ZSID
Z-80 Debugger for CP/M

by Alan R. Miller

One of the standard programs provided with CP/M is DDT (dynamic debugging tool), an independent program used to test and debug other programs.

SID (symbolic instruction debugger) is a more advanced debugger than DDT. The regular version operates on both the 8080 and Z-80, but the symbolic features refer only to the standard Intel 8080 assembly-language mnemonics. ZSID is a Z-80 version of SID utilizing the official Zilog mnemonics. Since ZSID is written with Z-80 instructions, it will not operate on an 8080 machine.

CP/M programs normally execute in the transient program areas (TPA) starting at 100 hex. When SID is executed, it is initially loaded into the TPA, then automatically relocates itself to the top of the usable CP/M memory. This leaves the beginning of the TPA available for the testing of other programs. After SID is in place, it can be directed to load a separate program into the TPA. Alternately, a separate program can be loaded along with SID by including the program name in the command line. For example, the statement:

```
B>A:SID PAYROLL.COM
```

will first load SID, then SID will load the program 'payroll.com.' If the program is a hex file, then SID will decode it, placing it where it belongs.

Monitor Features

SID contains most of the usual monitor features. For example, a command of: 'D100, 17f' will display memory locations from 100 to 17f hex. Each location will be given in hex and also in Ascii if the characters are printable. DDT and the 8080 version of SID give the Ascii values at the end of the same line as the hex code. ZSID, however, places the Ascii characters on the next line under the corresponding hex code. This format is more convenient for narrow video screens.

A block of memory can be moved with the M command. S allows the user to change (set) memory locations. Hexadecimal addition and subtraction can be performed with the H command. The X command allows the user to examine and alter the CPU registers. These include the general-purpose registers, the alternate register set, the flag register, the program counter, the stack pointer, and the X and Y index registers. There is no search command available in the debugger, nor is there a command for inputting or outputting through an I/O port.

The L command produces a symbolic disassembly of the machine code using the official Zilog mnemonics. The A command allows assembly-language mnemonics to be coded directly. Z-80 assemblers are not consistent in their mnemonic sets. Consider the subtract instructions:

```
SBC   A,C
SBB   A,C
```

The first is the official Zilog mnemonic for an 8-bit subtraction with borrow. The second is also allowed by some assemblers, since it resembles the 8080 mnemonic. But ZSID will not accept the second version during the A operation.

Another Z-80 peculiarity occurs with the set of addition and subtraction mnemonics. Zilog uses two operands for addition and subtraction operations in both 16-bit and 8-bit versions. However, only one operand is used for the 8-bit 'sub' instruction since there is no corresponding 16-bit operation.

16-bit		8-bit	
ADC	HL,DE	ADC	A,E
ADD	HL,DE	ADD	A,E
SBB	HL,DE	SBB	A,E
		SUB	E

Since the accumulator is always the destination register for the 8-bit operations, the argument is not really needed. For this reason, some assemblers do not require the destination operand for any of the 8-bit add and subtract operations. ZSID, however, requires two operands for all 'adc', 'add,'and 'sbb' instructions. Another common variation allows two operands for 'sub' command:

```
SUB   A,E
```

making the instruction look more like the other add and subtract operations. This form has been incorporated into ZSID although it is not a Zilog mnemonic.

Executing a Program with SID

Several commands allow the user to execute a program under control of SID. Suppose that a program works properly until it reaches the address of 244. The program can be loaded with SID, then the command 'G100,244' can be given. This will start the program at address 100 hex, the first argument, and set a breakpoint at address 244, the second argument. If the program gets to address 244, control will automatically return to SID. This happens because SID replaces the original byte at address 244 with an RST 38H instruction. When control returns to SID, the original byte is restored. Then the user can inspect the CPU registers, change them if desired, and continue execution of the program with another G command.

The P command is similar to the G command; it allows an instruction to be passed a given number of times before the program returns control to SID. This command is convenient for debugging loops. The T command can be used to single-step through a program. This command can also be used to trace back through the previous steps.

Arguments to SID Commands

Several different types of arguments can be given with SID commands. The default radix is hexadecimal,

but this can be changed with a prefix. A # sign precedes a decimal number and an apostrophe is used before an Ascii character. Hex arguments to ZSID are a little different from those of the 8080 version of SID. The latter accepts any valid hex number. With ZSID, however, the first digit of a hex number must be one of the decimal digits 0-9, i.e., a leading zero must be used if the first digit is A-F. This restriction is necessary because the Z-80 mnemonics have a different structure from the 8080 mnemonics.

Consider the 8080 instructions:

```
MOV   A,C
MVI   A,C
```

The first is a register-to-register move; the second instruction loads the value of C hex into the accumulator. The 8080 version of SID can distinguish between the two instructions since the mnemonics are different. The Z-80 equivalent, however, uses the same mnemonic for both instructions:

```
LD    A,C
LD    A,0C
```

and so a leading zero must be used to distinguish the hex value of C from the register name C. Arguments can be expressed relative to the previous address. For example, a command of D100, +5 will display memory from address 100 to 105.

Symbolic References

The Digital Research assemblers MAC and ASM and the Microsoft assembler Macro-80 with its linking loader can produce a separate symbol table. SID can be directed to load this symbol table along with the program being tested. Symbolic references from the symbol table can then be used as arguments to SID. The symbols can be used in three ways.

If preceded by a decimal point, the symbol is used as a pointer. Suppose the symbol table contains the label 'outhz' corresponding to the program address of 163. Then the command 'l.outhx, +#20' will disassemble 21 (decimal) locations starting at the address of 'outhx.' Furthermore, the symbol table entries will be given in the listing whenever possible, both as addresses and as operands to instructions. A typical disassembly, if the symbol table is loaded, might look like this:

```
OUTHX:
  163  CALL      0167     .HEX
  166  LD        A,B
HEX:
  167  AND       OF
```

The value stored at the address corresponding to the symbol can also be obtained. If the symbol is preceded by an at-sign, the 16-bit value at the referenced location is utilized. On the other hand, if the symbol is preceded by an equal sign, the byte value at that location is chosen. Suppose the symbol 'iobyte' has been defined as location 3. Then the current value stored at that address can be obtained with the command:

```
H = IOBYTE
```

Programs with Arguments

Some CP/M programs require one or two arguments on the command line. The statement:

```
A>LIST B:SORT.PAS
```

will direct CP/M to load and execute the file 'list.com' from drive B. Additionally, the filename 'b:sort.pad' is placed in memory at the file-control block starting at 5C hex. Finally, CP/M starts the program 'list' by branching to the address 100 hex. Programs such as 'list' that require arguments on the command line can be debugged with SID. In this case, the first command would be:

```
A>SID LIST.COM LIST.SYM
```

This line loads SID, list and list's symbol table. The next step is to set up the file-control block. The command:

```
#IB:SORT.PAS
```

will write the requested filename into the file-control block. Now the program 'list' can be started up under control of SID. The G, P or T command is given with the appropriate breakpoint so control will return to SID at some point.

When a program is run under control of SID, the stack is initially placed at 100 hex. The program being tested should change the stack pointer to something else if a disk read is to be performed. This step is necessary since CP/M uses the region 80 to FF hex as a buffer for the disk operations.

Undocumented Z-80 Instructions

Purchasers of ZSID are provided with the regular SID user's manual and a handy supplement that summarizes the official Zilog instruction set. A more complete summary of the instructions is available from Zilog. But there are many Z-80 instructions that have not been documented by Zilog. Most of these are 8-bit operations involving one of the two 16-bit index registers. The official instructions all involve 16-bit operations for the index registers.

Suppose that we want to move register C into the low-order byte of the IX register. One way to do this using the official instructions is:

```
PUSH    BC
POP     IX
```

But this method will also move register B into the high-order byte of IX. A more direct way to perform the move is to use one of the undocumented instructions. The assembly language mnemonics which can generate the necessary code are:

```
DEFB    0DDH
LD      L,C
```

The byte DD precedes all of the undocumented 8-bit IX operations and FD precedes all undocumented 8-bit IY operations. Then the appearance of L or H in the remaining part of the instruction is interpreted by the Z-80 CPU as the low or high order byte of the corresponding index register.

The move from register C to IX can be performed with ZSID. First write the value DD into memory with the set command:

```
#S4000
4000  00  DD
4001  00  .
```

Then use the A command to enter the move operation:

```
#A4001
4001  LD  L,C
4002
```

Load register C with the value of 12 using the X command:

```
#XB
BC  0000  12
```

Then display all the CPU registers with the X command. Execute this short program with the G command, being sure to set a breakpoint so control will return to SID:

```
#G4000, 4002
```

Finally, examine the CPU registers with the X command. The low 8-bits of the IX register will now contain the value 12.

All of the regular Z-80 8-bit instructions involving the H and L registers can be used in this way. If the regular instruction is preceded by a DD then the operation is performed on the corresponding half of the IX register. If the byte is FD then the IY register is used. The operations include register-to-register moves:

```
LD    H,r
LD    r,H
LD    L,r
LD    r,L
```

where r is one of the general purpose registers A, B, C, D, E. Registers H and L now refer to the index register. Thus the instruction:

```
DEFB    0DDH
LD      H,L
```

will move the low half of IX into the upper half of IX. Immediate instructions can also be performed. The low half of IY can be loaded with the value of 7 by using the instructions:

```
DEFB    0FDH
LD      L,7
```

The 8-bit arithmetic and logical operations, including the shift and rotate instructions can also be used in this way.

These undocumented Z-80 instructions have been incorporated into Allen Ashley's PDS software. But there are several potential problems with using undocumented codes. The immediate problem is that ZSID will not properly decode the instructions. Also, these instructions might not be available on future versions of the Z-80, especially if the chip is obtained from a second-source supplier. Nevertheless, these 8-bit instructions give the programmer some extra registers that may occasionally be needed.

Despool Program

A separate Despool program is available from Digital Research. This program keeps the printer busy while the user does other things on the video screen. Since the CPU operates so much faster than the peripherals, it is possible for all peripherals to operate at once. Large, main-frame computers and minicomputers typically work this way. With microcomputers only one peripheral usually operates at a time.

The KLH spooler (IA Apr 79) spools the operation of the printer in a variety of ways. It can be operated from Basic, from an editor, or from the systems level. It can only operate with 8-inch soft-sector disks; Despool can operate with any CP/M system and any type of disk. Unfortunately, files can only be spooled from the systems level, and the printer stops momentarily every time the console keyboard is in use. Nevertheless, Digital Research's Despool is better than nothing. □