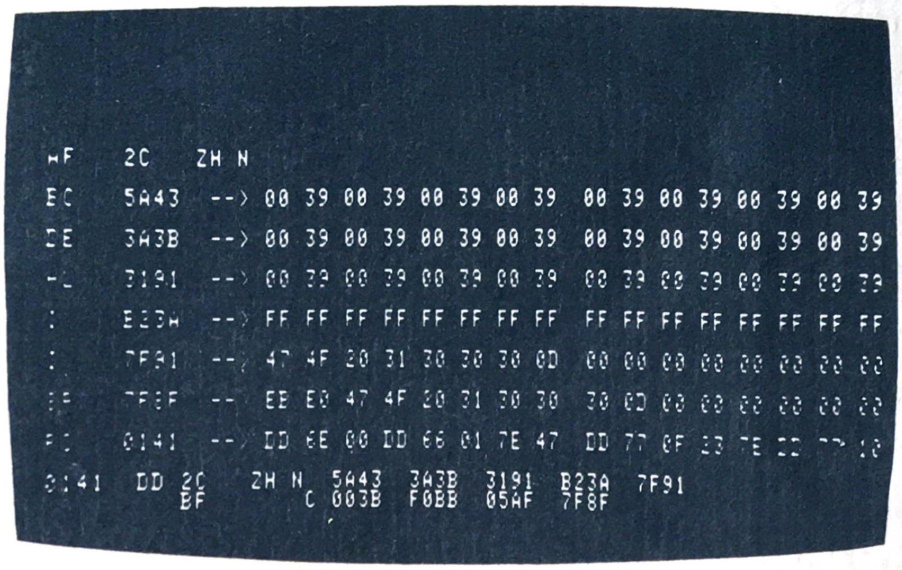
8



```
AF    2C    ZH N
BC    5A43  --> 00 39 00 39 00 39 00 39  00 39 00 39 00 39 00 39
DE    3A3B  --> 00 39 00 39 00 39 00 39  00 39 00 39 00 39 00 39
HL    3191  --> 00 39 00 39 00 39 00 39  00 39 00 39 00 39 00 39
IX    B23A  --> FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
IY    7F91  --> 47 4F 20 31 30 30 30 0D  00 00 00 00 00 00 00 00
SP    7F8F  --> EB E0 47 4F 20 31 30 30  30 0D 00 00 00 00 00 00
PC    0141  --> DD 6E 00 DD 66 01 7E 47  DD 77 0F 23 7E DD 77 10
0141  DD 2C    ZH N  5A43  3A3B  3191  B23A  7F91
         BF         C 003B  F0BB  05AF  7F8F
```

DEBUG

A Program for Debugging Machine Language Programs for the EXIDY SORCERER

By Bob Pierce

- **Single step processing of machine language instructions**
- **Multiple breakpoints**
- **Option to execute CALLS in a single step**
- **Several display options**
- **Relocatable**

A Product of QS QUALITY SOFTWARE

Published By Quality Software

LOADING INSTRUCTIONS

To run DEBUG follow these instructions. If you are in BASIC, enter the Sorcerer Monitor by typing BYE. Then type

```
> LOG DEBUG
```

If you are going to load another program after you have loaded DEBUG, it is better to load DEBUG without running it by typing

```
> LO DEBUG
```

Once your machine language program is loaded, you may execute DEBUG from the Monitor by typing

```
> GO 1000
```

You may exit DEBUG and return to BASIC or the Monitor by pressing RESET; however, occasionally this can destroy the program.

GETTING FAMILIAR WITH DEBUG

If you have just received a copy of DEBUG, you might like to follow this example to become acquainted with some of the capabilities of DEBUG. Load the program as outlined in the LOADING INSTRUCTIONS. Once the copyright message appears, enter Y to the question DO YOU WISH TO MOVE DEBUG? Enter 100(RETURN) when RELOCATION ADDRESS appears. A message stating that you have moved DEBUG to 0100-0FAF will appear along with the Sorcerer Monitor > prompt. This creates a second copy of DEBUG. The original will still be located at 1000-1EAF. Enter the Monitor command GO 1000(RETURN). The same copyright message will appear. This time answer the question about moving DEBUG with an N. Enter 141(RETURN) to the START ADDRESS prompt.

At this point you will have a display of the contents of registers and memory similar to that on the instruction booklet cover. Observe the values in the registers H and L. The first two instructions pointed to by the Program Counter, PC, will be DD instructions that load L and then H. These registers will be loaded with the first two values that are pointed to by the IX register. Press the space bar twice and observe that the values in L and H become the values pointed to by IX. Note that the Program Counter has increased to 0147 and points to the one byte instruction 7E, i.e. LD A, (HL). This instruction loads into A the value that is pointed to by the pair HL. Hit the SPACE BAR a third time and observe that register A contains this byte.

Enter the Command Mode by pressing C and observe the > on the bottom line verifying this. Enter 1000V(RETURN) and observe that 48 bytes of memory starting at address 1000 are displayed. Press ; several times and follow by pressing – several times to see the display incremented and decremented by 48 bytes. Press the SPACE BAR to return to the Command Mode. Enter 1000P to see a page of data from memory displayed. Again use the ; and – keys to change the display.

Return to the Command Mode by pressing the SPACE BAR. Enter 1F00L to load memory at 1F00. Enter AB CD EF(RETURN). View memory at 1F00 as before to see that these three bytes have been loaded into memory. Return to the Command Mode by pressing SPACE BAR and press R. Enter A=77(RE-

TURN) and notice that register A will contain 77. Now press G and observe that DEBUG wants a new start address. Enter the same start address 141(RETURN) as before.

To change the display mode press C and then M. Press S to obtain the scrolling display mode. Exit the Command Mode by pressing X. Use the SPACE BAR and then the REPT key to scroll the display until the display is filled. As many as 28 instructions along with register contents can be viewed simultaneously.

This example should have given you the opportunity to gain familiarity with some of the power that DEBUG can give you in analyzing your machine language programs. Be sure to read the section on entering hex data.

RELOCATING DEBUG

You may wish to relocate DEBUG either because its current location in memory (1000-1EAF) conflicts with your machine language program or because you wish to have it adjacent to your machine language program so that you can easily load/save both DEBUG and your program simultaneously. To relocate DEBUG simply answer the question 'DO YOU WISH TO MOVE DEBUG? (Y/N)' with a Y. You are then asked for the RELOCATION ADDRESS. Enter the address at which you wish DEBUG to begin and press the RETURN key. DEBUG occupies 0EAF bytes of memory and will be moved to the new address that you entered. All instructions such as CALL and JP that depend on DEBUG's memory location will be automatically modified. If there is insufficient RAM to relocate DEBUG at the address you requested, you will receive a very brief message 'OUT OF RAM' and you will be asked for the object program START ADDRESS. If the move is successful, control will go back to the Sorcerer Monitor. Remember that if you save the relocated DEBUG on cassette, you must SET X=(new start address of DEBUG) prior to saving if you want to use LOG to load the program.

GETTING STARTED

If you have used the Sorcerer Monitor command LOG to load DEBUG, you will see the copyright message and the question 'DO YOU WISH TO MOVE DEBUG (Y/N)?'. You may also arrive at this point by entering GO 1000 (unless DEBUG has been relocated) from the monitor after loading DEBUG. If you answer this question with an N, you will be asked to enter the START ADDRESS. Enter the hexadecimal address at which you wish single step processing to begin. If you make an error in entering the hex address, see the section on entering hex data to correct the error. After the start address is entered, the registers and some memory are displayed and single step processing begins.

DISPLAY OF REGISTERS

With the full display of registers you will be able to see the contents of all of the principal Z80 registers along with 16 bytes of memory at the locations to which the BC,DE,HL,IX,IY,SP, and PC registers point. At the top are seen the A register and the flags. The symbols and bit locations for the flags are given below.

SYMBOL	FLAG	BIT LOCATION
M	Sign Flag	7
Z	Zero Flag	6
H	Half-Carry Flag	4
P	Parity/Overflow Flag	2
N	Add/Subtract Flag	1
C	Carry Flag	0

Whenever a flag bit is set (that is, is on), the symbol designating the flag bit will appear; otherwise the space for the symbol will be blank.

Below the A register and flags will be seen the BC register and the contents of this register pair. To the right of the contents of the BC register an arrow appears followed by 16 bytes of data. These data are the 16 bytes of memory starting at the memory location that is contained in the BC register pair. The same applies to the other registers. The last 16 bit register, PC, is the Program Counter. The bytes to the right of the arrow on the PC row represent the current instruction (not yet executed) and the instructions following this one in memory.

Below the row containing the Program Counter data are two rows of register data that always appear during single step processing. These data represent the contents of registers. For a diagram defining the register contents displayed in these two rows, see the Map of Register Contents at the bottom of the last page of this booklet.

SINGLE STEP PROCESSING

In this mode you can execute your machine language program one instruction at a time. This is done by pressing the SPACE BAR or RETURN key. In addition you can execute CALL instructions in a single step by pressing S. However, if the CALL instruction is in ROM, this will not work. This key can also be used for executing instructions other than CALL.

To change display modes, view memory, load registers or memory, set breakpoints or restart single step processing, you need to enter the Command Mode. This is done by pressing C. Finally, if you have previously set breakpoints, pressing W executes the object program from the current instruction until a breakpoint is encountered.

ENTERING HEX DATA

To enter hexadecimal addresses for data into DEBUG you should keep in mind the following guidelines. The last four hex values entered are what DEBUG uses in inputting addresses, and the last two hex values are used in inputting one byte values. Thus, in response to START ADDRESS, if one enters

```
*1D1C24(RETURN)
```

DEBUG interprets this to be 1C24. Back spacing will not work.

COMMAND MODE

You are prompted that you are in the Command Mode by the presence of a > symbol on the bottom line. The acceptable character inputs in the Command Mode are X,K,W,V,P,R,L,M and G. For a description of what happens when one

of these is entered, see the Command Summary on the last page. Note that when you wish to view memory or load memory, you must enter the hex address first followed by the command (V,P, or L) with no spaces. Thus, entering

```
107CL
```

results in entering the Load subcommand mode with entries beginning at the address 107C.

BREAKPOINT SUBCOMMAND MODE

This mode is entered by typing K while in the Command Mode. Enter up to five hex addresses separated by a single space or comma. Note that two successive spaces will result in locating one breakpoint at 0000. Upon hitting the RETURN key, the object program will be executed starting at the current address until one of the breakpoints is encountered. When a breakpoint is encountered, control will be passed to the single step processing mode.

Caution should be exercised in entering breakpoints because the object program may blow up before a breakpoint is encountered. The method that DEBUG uses in establishing breakpoints is to insert three byte jump instructions at the specified breakpoints. These jump instructions cause control to be returned to DEBUG. When control is returned to DEBUG in this way, the jump instructions are replaced by the code originally in the object program. If one or more breakpoints are inserted and none is encountered, and your program does not blow up, then hitting RESET and restarting DEBUG will **not** remove these jump instructions and the object program will be polluted. This condition can sometimes be detected during single step processing by a sudden shift in instruction addresses from the object program to the DEBUG program. One should be aware that breakpoints cannot be entered into ROM. Execution of the object program before the breakpoints are encountered will be just as fast as though DEBUG were not in service.

The last set of breakpoints that were entered are stored by DEBUG and are available to be used again by means of the W command (see below). The breakpoints are lost when a new start address is entered.

W COMMAND (REPEAT PREVIOUS BREAKPOINTS)

If you have already established one or more breakpoints, then pressing W while in either the single step processing mode or the Command Mode causes the previous breakpoint(s) to be automatically re-entered and the object program to be executed from the current address until one of the breakpoints is encountered. The effect of the W command is the same as re-entering the previous breakpoint(s) and pressing RETURN. The same caution should be exercised with the W command as is exercised with breakpoints. The purpose of the W command is to free the programmer of the task of repeatedly inserting the same breakpoints. This can be especially useful in debugging routines which contain loops.

VIEW MEMORY SUBCOMMAND MODE

This mode is entered by inputting hhhhV while in the Command Mode which will result in displaying 48 bytes of memory starting at the hex address hhhh. Once in the subcommand mode, pressing ; displays the next 48 bytes and pressing — displays the previous 48 bytes. Pressing any other key returns control to the Command Mode.

PAGE MEMORY SUBCOMMAND MODE

This mode is entered by inputting hhhhP while in the Command Mode. This mode is similar to the View Subcommand Mode with the exception that ; increases the memory addresses displayed by 256 bytes and — decreases them by 256 bytes.

REGISTER MODIFICATION SUBCOMMAND MODE

This mode is entered by pressing R while in the Command Mode. To modify the contents of a register, type

*r=hh(RETURN)

where r is a register and hh is a one byte hex number. The acceptable values for r are A,F,B,C,D,E,H,L,IX,IX+1,IY,IY+1,SP,SP+1, A',F',B',C',D',E',H', and L'. Note that entering IX=hh modifies the lower order byte of the two byte address contained in the index register IX and that entering IX+1=hh affects the higher order byte. The same rule applies to IY and the stack pointer SP. To modify the program counter PC, simply return to the START ADDRESS prompt by hitting G while in the Command Mode, then type in the new PC address. To exit the Register Modification Mode, type X. This applies at any point other than when I has been entered. If X is entered after I, DEBUG interprets this to mean that you wish to modify the contents of the IX register.

LOAD MEMORY SUBCOMMAND MODE

This mode is entered by inputting hhhhL while in the Command Mode. Once in this mode entering a series of two hexadecimal digit bytes separated by single spaces will cause those bytes to be loaded into memory starting at the address hhhh. If you make an error with an entry, remember that DEBUG accepts the last two hex entries in a series as the input. Thus, if one enters

6B 7C7D 12(RETURN)

the bytes 6B,7D,12 will be loaded into memory at the address hhhh specified in the hhhhL command. If you do not enter any hex digits but press RETURN, the byte 00 will be loaded at the specified address. It is strongly recommended that you follow a Load with View to see that memory has been modified as you intended.

DISPLAY SUBCOMMAND MODE

This mode is entered by pressing M while in the Command Mode. Pressing any of the keys F,S, or L while in the Display Subcommand Mode causes a return to the Command Mode and establishes the single step processing display mode. These modes are

F: The **full** display as seen on the front page of this booklet. This is the default mode and is set at each restart of DEBUG.

S: The display **scrolls** as instructions are executed, displaying a history of the contents of the non-prime registers.

L: Single step processing modified only the bottom two rows of register data. This **low** mode is designed for debugging machine language programs that cause graphics to be displayed. It is also the fastest of the three display modes.

INPUTS TO OBJECT PROGRAM

If your object program has a CALL to E009, the location of the monitor input subroutine, DEBUG will detect this CALL and request an input. Upon receiving any input, DEBUG will load the value in the A register and continue with the next instruction.

BREAKPOINT(S) SUBCOMMAND SUMMARY

SUBCOMMAND	EXAMPLE	WHAT IT DOES
ADR1, ADR2, . . ., ADR5	*1B36,1CDF	Sets breakpoint address(es)
RETURN	RETURN	Executes program from current address until one of specified breakpoints is encountered
X	*1B36X	Returns to Command Mode with no breakpoints set

VIEW MEMORY SUBCOMMAND SUMMARY

SUBCOMMAND	EXAMPLE	WHAT IT DOES
;	;	Displays next 48 bytes of memory
—	—	Displays previous 48 bytes of memory
Any other key	X	Returns to Command Mode

PAGE MEMORY SUBCOMMAND SUMMARY

SUBCOMMAND	EXAMPLE	WHAT IT DOES
;	;	Displays next 256 bytes of memory
—	—	Displays previous 256 bytes of memory
Any other key	X	Returns to Command Mode

REGISTER MODIFICATION SUBCOMMAND SUMMARY

SUBCOMMAND	EXAMPLE	WHAT IT DOES
r=hh	*L'=78	Loads register r with the value hh
X	*A=X	Returns to Command Mode

LOAD MEMORY SUBCOMMAND SUMMARY

SUBCOMMAND	EXAMPLE	WHAT IT DOES
hhƀ[hhƀ[hhƀ. . .]]	07 11	Enters data into memory at the previously specified location
RETURN	RETURN	Returns to Command Mode after entering data

DISPLAY SUBCOMMAND SUMMARY

SUBCOMMAND	EXAMPLE	WHAT IT DOES
F	*F	Sets display mode to **full** and returns to Command Mode
S	*S	Sets display mode to **scrolling** and returns to Command Mode
L	*L	Sets display mode to **low** and returns to Command Mode

SINGLE STEP PROCESSING KEY PRESS SUMMARY

KEY PRESS	EXAMPLE	WHAT IT DOES
SPACE BAR or RETURN key	SPACE BAR or RETURN key	Executes current instruction
S	S	Executes current instruction; if instruction is a CALL, executes entire subroutine in one step
C	C	Enters Command Mode
W	W	Executes program from current address until one of previously set breakpoints is encountered

DEBUG COMMAND SUMMARY

COMMAND	EXAMPLE	WHAT IT DOES
X	>X	Exits Command Mode and returns to single step processing
K	>K	Enters Breakpoint Subcommand Mode
W	>W	Executes program from current address until one of previously set breakpoints is encountered
hhhhV	>1B6CV	Displays 48 bytes of memory beginning at specified address and enters View Subcommand Mode
hhhhP	>1000P	Displays a Page of memory beginning at specified address and enters Page Subcommand Mode
R	>R	Enters Register Modification Subcommand Mode
hhhhL	>1F08L	Enters Load Memory Subcommand Mode
M	>M	Enters Display Subcommand Mode
G	>G	Returns to 'START ADDRESS' message. Requests entry of new single step processing address

MAP OF REGISTER CONTENTS

program counter	first byte of current instruction	A	flags	BC	DE	HL	IX	IY
1044	DD	2C	ZH N	5A43	3A3B	3191	B32A	7F91
		FF	M HP C	FFFF	FF08	FFFF	7F8F	
		A'	'flags	B'C'	D'E'	H'L'	SP	

QS QUALITY SOFTWARE
6660 Reseda Blvd., Suite 105, Reseda, Ca. 91335
(213) 344-6599