Finding the Crash Site: A Simple Tracer for the 8080

Sometimes an unorthodox hardware mod can greatly simplify a complex programming task.

Albert S. Woodhull
33 Enfield Road, RFD 2
Amherst MA 01002

Our college has a large Imsai system with a disk-based CP/M software package. We also have an Altair 8800 with Teletype and cassette interfaces, which we use for projects requiring portability, such as laboratory work, theatre lighting and electronic music.

One of the nice things about the big system is the Trace command in the DDT debugging package. This program takes control over the program being debugged and displays the program counter and register contents after every program step. I found myself wanting this capability on the smaller Altair system.

It really came to a head when I couldn't get the Mits BASIC tape to run on the Altair because the non-Mits serial I/O boards used different conventions than were provided for on the tape. To patch the program was more practical than changing the I/O hardware; it seemed the answer was a tracer to find the endless loops that were trying to read the wrong status bits.

Writing a tracing routine looked as though it would be quite a job, but then I recalled other occasions when doing something a little unorthodox with the hardware greatly simplified a software problem. How about using an interrupt to call the tracer routine after every step in the program being debugged? That way the tracer itself would not control execution of the object program; instead, control would alternate between the object program and the tracer.

I thought it was a pretty clever idea until I realized that when the tracer routine reenabled the interrupt on the 8080, the first thing it would trace would be its own RET instruction. Everything considered, it didn't seem very useful.

Hardware Solution

The solution to the problem came from my looking at the specifications for the 8080 and realizing that some of the status signals could help. Since the Altair bus uses an active low interrupt signal, $\overline{\text{PINT}}$, what I needed was a line that would be low most of the time but would be high immediately after the return from the tracer subroutine. The stack status signal, SSTACK, was just right. There couldn't be a simpler hardware modification.

Wires from bus lines 73 ($\overline{\text{PINT}}$) and 98 (SSTACK) were run to the unused AUX switch on the front panel of the Altair.

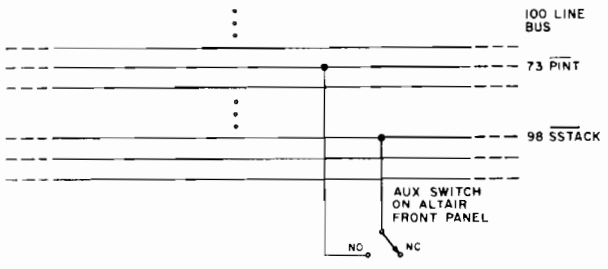

Fig. 1. The drastic modifications needed to implement the tracer scheme.

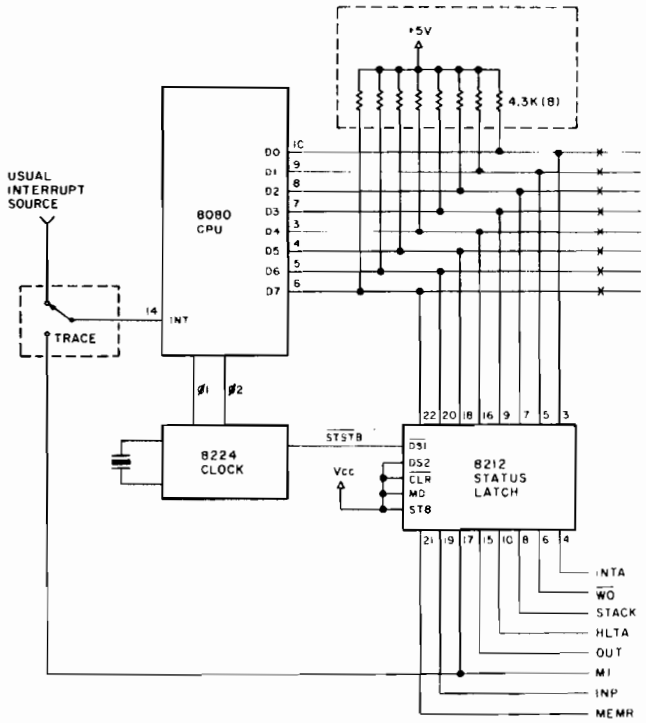

Fig. 2. Many home-brew or non-Altair-compatible 8080 systems will have a CPU interface resembling this circuit. To implement a tracer, pull-up resistors must be added to the unbuffered CPU data bus, and a switch should be installed to allow the M1 status signal to control the 8080 interrupt line.

```
TRACE   SHLD HSTORE     stow the contents of the H and L registers
        POP H           get the return address (program counter when in-
                        terrupted) off the stack
        PUSH PSW        stow accumulator and flags
        MOV A,H
        CALL OCTPRINT   print high byte of address
        MOV A,L
        CALL OCTPRINT   print low byte of address
        MVI A,040₈
        CALL ASCPRINT   print an extra space for clarity
        POP PSW         restore accumulator and flags
        PUSH H          put return address back onto stack
        LHLD HSTORE     restore H and L registers
        EI              enable interrupt
        RET
HSTORE  DB              two bytes reserved for storing con-
        DB              tents of H and L registers
START   EI              to start tracing, load the address of
        JMP OBJ         the start of the object program here
```

Program A. A TRACE program for the 8080. A jump to the start of the TRACE routine must be stored at location 0-070₈. Two additional subroutines are needed. ASCPRINT interprets the contents of the accumulator as an ASCII character and prints or displays it. This subroutine should also take care of bookkeeping and insert carriage returns and line feeds as needed. OCTPRINT unpacks the contents of the accumulator and calls ASCPRINT to show the octal value of the accumulator byte. (See Program B.)

```
OCTPRINT  PUSH PSW      preserve registers
          PUSH B
          MVI B,003₈    set up character counter
          ORA A         clear the carry flag
UNPACK    PUSH PSW      put shifted versions of accumulator on
          RAR           stack from right to left
          RAR           rotate three bits right
          RAR
          DCR B         decrement character counter
          JNZ UNPACK    do this three times
          MVI B,003₈    reload character counter
ASCGEN    POP PSW       get last data first
          ANI 007₈      mask off all but low three bits
          ADI 060₈      convert to ASCII for numbers 0 to 7
          CALL ASCPRINT print it
          DCR B         decrement the counter
          JNZ ASCGEN    do another until done
          MVI A,040₈    load code for space
          CALL ASCPRINT print a space
          POP B         restore registers
          POP PSW
          RET
```

Program B. The OCTPRINT routine for unpacking a byte. If you prefer hexadecimal notation, a similar technique can be used.

With the momentary action switch in its normal open position, the computer runs exactly as it always did. When the switch is pressed, if the computer has executed an enable interrupt (EI) instruction, an interrupt is generated on the beginning of any instruction except those instructions immediately following an instruction that accesses the stack, such as a CALL, RET, PUSH, POP, RST or XTHL instruction.

The pull-up resistors on the data bus ensure the 8080 reads a RST 7 (377_8) when interrupted. This instruction calls the subroutine at location $0\text{-}070_8$. Program A shows a simple tracer routine that will print out the program counter.

Anything so cheap and easy can't be entirely perfect, of course. The major limitation of this scheme is that it doesn't trace every step. I haven't found that a great disadvantage, however. Even though steps following a stack operation are not traced, there is no problem following the flow of a program. It is also not very difficult to write more sophisticated versions of the software—the routine of Program A is shown here because of its simplicity. It doesn't take much more programming to display all the registers or to restrict the printout to particular regions of memory.

Although the particular scheme suggested is very easy to implement on an S-100 bus, other systems will usually allow similar approaches, perhaps using different status signals. For example, on my home-brew 8080 system I have not defined an active low $\overline{\text{PINT}}$ signal. I could have used an extra inverter section to invert the STACK status signal to drive the 8080 INT line, but I didn't have a single extra gate or inverter on my CPU board. Instead I used the M1 status signal to drive the INT line.

In this implementation an interrupt and a call to the tracer are generated immediately after any instruction that makes only a single memory reference. This avoids the problem of calling the tracer immediately after its own RET instruction; yet, as with the version described for the Altair, enough steps in the object program are traced to clearly show what is happening. ■