;RETURN CARRY IF TTY?
00024E  0DA 06E 002          JC ENDDP        ;YES TERMINATE DUMP
000251  023                  INX H           ;INCREMENT ADDRESS
000252  0CD 006 003          CALL CHKAD      ;SET CARRY IF HL GT DE
000255  0DA 06E 002          JC ENDDP        ;YES, LAST ADDRESS DONE
000258  0E5                  PUSH H          ;STORE ADDRESS
000259  03A 0D2 083          LDA PFLAG
00025C  0FE 050              CPI 'P'         ;IS PUNCH ON?
00025E  0CA 000 002          JZ PUN4         ;YES, CONTINUE PUNCHING
000261  0E1                  POP H           ;KEEP PLACE ON STACK
000262  07D                  MOV A,L         ;SENSE END OF LINE
000263  0E6 00F              ANI 0FH         ;MASK
000265  0CA 044 002          JZ NEWLN        ;IF END, BEGIN NEXT LINE
000268  0CD 097 004          CALL SPACE      ;PRINT A SPACE
00026B  0C3 047 002          JMP DOUT        ;LOOP
00026E  03A 0D2 083 ENDDP→   LDA PFLAG
000271  0FE 050              CPI 'P'         ;IS PUNCH ON?
000273  0CC 014 003          CZ PELTR        ;YES GO FINISH PUNCH
000276  0C3 0A6 000          JMP START       ;ALL DONE

                     ;MOVE CODE FROM AAAA-BBBB TO CCCC(NN TIMES)
000279  0CD 0D9 001 MOVE→    CALL GETAD      ;GET START AND END ADDRESSES
00027C  0CD 049 003          CALL ADRIN      ;GET MOVE START ADDRESS
00027F  022 0E6 083          SHLD TEMP3      ;ALSO STORE IN TEMP3
000282  03E 001              MVI A,1         ;E=1(CR)
000284  0BB                  CMP E           ;TEST FOR CR
000285  0C2 08C 002          JNZ MOVE1       ;NOT A DEFAULT
000288  06F                  MOV L,A         ;SETL=1,DEFAULT NO. OF TIMES
000289  0C3 08F 002          JMP MOVE2
00028C  0CD 049 003 MOVE1→   CALL ADRIN      ;GET NO. OF TIMES
00028F  022 0D3 083 MOVE2→   SHLD PROM       ;LS PART IN L=PROM,H=MS PART IN PROM+1
000292  02A 0E6 083 MOVE3→   LHLD TEMP3      ;GET MOVE ADDRESS
000295  022 0D7 083          SHLD MOVAD      ;STORE IN MOVAD
000298  0CD 0E6 001          CALL GETA1      ;DE=END ADD.,HL=START ADD.
00029B  013                  INX D           ;+1 TO END ADDRESS
00029C  0CD 006 003 MOVE4 -  CALL CHKAD      ;ZERO IF START.EQ.END ADDRESS+1
00029F  0CA 0B1 002          JZ MOVE5        ;FINISHED MOVE,CHECK NO. OF TIMES
0002A2  04E                  MOV C,M    ;GET BYTE
0002A3  0E5                  PUSH H     ;SAVE BYTE ADD.
0002A4  02A 0D7 083          LHLD MOVAD      ;GET CURRENT MOVE ADD.
0002A7  071                  MOV M,C    ;PUT BYTE INTO NEW HOME
0002A8  023                  INX H           ;INC. MOVE ADDRESSS
0002A9  022 0D7 083          SHLD MOVAD      ;PUT IT AWAY
```

```
8080 MACRO ASSEMBLER, VER 2.0  ERRORS = 0 PAGE 9


0002AC   0E1              POP H   ;GET CURRENT BYTE ADDRESS
0002AD   023              INX H
0002AE   0C3 9C 002       JMP MOVE4   ;CONTINUE MOVE
0002B1   03A 0D3 083 MOVE5→ LDA PROM   ;GET NO. OF TIMES
0002B4   03D              DCR A
0002B5   0CA 0BE 002      JZ ENDMV    ;ALL MOVES DONE
0002B8   032 0D3 083      STA PROM    ;STORE NO. OF TIMES
0002BB   0C3 092 002      JMP MOVE3   ;GO DO ANOTHER MOVE
                     ;   CHECK THAT OLD CODE=NEW CODE,  AFTER NN MOVES
0002BE   0CD 0E6 001 ENDMV→ CALL GETA1   ;DE=END ADD,HL=START ADD.
0002C1   013              INX D       ;+1 TO END ADDRESS
0002C2   0CD 00F 003 CHKMV→ CALL ENDOP     ;CARRY IF TTY?
0002C5   0DA 0A6 000      JC START   ;ABORT
0002C8   04E              MOV C,M       ;GET OLD BYTE
0002C9   023              INX H         ;INC. OLD ADDRESS
0002CA   022 0E2 083      SHLD TEMP1    ;STORE IT
0002CD   02A 0E6 083      LHLD TEMP3    ;GET MOVE ADD.
0002D0   0CD 00C 003      CALL MVCMP    ;DOES OLD=NEW?
0002D3   023              INX H
0002D4   022 0E6 083      SHLD TEMP3   ;STORE MOVE ADD.
0002D7   0C4 0E6 002      CNZ MVERR     ;GO REPORT ERROR
0002DA   02A 0E2 083      LHLD TEMP1    ;GET NEXT ADD.
0002DD   0CD 006 003      CALL CHKAD    ;ARE WE DONE?
0002E0   0C2 0C2 002      JNZ CHKMV    ;NO CONTINUE
0002E3   0C3 0A6 000      JMP START   ; DONE
0002E6   02A 0E2 083 MVERR→ LHLD TEMP1   ;OLD ADD+1
0002E9   0CD 0F4 002      CALL MVEP2   ;DCX H,PRINT CRLF,ADD,CONTENTS
0002EC   02A 0E6 083      LHLD TEMP3   ;GET MOVE ADD+1
0002EF   02B              DCX H
0002F0   0CD 0F8 002      CALL MVEP1    ;PRINT ADD, CODE
0002F3   0C9              RET       ;CONTINUE
                     ;
                     ;MOVE ERROR PRINT (SP,ADD.SP,CODE,SP) HL=ADDRESS
0002F4   02B         MVEP2→ DCX H
0002F5   0CD 0E6 004 MVEPR→ CALL CRLF
0002F8   0CD 097 004 MVEP1→ CALL SPACE
0002FB   0CD 0B0 003      CALL RPOUT
0002FE   04E              MOV C,M   ;GET BYTE
0002FF   0CD 041 005      CALL HEXCO      ;PRINT BYTE
000302   0CD 097 004      CALL SPACE
000305   0C9              RET
                     ; DE=END ADD.,HL=CURRENT ADD., RET WITH ZERO IF DE=HL
                     ; RETURN WITH CARRY IF HL.GT.DE
000306   07A         CHKAD→ MOV A,D    ;GET MS ADD. PART
000307   0BC              CMP H       ;CHECK MS PART,H.GT.D SET CARRY
000308   0C0              RNZ       ; THEY ARE NOT =
000309   07B              MOV A,E    ;GET LS PART
00030A   0BD              CMP L      ;CHECK LS PART
00030B   0C9              RET
                     ;HL = NEW CODE ADDRESS,C= OLD CODE, RETURN ZERO IF THEY ARE =
00030C   07E         MVCMP→ MOV A,M     ;GET NEW BYTE
```

```
8080 MACRO ASSEMBLER, VER 2.0  ERRORS = 0 PAGE 10

 00030D   0B9                    CMP C       ;C =OLD CODE
 00030E   0C9                    RET
                          ;ENDOP LOOKS FOR CODE FROM TTY= CARRY
 00030F   0DB 003         ENDOP→ IN FLAG     ; ANYTHING FROM TTY?
 000311   01F                    RAR
 000312   01F                    RAR   ; PUT UP CARRY IF ANYTHING
 000313   0C9                    RET
                          ;
                          ; END  OF PROM, PUNCH / AND LEADER / TRAILER
 000314   00E 02F         PELTR→ MVI C,'/'
 000316   0CD 049 000           CALL PO       ; PUNCH /
 000319   0CD 0D8 005           CALL PCRLF
 00031C   0CD 0BC 003           CALL LDRTR    ;PUNCH LEADER/TRAILER
 00031F   0C9                   RET
                          ;
 000320   00E 08A         PSTRT→  MVI C,8AH
 000322   0CD 046 000            CALL CO       ;PRINT A LINE FEED
 000325   001 007 004            LXI B,PMESS   ;PUNCH ON MESSAGE
 000328   0CD 0D0 004            CALL PRMSG
 00032B   0CD 0BC 003            CALL LDRTR    ;PUNCH LEADER/TRAILER
 00032E   0C9                    RET
                          ;
                          ; SUBROUTINE PUNCH SETS A FLAG SO THAT DUMP WILL CAUSE A
                          ; PAPER TAPE TO BE PUNCHED. FORMAT IS  (ADDRESS,SPACE,
                          ; DATA,CR,LF),( ),( ), ETC. / .
                          ; ADDRESS= 2 OR 3 HEX ASCII CHARS. DATA= 2 HEX ASCII CHARS.
                          ; SIZE OF PROM IS DECODED TO DETERMINE WHETHER 2 OR 3
                          ; CHAR. ADDRESS IS NEEDED.
 00032F   032 0D2 083 PUNCH→ STA PFLAG    ;PUT P INTO PUNCH FLAG
 000332   0CD 049 003        CALL ADRIN    ; GET PROM SIZE
 000335   0E5                PUSH H   ;STORE PROM SIZE ON STACK
 000336   022 0D3 083        SHLD PROM     ;STORE PROM SIZE(MS AT PROM+1=H)
 000339   0CD 049 003        CALL ADRIN    ;GET PROM STARTING ADDRESS
 00033C   022 0D5 083        SHLD PADDR    ;STORE(PADDR=LS=L)
 00033F   0D1                POP D     ;PROM SIZE IN DE, S.A. IN HL
 000340   0CD 006 003        CALL CHKAD    ;RET CARRY IF S.A..GT.SIZE
 000343   0DA 0F0 000        JC ERROR      ;ERROR, RESTART
 000346   0C3 0AE 000        JMP PUN1      ; GO GET DUMP PARAMETERS
                          ;
                          ;SUBROUTINE ADRIN READS AND SAVES IN THE STACK A HEX CHARACTER STRING
                          ;UNTIL A TERMINATING CHARACTER IS ENCOUNTERED. THE UP TO FOUR MOST
                          ;RECENT CHARACTERS ARE THEN ASSEMBLED AND MOVED INTO REGISTER PAIR HL.
                          ;THE STACK POINTER IS RESTORED.TERMINATING CHARACTERS ARE SPACE,COMMA,
                          ;LF, AND CR. ON RETURN E=1(FOR A CR), E=2(A LF),
                          ; D=0(VALID CHARACTERS IN HL, AND D=1(NO CHARACTERS IN HL).
                          ; ANY OTHER NON-HEX CHARACTER WILL ABORT ROUTINE WITH AN ERROR...
 000349   016 000      ADRIN→ MVI  D,0      ;INITIALIZE CHARACTER COUNT.
 00034B   05A                 MOV E,D
 00034C   062                 MOV H,D
 00034D   06A                 MOV L,D   ;SETH,L =0
 00034E   0CD 0FD 004 AGAIN→ CALL NECHO     ;READ ASCII CHARACTER.
```

8080 MACRO ASSEMBLER, VER 2.0 ERRORS = 0 PAGE 11

```
000351  04F                     MOV C,A      ;PUT ASCII INTO C
000352  0FE 00A                 CPI 0AH     ;SENSE LF
000354  0CA 06B 003             JZ TERM2
000357  0FE 02C                 CPI ','      ;SENSE COMMA
000359  0CA 06D 003             JZ   TERM
00035C  0FE 00D                 CPI 0DH     ;SENSE CR
00035E  0CA 06C 003             JZ TERM1
000361  0FE 020                 CPI 20H     ;SENSE SPACE
000363  0CA 06D 003             JZ  TERM
000366  0C5                     PUSH B       ;SAVE CHARACTER
000367  014                     INR  D       ;INCREMENT COUNT.
000368  0C3 04E 003             JMP  AGAIN   ;READ NEXT CHARACTER.
00036B  01C             TERM2→  INR E     ;+1 TO E
00036C  01C             TERM1→  INR E     ;+1 TO E
00036D  0AF             TERM→   XRA A     ;A=0
00036E  0BA                     CMP D   ;D=NO. OF CHARS. INPUTTED
00036F  0CA 0AB 003             JZ TERM3   ;HL=0,GO SET D=1
000372  0C1                     POP  B       ;RETRIEVE ASCII CHARACTER.
000373  0CD 003 005             CALL HEXNB   ;CONVERT TO HEX.
000376  0DA 0F0 000             JC ERROR
000379  06F                     MOV  L,A     ;SAVE LEAST DIGIT.
00037A  0AF                     XRA A  ; SET A=0
00037B  067                     MOV H,A      ;INITIALIZE H
00037C  015                     DCR D
00037D  0C8                     RZ           ;IF 1 THEN ADRIN RETURN.
00037E  0C1                     POP  B       ;RETRIEVE NEXT CHARACTER.
00037F  0CD 003 005             CALL HEXNB   ;CONVERT TO HEX.
000382  0DA 0F0 000             JC ERROR
000385  0CD 025 005             CALL SHFT4       ;SHIFT LEFT FOUR.
000388  085                     ADD  L       ;ADD LEAST DIGIT.
000389  06F                     MOV  L,A     ;LOAD LOW DIGITS.
00038A  015                     DCR D
00038B  0C8                     RZ           ; IF 2 THEN ADRIN RETURN
00038C  0C1                     POP  B       ;RETRIEVE NEXT CHARACTER.
00038D  0CD 003 005             CALL HEXNB   ;CONVERT TO HEX.
000390  0DA 0F0 000             JC ERROR
000393  067                     MOV  H,A     ;SAVE PENULTIMATE DIGIT.
000394  015                     DCR D
000395  0C8                     RZ           ;IF 3 THEN ADRIN RETURN
000396  0C1                     POP  B
000397  0CD 003 005             CALL HEXNB   ;CONVERT TO HEX.
00039A  0DA 0F0 000             JC ERROR
00039D  0CD 025 005             CALL SHFT4     ;SHIFT LEFT FOUR.
0003A0  084                     ADD  H
0003A1  067                     MOV  H,A     ;LOAD HIGH DIGITS.
0003A2  015                     DCR D
0003A3  0C8                     RZ         ;FOUR RETURN
0003A4  033             AJSP→   INX  SP      ;ADJUST STACK POINTER
0003A5  033                     INX  SP
0003A6  015                     DCR  D       ;DECREMENT COUNT.
0003A7  0C8                     RZ           ;ADRIN RETURN.
```

```
8080 MACRO ASSEMBLER, VER 2.0  ERRORS = 0 PAGE 12

0003A8   0C3 0A4 003          JMP  AJSP
0003AB   014           TERM3→ INR D    ;+1 TO D
0003AC   0C9                  RET

                       ;
                       ;BEGIN NEW LINE AND PRINT CONTENTS OF REGISTER PAIR HL
0003AD   0CD 0E6 004   ADOUT→ CALL CRLF     ;PRINT ADDRESS
0003B0   04C           RPOUT→ MOV  C,H      ;PRINT HIGH DIGITS
0003B1   0CD 041 005          CALL HEXCO
0003B4   04D                  MOV  C,L      ;PRINT LOW DIGITS
0003B5   0CD 041 005          CALL HEXCO
0003B8   0CD 097 004          CALL SPACE    ;PRINT A SPACE
0003BB   0C9                  RET           ;ADOUT/RPOUT RETURN

                       ;
                       ; PUNCH 4 INCHES OF LEADER/TRAILER
0003BC   03E 029       LDRTR→ MVI A,41D     ; 4 INCHES WORTH
0003BE   0E5           LDRT1→ PUSH H      ;SAVE H
0003BF   067                  MOV H,A
0003C0   0AF                  XRA A         ; SET A=0
0003C1   04F                  MOV C,A
0003C2   025           LT1→   DCR H         ;DECREMENT COUNTER
0003C3   0BC                  CMP H         ;SENSE END
0003C4   0CA 0CD 003          JZ LT2     ;RETURN
0003C7   0CD 049 000          CALL  PO      ;PUNCH 1 SPROCKET HOLE
0003CA   0C3 0C2 003          JMP LT1       ; GO ON
0003CD   0E1           LT2→   POP H         ;RESTORE H
0003CE   0C9                  RET

                       ;
                       ;REGISTER DISPLAY HEADING
0003CF   041 020 046 HDING→ DB   'A F  B C  D E  H L  ADD  SP  STACK',0H
0003D2   020 020 042
0003D5   020 043 020
0003D8   020 044 020
0003DB   045 020 020
0003DE   048 020 04C
0003E1   020 020 041
0003E4   044 044 020
0003E7   020 053 050
0003EA   020 020 053
0003ED   054 041 043
0003F0   04B 000
0003F2   020 046 03D FLGWD→ DB  ' F=S Z 0 AC 0 P 1 CY',0H
0003F5   053 020 05A
0003F8   020 030 020
0003FB   041 043 020
0003FE   030 020 050
000401   020 031 020
000404   043 059 000
000407   050 055 04E PMESS→ DB 'PUNCH ON?'
00040A   043 048 020
00040D   04F 04E 03F
000410   00D 00A 000          DB 015Q,012Q,0H
```

```
8080 MACRO ASSEMBLER, VER 2.0  ERRORS = 0 PAGE 13

                            ;       BREAKPOINT ROUTINE
                            ;PRINT CURRENT CONTENT OF ALL REGISTER PAIRS
000413  0F5           HEXBR→ PUSH PSW      ;SAVE FLAGS AND A REG
000414  03A 0DE 083          LDA BCTR      ;GET BREAK CTR
000417  03D                  DCR A         ; -1
000418  0CA 022 004          JZ REGPR      ;OK GO PRINT REGISTERS
00041B  032 0DE 083          STA BCTR      ;SAVE BCTR-1
00041E  0F1                  POP PSW       ;RESTORE A AND FLAGS BEFORE RETURN
00041F  0C3 0D9 083          JMP SAVE1     ;CONTINUE PROGRAM
000422  0F1           REGPR→ POP PSW     ;RESTORE A AND FLAGS
000423  0E5                  PUSH H        ; SAVE ALL REGISTERS
000424  0D5                  PUSH D
000425  0C5                  PUSH B
000426  0F5                  PUSH PSW
000427  0CD 0E6 004 BRPRT→   CALL CRLF    ; DISPLAY ALL REGISTERS
00042A  001 0CF 003          LXI  B,HDING  ;INITIALIZE HEADING PPOINTER
00042D  0CD 0D0 004          CALL PRMSG    ;PRINT CHAR  STRING
000430  001 0F2 003          LXI B,FLGWD   ;PRINT FLAG
000433  0CD 0D0 004          CALL PRMSG    ; WORD DESCRIPTION
000436  0E1                  POP  H        ;RETRIEVE A AND FLAGS
000437  07D                  MOV A,L
000438  032 0D3 083          STA PROM    ;STORE FLAG
00043B  0CD 0AD 003          CALL ADOUT    ;PRINT CRLF,FLAG AND A
00043E  0E1                  POP  H        ;RETRIEVE PAIR B
00043F  0CD 0B0 003          CALL RPOUT
000442  0E1                  POP  H        ;RETRIEVE PAIR D
000443  0CD 0B0 003          CALL RPOUT
000446  0E1                  POP  H        ;RETRIEVE PAIR H
000447  0CD 0B0 003          CALL RPOUT.
00044A  0E1                  POP  H        ;RETRIEVE BREAKPOINT ADDRESS
00044B  02B                  DCX  H        ;RST IS A CALL
00044C  03A 0D2 083          LDA PFLAG     ;TEST FOR TRAP?
00044F  0FE 054              CPI 'T'
000451  0CA 091 004          JZ REGPT      ;YES TAKE TRAP BRANCH
000454  03A 0D9 083          LDA  SAVE1    ;RESTORE BREAKPOINT  1ST BYTE
000457  077                  MOV  M,A
000458  0CD 0B0 003          CALL RPOUT    ; PRINT ADDRESS
00045B  03A 0DD 083          LDA NB        ;GET NO. OF BYTES
00045E  047                  MOV B,A       ;NO. BYTES INTO B
00045F  005                  DCR B         ;-1 FROM B
000460  0CA 071 004          JZ  PRTSP     ;DONE,GO PRINT SP
000463  03A 0DA 083          LDA SAVE2     ;GET 2ND BYTE
000466  023                  INX H         ;+1 TO MEM PTR
000467  077                  MOV M,A       ;RESTORE 2ND BYTE
000468  005                  DCR B         ;-1 FROM B
000469  0CA 071 004          JZ  PRTSP     ;DNE,GO PRINT SP
00046C  023                  INX H         ;INC, MEM PTR
00046D  03A 0DB 083          LDA SAVE3     ;GET 3RD BYTE
000470  077                  MOV M,A       ;RESTORE 3RD BYTE
000471  0AF           PRTSP→ XRA  A        ; SET A=0
000472  06F                  MOV L,A
```

```
8080 MACRO ASSEMBLER, VER 2.0  ERRORS = 0  PAGE 14

0004A3  06 00                   MOV H,A         ;SET H+L=0
0004A4  039                     DAD  SP         ;ADD SP TO H + L
0004A5  0CD 0B0 003             CALL RPOUT      ;PRINT SP
0004A8  0E3                     XTHL            ;STACK CONTENTS TO H+L
0004A9  0CD 0B0 003             CALL RPOUT      ;PRINT STACK
0004AC  0CD 097 004             CALL SPACE      ;PRINT A SPACE
0004AF  0CD 097 004             CALL SPACE      ;PRINT 2ND SPACE
0004B2  03A 0D3 083             LDA PROM        ;GET FLAG
0004B5  0CD 0A7 004             CALL BITP1      ;DO 1ST 4 BITS OF THE FLAG
0004B8  0CD 097 004             CALL SPACE      ;PRINT A SPACE
0004BB  0CD 0A7 004             CALL BITP1      ;DO LAST 4 BITS
0004BE  0C3 0A6 000             JMP START       ;GO START
0004C1  0CD 0B0 003 REGPT→      CALL RPOUT      ;PRINT ADDRESS
0004C4  0C3 071 004             JMP PRTSP       ;PRINT SP, ETC

0004C7  00E 020         SPACE→  MVI C,' '       ;A SPACE
0004C9  0CD 046 000     SPAC1→  CALL CO         ;PRINT A CHAR
0004CC  0C9                     RET
0004CD  00E 030         PRTZR→  MVI C,'0'       ;PRINT ZERO
0004CF  0C3 0C9 004             JMP SPAC1
0004D2  00E 031         PRT1→   MVI C,'1'       ;PRINT ONE
0004D4  0C3 0C9 004             JMP SPAC1
0004D7  006 004         BITP1→  MVI B,4         ;4 CTR
0004D9  017             BITP2→  RAL             ;ROT LEFT TO CARRY
0004DA  0D4 0CD 004             CNC PRTZR       ;0=NO CARRY
0004DD  0DC 0D2 004             CC PRT1         ;1=CARRY
0004E0  0CD 097 004             CALL SPACE      ;PRINT A SPACE
0004E3  005                     DCR B
0004E4  0C2 0D9 004             JNZ BITP2       ;NOT 4 TIMES
0004E7  0C9                     RET             ;DONE

                        ;
                        ;       TRAP ROUTINE
0004E8  0E5             TRAP→   PUSH H      ;SAVE ALL REGISTER PAIRS
0004E9  0D5                     PUSH D
0004EA  0C5                     PUSH B
0004EB  0F5                     PUSH PSW
0004EC  03E 054                 MVI A,'T'       ;SET A=T
0004EE  032 0D2 083             STA PFLAG       ;STORE IN PFLAG
0004F1  0CD 0E6 004             CALL CRLF       ;PRINT CR + LF
0004F4  001 0FD 004             LXI B,TMESS     ;INITIALIZE TRAP MESSAGE PTR
0004F7  0CD 0D0 004             CALL PRMSG      ;PRINT TRAP MESSAGE
0004FA  0C3 0B6 000             JMP STRD        ;GO READ TTY INPUT CONTROL
0004FD  054 03D 000 TMESS→      DB 'T=',0H

                        ;
                        ;PRINT A MESSAGE TERMINATED BY A '0'
                        ;PRINTS AN ASCII CHARACTER STRING POINTED BY BC
                        ;AND TERMINATED BY A 0 OR NULL CHARACTER TO CONSOLE
0000D0  0F5             PRMSG→  PUSH    PSW
0000D1  0C5                     PUSH    B
0000D2  0E5                     PUSH    H
0000D3  060                     MOV     H,B     ;HC=POINTER TO CHARACTER STRING
```

```
8080 MACRO ASSEMBLER, VER 2.0  ERRORS = 0 PAGE 15

0004D4   069                    MOV     L,C
0004D5   07E            PRMS1→  MOV     A,M     ;JUMP IF CHAR=0
0004D6   0B7                    ORA     A
0004D7   0CA 0E2 004            JZ      PRMS2
0004DA   04F                    MOV     C,A     ;PRINT CHAR
0004DB   0CD 046 000            CALL    CO
0004DE   023                    INX     H       ;ADVANCE POINTER
0004DF   0C3 0D5 004            JMP     PRMS1
0004E2   0E1            PRMS2→  POP     H
0004E3   0C1                    POP     B
0004E4   0F1                    POP     PSW
0004E5   0C9                    RET
                        ;  PRINT A CRLF TO THE CONSOLE
0004E6   0C5            CRLF→   PUSH    B
0004E7   00E 08D                MVI     C,8DH   ;PRINT CR
0004E9   0CD 046 000            CALL    CO
0004EC   00E 08A                MVI     C,8AH   ;PRINT LF
0004EE   0CD 046 000            CALL    CO
0004F1   0C1                    POP     B
0004F2   0C9                    RET
                        ; GET A CHARACTER FROM THE CONSOLE AND ECHOS IT
0004F3   0C5            CECHO→  PUSH    B
0004F4   0CD 040 000            CALL    CI      ;A=CONSOLE CHAR
0004F7   04F                    MOV     C,A     ;PRINT A
0004F8   0CD 046 000            CALL    CO
0004FB   0C1                    POP     B
0004FC   0C9                    RET
0004FD   0CD 0F3 004    NECHO→  CALL CECHO
000500   0E6 07F                ANI 07FH        ;MASK 8TH BIT
000502   0C9                    RET
                        ;HEX TO NIBBLE ROUTINE
                        ;CONVERTS HEX CHAR IN C TO NIBBLE IN A
                        ;CARRY SET FOR NONE HEX CHAR
000503   079            HEXNB→  MOV     A,C     ;A=CHAR-'0'
000504   0D6 030                 SUI     '0'
000506   0D8                     RC              ;RETURN IF CHAR<'0'
000507   0C6 0E9                 ADI     '0'-'G' ;A=A+'0'-'G'
000509   0D8                     RC              ;RETURN IF CHAR>='F'
00050A   0C6 006                ADI     6       ;A=A+6
00050C   0F2 012 005            JP      HEXNO   ;JUMP IF A>=0
00050F   0C6 007                ADI     7       ;A=A+7
000511   0D8                     RC              ;RETURN IF CHAR>#9# OR CHAR<#A#
000512   0C6 00A        HEXNO→   ADI     10      ;RETURN A+10
000514   0B7                     ORA     A
000515   0C9                     RET
                        ;BYTE TO 2 HEX CHARACTERS
                        ;CONVERTS 8 BIT (C) INTO 2 HEX CHAR IN BA
000516   0C5            BTHEX→   PUSH    B       ;SAVE LOW NIBBLE
000517   079                     MOV     A,C     ;A=ROR(C,4)
000518   0CD 025 005             CALL SHFT4      ;SHIFT LEFT 4
00051B   04F                    MOV     C,A     ;A=HEXNB(ROR(C,4))
```

```
8.80 MACRO ASSEMBLER, VER 2.0  ERRORS = 0 PAGE 16

000518  0CD 036 005          CALL    NBHEX
00051B  0C1                  POP     B       ;B=HEX CHAR OF HIGH NIBBLE
00051C  047                  MOV     B,A     ;A=HEXNB(C)
00051D  0CD 036 005          CALL    NBHEX
000520  0C9                  RET
000521  007          SHFT4→  RLC     ;SHIFT LEFT 4
000522  007                  RLC
000523  007                  RLC
000524  007                  RLC
000525  0C9                  RET
                     ;PUNCH  2 HEX CHARS  FROM C REG
000526  0CD 016 005 PTHEX→    CALL BTHEX    ;CONVERT C TO MS HEX IN B
000529  048                  MOV C,B       ;  AND LS HEX IN A
00052A  0CD 049 000          CALL PO       ; PUNCH MS HEX
00052D  04F                  MOV C,A
00052E  0CD 049 000          CALL PO       ;PUNCH LS HEX
000531  0C9                  RET
                     ;NIBBLE TO HEX ROUTINE
                     ;CONVERTS NIBBLE IN C TO HEX IN A
000532  079          NBHEX→  MOV     A,C     ;MASK OFF NIBBLE
000533  0E6 00F              ANI     0FH
000535  0C6 030              ADI     '0'     ;A=A+'0'
000537  0FE 03A              CPI     '9'+1   ;RETURN IF A[=9
000539  0F8                  RM
00053A  0C6 007              ADI     'A'-'0'-10 ;A=A+'A'-'0'+10
00053C  0C9                  RET
                     ;BYTE TO 2HEX CHAR ON CONSOLE
                     ;C CONTAINS BYTE TO BE OUTPUTTED TO CONSOLE AS 2 HEX
                     ;CHARACTERS
00053D  0C5          HEXCO→  PUSH    B
00053E  0F5                  PUSH    PSW     ;BA=BTHEX(BC)
00053F  0CD 016 005          CALL    BTHEX
000542  048                  MOV     C,B     ;PRINT B
000543  0CD 046 000          CALL    CO
000546  04F                  MOV     C,A     ;PRINT A
000547  0CD 046 000          CALL    CO
00054A  0F1                  POP     PSW
00054B  0C1                  POP     B
00054C  0C9                  RET
                     ;THIS ROUTINE OUTPUT A SPACE THE 2 HEX CHARACTERS OF THE
                     ;BYTE CONTAINED IN C TO THE CONSOLE
00054D  0C5          HX1CO→  PUSH    B
00054E  00E 020              MVI     C,' '   ;PRINT SPACE
000550  0CD 046 000          CALL    CO
000553  0C1                  POP     B
000554  0C3 041 005          JMP     HEXCO   ;PRINT 2 HEX CHAR TO CONSOLE
                     ;THIS ROUTINE OUTPUTS A SPACE, THEN 4 HEX CHARACTERS
                     ;CONTAINED IN BC TO THE CONSOLE
000557  0C5          HX2CO→  PUSH    B
000558  00E 020              MVI     C,' '   ;PRINT SPACE
00055A  0CD 046 000          CALL    CO
```

```
8080 MACRO ASSEMBLER, VER 2.0  ERRORS = 0 PAGE 17

000561  048                     MOV     C,B     ;PRINT 2 HEX CHAR CONTAINED IN B
000562  0CD 041 005             CALL    HEXCO
000565  0C1                     POP     B
000566  0C3 041 005             JMP     HEXCO   ;PRINT 2 HEX CHARACTERS CONTAINED IN C
                        ;THIS ROUTINE SAVES (C) AS 2 HEX CHARACTERS , PUNCHS THEM,
                        ;AND ADDS THEM TO THE CHECKSUM IN D
000569  0C5             SMXBT-  PUSH    B
00056A  0F5                     PUSH    PSW
00056B  079                     MOV     A,C     ;CHECKSUM=CHECKSUM+C
00056C  082                     ADD     D
00056D  057                     MOV     D,A
00056E  0CD 016 005             CALL    BTHEX   ;CONVERT C INTO 2 HEX CHAR IN BA
000571  048                     MOV     C,B     ;SAVBT B
000572  0CD 049 000             CALL    PO
000575  04F                     MOV     C,A
000576  0CD 049 000             CALL    PO
000579  0F1                     POP     PSW
00057A  0C1                     POP     B
00057B  0C9                     RET
                        ; PUNCH HEX RECORDS
                        ;THIS ROUTINE CONVERTS THE CONTENTS OF MEMORY FROM
                        ;BC TO DE INTO HEXADECIMAL RECORDS OF UP TO 16 BYTES IN LENGHT
                        ;AND USES THE ROUTINE STORED AT SAVBT TO OUTPUT THE CHARACTERS
00057C  0CD 0E6 001     WRITE-  CALL GETA1      ;DE=END ADD ,HL=START ADD
00057F  07B                     MOV     A,E     ;DE=ENDING ADDRESS-STARTING ADDRESS
000580  095                     SUB     L
000581  05F                     MOV     E,A
000582  07A                     MOV     A,D
000583  09C                     SBB     H
000584  057                     MOV     D,A
000585  0D8                     RC              ;RETURN IF STARTING ADDRESS)ENDING ADDRESS
000586  013                     INX     D       ;COUNT=DE=DE+1
000587  03E 005         WTHE0-  MVI A,5
000589  0CD 0BE 003             CALL [DRTI      ;PUNCH 5 SPACES
00058C  00E 03A                 MVI     C,3AH   ;SENDOUT COLON,BEGINING OF RECORD
00058E  0CD 049 000             CALL    PO
000591  0D5                     PUSH    D       ;SAVE COUNT
000592  001 010 000             LXI     B,16    ;DE=COUNT-16
000595  07B                     MOV     A,E
000596  091                     SUB     C
000597  05F                     MOV     E,A
000598  07A                     MOV     A,D
000599  098                     SBB     B
00059A  057                     MOV     D,A
00059B  0D2 0A5 005             JNC     WTHE1   ;JUMP IF COUNT1=16
00059E  0C1                     POP     B       ;BC=COUNT
00059F  011 000 000             LXI     D,0     ;COUNT=0
0005A2  0C3 0A7 005             JMP     WTHE2
0005A5  033             WTHE1-  INX     SP      ;POP COUNT
0005A6  033                     INX     SP
0005A7  0D5             WTHE2-  PUSH    D       ;SAVE COUNT
```

```
8080 MACRO ASSEMBLER, VER 2.0  ERRORS = 0 PAGE 18

0005A8   0CD 0C3 005          CALL    WTHE5     ;CHECKSUM=0, SEND BC, HL
0005AB   00E 000              MVI     C,0       ;SEND OUT 0 (RECORD TYPE)
0005AD   0CD 069 005          CALL    SHXBT
0005B0   04E            WTHE3→ MOV    C,M       ;SEND OUT CONTENTS OF MEMORY
0005B1   0CD 069 005          CALL    SHXBT
0005B4   023                  INX     H         ;ADVANCE POINTERS
0005B5   005                  DCR     B         ;JUMP IF NOT DONE
0005B6   0C2 0B0 005          JNZ     WTHE3
0005B9   0CD 0D2 005          CALL    WTHE6     ;SEND CHECKSUM,CR,LF
0005BC   0D1                  POP     D         ;COUNT
0005BD   07B                  MOV     A,E       ;JUMP IF COUNT<0
0005BE   0B2                  ORA     D
0005BF   0C2 087 005          JNZ     WTHE0
0005C2   0C9                  RET
0005C3   016 000        WTHE5→ MVI    D,0       ;CHECKSUM=0
0005C5   041                  MOV     B,C       ;SAVE LENGHT
0005C6   0CD 069 005          CALL    SHXBT     ;SEND LENGHT
0005C9   04C                  MOV     C,H       ;SEND ADDRESS (HL)
0005CA   0CD 069 005          CALL    SHXBT
0005CD   04D                  MOV     C,L
0005CE   0CD 069 005          CALL    SHXBT
0005D1   0C9                  RET
0005D2   0AF            WTHE6→ XRA    A         ;SEND -CHECKSUM
0005D3   092                  SUB     D
0005D4   04F                  MOV     C,A
0005D5   0CD 069 005          CALL    SHXBT
0005D8   00E 08D        PCRLF→ MVI    C,8DH     ;PUNCH CR
0005DA   0CD 049 000          CALL    PO
0005DD   00E 08A              MVI     C,8AH     ;PUNCH LF
0005DF   0CD 049 000          CALL    PO
0005E2   0C9                  RET
                        ;THIS ROUTINE OUTPUTS A HEXADECIMAL EOF RECORD.
0005E3   02E 000        WTEOF→ MVI L,0       ;EOF HAS ADDR.=0000
0005E5   026 000              MVI H,0
0005E7   03E 005              MVI A,5
0005E9   0CD 0BE 003          CALL LDRT1   ;PUNCH 5 SPACES
0005EC   00E 03A              MVI     C,3AH     ;SEND COLON, BEGINING OF RECORD
0005EE   0CD 049 000          CALL    PO
0005F1   00E 000              MVI     C,0       ;SEND 0 LENGTH
0005F3   0CD 0C3 005          CALL    WTHE5     ;SEND LENGTH, AND ADDR.=0
0005F6   00E 001              MVI     C,1       ;RECORD TYPE=1
0005F8   0CD 069 005          CALL    SHXBT     ;SEND RECORD TYPE
0005FB   0C9                  RET
                        ;THIS ROUTINE READS A HEXIDECIMAL TAPE AND STORES IT IN MEMORY.
                        ;HEX PAPER TAPE FORMAT
                        ;  X(COLON OR →) =START OF RECORD
                        ; XX 1ST PAIR OF ASCII CHARS. = RECORD LENGTH
                        ; XX 2ND PAIR OF ASCII CHARS. = HIGH ADDRESS
                        ; XX 3RD PAIR OF ASCII CHARS. = LOW PART OF ADDRESS
                        ; XX 4TH PAIR OF ASCII CHARS. = RECORD TYPE
                        ; XX  FOLLOWING PAIRS OF ASCII CHARS. = DATA
```

```
8080 MACRO ASSEMBLER, VER 2.0  ERRORS = 0 PAGE 19

                        ;XX LAST PAIR OF ASCII CHARS = CHECKSUM
0005FC  0AF             HEXTO→  XRA A
0005FD  032 0DD 083             STA NB      ;SET ERROR CTR=0
000600  0CD 043 000     HEXT1→  CALL    RI      ;READ INPUT CHAR
000603  0E6 07F                 ANI     177Q    ;MASK OFF PARITY
000605  0FE 03A                 CPI     3AH     ;IF COLON
000607  0CA 00F 006             JZ      HEXTS   ;GO LOAD
00060A  0FE 05F                 CPI     05FH    ;IF →
00060C  0C2 000 006             JNZ     HEXT1   ;IF NOT COLON OR → ;DO NOT START
00060F  016 000         HEXTS→  MVI     D,0     ;ZERO CHECKSUM
000611  0CD 057 006             CALL    HEXRD   ;READ LENGTH
000614  047                     MOV     B,A     ;STORE LENGTH IN B
000615  0CD 057 006             CALL    HEXRD   ;READ HIGH ADDRESS
000618  067                     MOV     H,A     ;H=HIGH ADDRESS
000619  0CD 057 006             CALL    HEXRD   ;READ LOW ADDRESS
00061C  06F                     MOV     L,A     ;L=LOW ADDRESS
00061D  04A                     MOV     C,D     ;SAVE CHECKSUM
00061E  0EB                     XCHG            ;PUT ADD IN D,E
00061F  02A 0D7 083             LHLD    MOVAD   ;GET OFFSET ADDRESS
000622  019                     DAD     D       ;ADD  IN H,L:ADD  +OFFSET
000623  058                     MOV     E,B     ;LENGTH IN E
000624  051                     MOV     D,C     ;RESTORE CHECKSUM
000625  0CD 057 006             CALL    HEXRD   ;READ TYPE, IGNORE IT
000628  0CD 08F 006             CALL    HEOF
00062B  0CD 057 006     HEXT2→  CALL    HEXRD   ;READ DATA
00062E  077                     MOV     M,A     ;STORE DATA IN MEMORY
00062F  023                     INX     H       ;ADVANCE POINTER
000630  01D                     DCR     E       ;JUMP IF NOT DONE
000631  0C2 02B 006             JNZ     HEXT2
000634  0CD 057 006             CALL    HEXRD   ;READ CHECKSUM
000637  07A                     MOV     A,D     ;JUMP IF CHECKSUM=0
000638  0B7                     ORA     A
000639  0CA 000 006             JZ      HEXT1
00063C  001 048 006             LXI B,CKSUM     ;PRINT CHECKSUM ERROR
00063F  0CD 0AB 006             CALL HXPRM      ;HEX PRINT MESSAGE
000642  0CD 0E6 004             CALL CRLF
000645  0C3 000 006             JMP HEXT1       ;CONTINUE TO READ NEXT RECORD
000648  043 048 045 CKSUM→       DB 'CHECKSUM ERROR',0H
00064B  043 048 053
00064E  055 04D 020
000651  045 052 052
000654  04F 052 000
                        ;THIS ROUTINE READS A HEX BYTE AND CONVERTS IT TO A BINARY BYTE
000657  0C5             HEXRD→  PUSH    B
000658  0CD 043 000             CALL    RI      ;FETCH INPUT CHARACTER
00065B  0E6 07F                 ANI     7FH     ;MASK PARITY BIT
00065D  0DA 0A2 006             JC      HEXER   ;JUMP IF NOT HEX
000660  04F                     MOV     C,A     ;CONVERT IF TO A NIBBLE
000661  0CD 003 005             CALL    HEXNB
000664  0DA 0A2 006             JC      HEXER   ;JUMP IF NON HEX CHARACTER ENCOUNTERED
000667  0CD 025 005             CALL SHFT4      ;NIBBLE=NIBBLE*16.
```

```
8080 MACRO ASSEMBLER, VER 2.0  ERRORS = 0 PAGE 20

000066A   047                     MOV     B,A     ;SAVE IN B
000066B   0CD 043 000             CALL    RI      ;FETCH SECOND HEX CHARACTER
000066E   0E6 07F                 ANI     7FH     ;MASK PARITY BIT
0006*0    0DA 0A2 006             JC      HEXER   ;JUMP IF NOT HEX
000673    04F                     MOV     C,A     ;CONVERT IT INTO A NIBBLE
000674    0CD 003 005             CALL    HEXNB
000677    0D2 09B 006             JNC     HEXR1   ;JUMP IF HEX CHAR ENCOUNTERED
00067A    001 081 006 HEXR0→      LXI B,BADHX     ;PRINT ERROR MESSAGE
00067D    0CD 0AB 006             CALL HXPRM  ;HEX PRINT MESSAGE
000680    0C9                     RET
000681    045 04F 046 BADHX→      DB 'EOF OR BADHEX',0H
000684    020 04F 052
000687    020 042 041
00068A    044 048 045
00068D    058 000
00068F    0FE 001       HEOF→     CPI 1H  ;EOF TYPE=1
000691    0C0                     RNZ     ;NOT AN EOF RETURN
000692    07B                     MOV A,E  ;GET LENGTH
000693    0B7                     ORA A
000694    0C0                     RNZ     ;NOT EOF , RETURN
000695    0CD 07A 006             CALL HEXR0      ;AN EOF HAS BEEN FOUND
000698    0C3 0A6 000             JMP START   ;ALL DONE
00069B    0B0           HEXR1→    ORA     B       ;COMBINE NIBBLES
00069C    04F                     MOV     C,A
00069D    082                     ADD     D       ;ADD TO CHECKSUM
00069E    057                     MOV     D,A
00069F    079                     MOV     A,C     ;RETURN NIBBLES
0006A0    0C1                     POP     B
0006A1    0C9                     RET
0006A2    0CD 07A 006 HEXER→      CALL HEXR0  ;PRINT MESSAGE
0006A5    0CD 0E6 004             CALL CRLF
0006A8    0C3 000 006             JMP HEXTI   ;GO FOR NEXT RECORD
0006AB    0CD 0D0 004 HXPRM→      CALL PRMSG
0006AE    03A 0DD 083             LDA NB
0006B1    03C                     INR A
0006B2    032 0DD 083             STA NB   ;ERROR COUNTER +1
0006B5    0C9                     RET
                      ;THIS ROUTINE READS THE INPUT CHARACTERS FROM A
                      ;BINARY TAPE AND STORES THEM IN MEMORY
0006B6    0CD 043 000 BINTP→      CALL    RI      ;FETCH CHARACTER FROM INPUT ROUTINE
0006B9    0FE 080                 CPI     200Q    ;TEST FOR LEADER
0006BB    0CA 0C9 006             JZ      BINTO   ; YES ON LEADER
0006BE    047                     MOV     B,A     ;SAVE IT IN B
0006BF    0E6 0C0                 ANI     300Q    ;IS IT A FIELD?
0006C1    0FE 0C0                 CPI     300Q    ;TEST
0006C3    0CA 0D9 006             JZ      BINT2   ;YES A NEW FIELD
0006C6    0C3 0B6 006             JMP     BINTP   ; NOT ON LEADER YET
0006C9    0CD 043 000 BINTO→      CALL    RI      ;FETCH CHARACTER FROM INPUT ROUTINE
0006CC    0FE 080                 CPI     200Q    ;JUMP IF STILL ON LEADER
0006CE    0CA 0C9 006             JZ      BINTO
0006D1    047                     MOV     B,A     ;SAVE IN B
```

```
8080 MACRO ASSEMBLER, VER 2.0  ERRORS = 0 PAGE 21

0006D2  0E6 0C0          ANI     300Q    ;SHOULD BE FIELD DEFINITION
0006D4  0FE 0C0          CPI     300Q    ;JUMP IF IT ISN≠T
0006D6  0C2 0F0 000      JNZ     ERROR
0006D9  078      BINT2-  MOV     A,B     ;FIELD*16 AND F0H
0006DA  0CD 025 005      CALL    SHFT4   ;SHIFT LEFT 4
0006DD  0E6 0F0          ANI     360Q
0006DF  067              MOV     H,A     ;SAVE IN H
0006E0  0CD 043 000      CALL    RI      ;FETCH CHARACTER FROM INPUT ROUTINE
0006E3  047              MOV     B,A     ;SAVE IN B
0006E4  0E6 040          ANI     100Q    ;SHOULD BE AN ADDRESS DEFINITION
0006E6  0B7              ORA     A       ;JUMP IF IT ISN≠T
0006E7  0CA 0F0 000      JZ      ERROR
0006EA  078              MOV     A,B     ;ADDRESS*4
0006EB  00F              RRC
0006EC  00F              RRC
0006ED  047              MOV     B,A     ;SAVE IN B
0006EE  0E6 00F          ANI     17Q     ;H=H OR ADDRESS*4 AND 0FH
0006F0  0B4              ORA     H
0006F1  067              MOV     H,A
0006F2  078              MOV     A,B     ;L=ADDRESS*4 AND 0C0H
0006F3  0E6 0C0          ANI     300Q
0006F5  06F              MOV     L,A
0006F6  0CD 043 000      CALL    RI      ;FETCH CHARACTER FROM INPUT ROUTINE
0006F9  0E6 03F          ANI     77Q     ;L=L OR CHARACTER AND 3FH
0006FB  0B5              ORA     L
0006FC  06F              MOV     L,A
0006FD  0EB              XCHG            ;PUT ADDRESS IN D,E
0006FE  02A 0D7 083      LHLD    MOVAD     ;GET OFFSET ADDRESS
000701  019              DAD     D       ;ADD  IN H,L=ADD.+OFFSET
000702  0CD 043 000 BINT1- CALL   RI      ;FETCH CHARACTER FROM INPUT ROUTINE
000705  0FE 080          CPI     200Q    ;JUMP IF LEADER ENCOUNTERED
000707  0CA 0C9 006      JZ      BINT0   ;B=CHAR*4 AND C0H
00070A  047              MOV     B,A     ;SAVE IT IN B
00070B  0E6 0C0          ANI     300Q    ;TEST FOR NEW FIELD
00070D  0FE 0C0          CPI     300Q    ;TEST
00070F  0CA 0D9 006      JZ      BINT2     ;YES, A NEW FIELD
000712  078              MOV     A,B     ;RESTORE A
000713  00F              RRC
000714  00F              RRC
000715  0E6 0C0          ANI     300Q
000717  047              MOV     B,A
000718  0CD 043 000      CALL    RI      ;FETCH CHARACTER FROM INPUT ROUTINE
00071B  0E6 03F          ANI     77Q     ;A=CHAR AND 3FH OR B
00071D  0B0              ORA     B
00071E  077              MOV     M,A     ;SAVE A IN MEMORY
00071F  023              INX     H       ;ADVANCE POINTER
000720  0C3 002 007      JMP     BINT1
                ;THIS ROUTINE READS A CHARACTER INTO A FROM THE TTY
000723  0DB 003  TTYIN-  IN      FLAG    ;INPUT FLAGS
000725  0E6 002          ANI     TTYDA   ;JUMP IF TTY DOESN≠T HAVE A CHARACTER
000727  0CA 023 007      JZ      TTYIN
```

```
8080 MACRO ASSEMBLER, VER 2.0  ERRORS = 0 PAGE 22


  000729    0DB 002             IN      TTY     ;INPUT CHARACTER
  000729    0C9                 RET
                        ;THIS ROUTINE PRINTS THE (C) TO THE TTY
                        ; NO REGISTERS MODIFIED
  00072D    0F5         TTY0T-> PUSH    PSW
  00072E    0DB 003     TTY01-> IN      FLAG    ;INPUT FLAGS
  000730    0E6 001             ANI     TTYTR   ;JUMP IF TTY NOT READY
  000732    0CA 02E 007         JZ      TTY01
  000735    079                 MOV     A,C     ;SEND CHARACTER TO TTY
  000736    0F6 080             ORI     200Q    ;COMPLETE ASCII
  000738    0D3 002             OUT     TTY
  00073A    0C5                 PUSH    B               ;SAVE CHAR
  00073B    0FE 0BD             CPI     BDH             ;TEST FOR CR
  00073D    0C2 04A 007         JNZ     TTY03           ;NO-GO ON
  000740    03E 033             MVI     A,33H           ;DELAY FOR LONG CR
  000742    00C         TTY02-> INR     C
  000743    0C2 042 007         JNZ     TTY02
  000746    03D                 DCR     A
  000747    0C2 042 007         JNZ     TTY02
  00074A    0C1         TTY03-> POP     B               ;RESTORE CHAR
  00074B    0F1                 POP     PSW
  00074C    0C9                 RET
                                END
NO PROGRAM ERRORS
```

8080 MACRO ASSEMBLER, VER 2.0 ERRORS = 0 PAGE 23

SYMBOL TABLE

* C1

A	000007	ADD2	000213	ADOUT	0003AD	ADRIN	000349
AGAIN	00034E	AJSP	0003A4	B	000000	BADHX	000681
BCTR	0083DE	BINT0	0006C9	BINT1	000702	BINT2	0006D9
BINTP	0006B6	BITP1	0004A7	BITP2	0004A9	BKP1	000192
BKP2	00019C	BKPNT	000154	BRPRT	000427	BTHEX	000516
BYTE	0001B6	C	000001	CECHO	0004F3	CHKAD	000306
CHKM1	000108	CHKMV	0002C2	CI	000040	CKMEM	0000FB
CKSUM	000648	CMD	000025	CO	000046	CRLF	0004E6
D	000002	DOUT	000247	DOUT2	000248	DUMP	0001EE
E	000003	ENDDP	00026E	ENDMV	0002BE	ENDOP	00030F
ERROR	0000F0	EXMNE	0001A6	EXTRA	0083E8	FLAG	000003
FLGWD	0003F2	GETA1	0001E6	GETAD	0001D9	GO	0001AF
H	000004	HDING	0003CF	HEOF	00068F	HEXBR	000413
HEXCO	000541	HEXER	0006A2	HEXNO	000512	HEXNB	000503
HEXRC	00067A	HEXR1	000698	HEXRD	000657	HEXTO	0005FC
HEXT1	000600	HEXT2	00062B	HEXT5	00060F	HWRIT	000142
HX1CO	000551	HX2CO	000558	HXPRM	0006AB	INIT	00008E
L	000005	LDRT1	0003BE	LDRTR	0003BC	LOAD	0001B3
LOAD1	0001C5	LT1	0003C2	LT2	0003CD	M	000006
MODE	0000CE	MOVAD	0083D7	MOVE	000279	MOVE1	00028C
MOVE2	00028F	MOVE3	000292	MOVE4	00029C	MOVE5	0002B1
MVCMP	00030C	MVEP1	0002F8	MVEP2	0002F4	MVEPR	0002F5
MVERR	0002E6	NB	0083DD	NBHEX	000536	NECHO	0004FD
NEWLN	000244	PADDR	0083D5	PCRLF	0005D8	PELTR	000314
PFLAG	0083D2	PMESS	000407	PO	000049	PRMS1	0004D5
PRMS2	0004E2	PRMSG	0004D0	PROM	0083D3	PRT1	0004A2
PRTSP	000471	PRTZR	00049D	PSTRT	000320	PSW	000006
PTHEX	00052A	PUN1	0000AE	PUN3	00023C	PUN4	000200
PUNCH	00032F	RAMJP	0083CF	READE	000123	REGPR	000422
REGPT	000491	RES0	000070	RES1	000076	RES2	00007C
RES3	000082	RES4	000088	RI	000043	RPOUT	0003B0
S000	000000	SAVE1	0083D9	SAVE2	0083DA	SAVE3	0083DB
SAVE4	0083DC	SHFT4	000525	SHXBT	000569	SP	000006
SPAC1	000499	SPAC'	000497	STACK	0083D0	START	0000A6
STRD	000086	TEMP	0083E2	TEMP2	0083E4	TEMP3	0083E6
TERM	00036D	TFR'.1	00036C	TERM2	00036B	TERM3	0003AB
TMESS	0004CD	TRAP	0004B8	TTY	000002	TTYDA	000002
TTYIN	000723	TTYO1	00072E	TTYO2	000742	TTYO3	00074A
TTYOT	00072D	TTYTR	000001	WRITE	00057C	WTEOF	0005E3
WTHE0	000587	WTHE1	0005A5	WTHE2	0005A7	WTHE3	0005B0
WTHE5	0005C3	WTHE6	0005D2				

APPENDIX B: HEX(INTEL) FORMAT

The HEX(INTEL) format object file is a way of representing a BINARY object file in ASCII. The ASCII character set is defined by the "American National Standard Institute, Code for Information Interchange, x3.4-1968"

For example the hexadecimal 8-bit Byte, 3F, is represented in ASCII by an 8-bit Byte containing the ASCII code for 3 of 33 and a second 8-bit Byte containing the ASCII code for F of 46. Thus, the representation of an 8-bit Byte requires twice as many bytes as the Hex representation.

The HEX(INTEL) format is described below according to the fields that constitute a record.

RECORD MARK FIELD: Frame Ø

The ASCII code for a colon (:) is used to signal the start of a record. To allow LBLHEX to load assembled files from the LBL computer center, we have also enabled the ASCII code for (→) to mark the start of a record. The assembly listing, Appendix A, page 19, shows how this done and how the user may change this to any other unique character.

RECORD LENGTH FIELD: Frames 1 and 2

The number of data bytes in the record is represented by two ASCII hexadecimal digits in this field. The maximum number of data bytes in a record is 255 (FF in hexadecimal).

LOAD ADDRESS FIELD: Frames 3 to 6

The four ASCII Hexadecimal digits in Frames 3 to 6 give the address at which the data is loaded. The most significant digit is in Frame 3, with the least significant digit in Frame 6. The first data byte is stored in the location indicated by the load address and successive data bytes are stored in sucessive memory locations.

RECORD TYPE FILED: Frames 7 and 8

The two ASCII hexadecimal digits in this field specify the record type. The most significant digit is in Frame 7. All data type records are type ØØ and end-of-file records are type Ø1. As of the present time, other possible values for this field have not been specied. It does not take too much imagination to foresee a label field of type Ø3 used to identify a data set containing a name (label). One may build up a tape or disk directory based on such a structure.

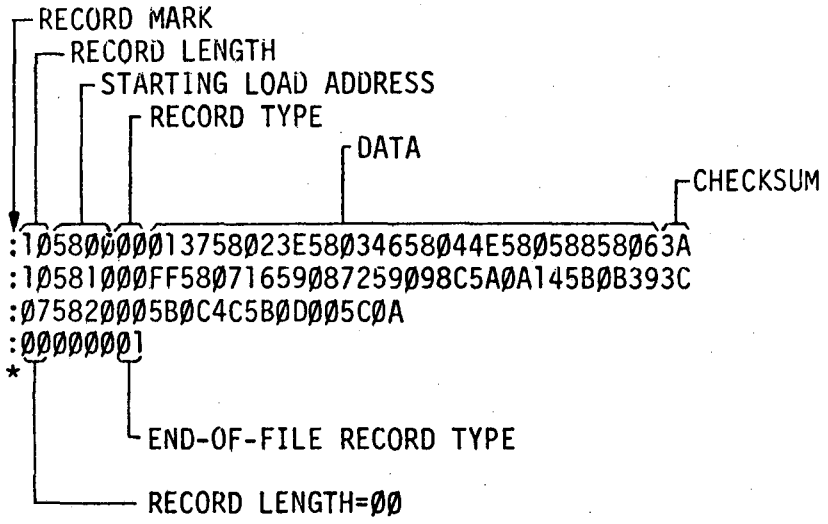
DATA FIELD: Frames 9 to 9+2*(RECORD LENGTH) - 1

A data byte is represented by two frames containing the Hex ASCII characters, with the most significant character first.

CHECKSUM FIELD: Last two frames

The checksum field contains the ASCII hexadecimal representation of the twos complement of the 8-bit sum of the Hex ASCII digits in each frame of the record excluding the record mark. Therefore, the sum of all the Hex ASCII characters in a record, from the record length field to and including the checksum field, is zero.

SAMPLE HEX(INTEL) FORMAT:



AN END-OF-FILE RECORD:

An end-of-file record is of zero length. The address may be the starting address of the program. LBLHEX loads the address 0000H in an end-of-file record. The record type for an end-of file record is 01. There are no data bytes and no checksums.

APPENDIX C: BINARY (LLL) FORMAT

The BINARY (LLL) format is a paper tape format for storage of 8-bit data. This format is a compact means (short tape lengths) of transferring and storing 8-bit Bytes of data. Unfortunately, there are two main disadvantages, there is no checksum and this BINARY format is not the same as other BINARY formats[5].

Each bit of an 8-bit data Byte is represented by the presence or absence of a hole in a paper tape channel. The BINARY (LLL) format is described below:

PAPER TAPE CHANNEL NUMBERS:

A paper tape frame consists of 8-channel positions. Channels 1, 2, and 3 are to the right of the sprocket hole while channels 4,5,6,7, and 8 are to the left of the sprocket hole.

LEADER: Any number of frames

An 8th channel punch signifies leader tape prior to the start of a record.

PAGE AND LOCAL ADDRESS: Frames 1, 2, and 3

The BINARY (LLL) format treats memory addresses (16-bit numbers) by considering, the most significant 8-bits as a page address and the least significant 8-bits as a local address.

A 7th and 8th channel punch signifies the start of a record. Frame 1 contains, a 7th and 8th channel punch and the four-most-significant-bits of the page address in channels 4,3,2, and 1.

Frame 2 contains, a 7th channel punch, the four-least-significant-bits of the page address (in channels 6,5,4, and 3), and the two-most-significant-bits of the local address (in channels 2 and 1.

Frame 3 contains, the next six-bits of the local address in channels 6 though 1.

DATA: Frames 4 to 2*N+4

Each 8-bit data byte occupies two frames. Up to 256 bytes may be included in one record. Therefore N may range from 1 to 256.

Frame 4 contains, the two-most significant-bit in channels 2 and 1. Frame 5 contains, the remaining six-bits in Frames 6 to 1.

The data byte contained in Frames 4 and 5 is loaded at the address contained Frames 1,2, and 3. Successive data bytes are loaded in successive addresses.

TRAILER: Any number of frames

An 8th channel punch signifies trailer tape at the end of a record.

TECHNICAL INFORMATION DEPARTMENT
LAWRENCE BERKELEY LABORATORY
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA 94720