computer corner

Z-80 *A close look at the three different interrupts available in the Z-80*

WILLIAM BARDEN, JR.

LAST MONTH, WE LOOKED AT THE VARIOUS addressing modes of the Z-80. This month, we'll examine three different ways the Z-80 can be interrupted.

Interrupts

The Z-80 has two interrupt inputs—INT, which typically comes from an external device requesting I/O (*I*nput/*O*utput) service; and a nonmaskable interrupt, NMI. The nonmaskable interrupt is the least sophisticated of the two; so, NMI will be examined first.

The NMI interrupt *cannot* be disabled by the DI (*D*isable *I*nterrupt) instruction. This means that even if the NMI line to the Z-80 is brought to a logic 0 level, the CPU will *always* act on the interrupt. The NMI signals important system conditions that must be acted on immediately, such as system power failure or system reset. Whenever the NMI line is brought to a logic-low level by external circuitry, the CPU executes a RESTART instruction to memory location 0066H. You will recall that a RESTART instruction automatically saves the contents of the program counter by pushing it into the stack, and the RESTART instruction initiated by the NMI interrupt performs the same action. The NMI is not implemented in the 8080 microprocessor.

The INT interrupt operates in three modes, selected by prior execution of one of three special mode instructions—IM0, IM1 or IM2. For this type of interrupt to occur, the interrupt enable flip-flop must be set. The interrupt enable flip-flop is set or reset by two interrupt control instructions, EI and DI (*E*nable *I*nterrupts and *D*isable *I*nterrupts). As in the 8080-based and other microcomputers, there are times when interrupts are permitted and other times when interrupts must be inhibited. An obvious example of a time when interrupts must be disabled is when a previous interrupt has *just* occurred and is in the first stages of being processed. If a second interrupt was to occur while the status of CPU flags and registers is being saved in the stack, the second interrupt might destroy the previous contents of the CPU registers and status flags. Another example of an interrupt-disable period is when the system is first initialized. If an interrupt were allowed to occur before system devices were reset and initialized, a spurious or unexpected interrupt might result and be erroneously processed.

If an IM0 instruction has been executed, the Z-80 is in mode 0 which is identical to the 8080 interrupt mode. If the interrupt enable flip-flop is set and signal $\overline{INT}$ is brought down to a logic 0 level, the CPU enters an interrupt state and signals the interrupting external device by the $\overline{IORQ}$ signal together with the $\overline{M1}$ signal. When the interrupting device receives these two signals, it responds and the Z-80 behaves in a fashion identical to the 8080. A RESTART instruction is jammed onto the data bus. Encoded within the one-byte RESTART instruction is a three-bit field with a value of 0 through 7 and the CPU transfers control to memory location 0, 8, 10H, 18H, 20H, 28H, 30H, or 38H, depending upon the value of the field (0, 1, 2, 3, 4, 5, 6, or 7). At the same time, the CPU saves the contents of the program counter in the stack. The eight locations typically contain jumps to interrupt processing routines elsewhere in memory, since eight bytes is not really enough memory to process most interrupts. At the end of interrupt processing, a Return (RET) instruction pops the address of the interrupted instruction from the stack and transfers control back to the main program at the point of interruption.

When interrupt mode 1 has been initiated by execution of an IM1 instruction, an interrupt on the $\overline{INT}$ input pin while the interrupt enable flip-flop is set causes a RESTART to location 38H. Why is this mode convenient? Because mode 1 needs no external hardware to jam the RESTART instruction onto the data bus at the proper time. The CPU automatically transfers control to the proper location, as in the case of the NMI interrupt. This mode is not implemented in the 8080 microprocessor.

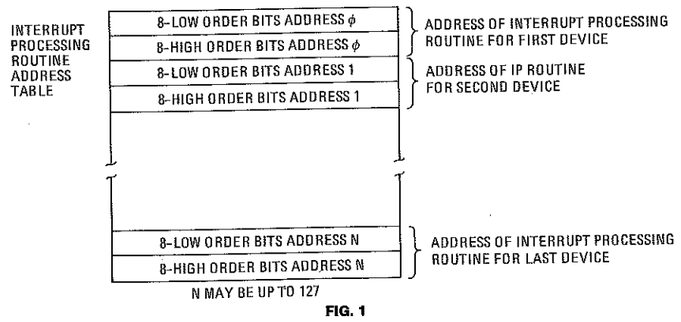


FIG. 1

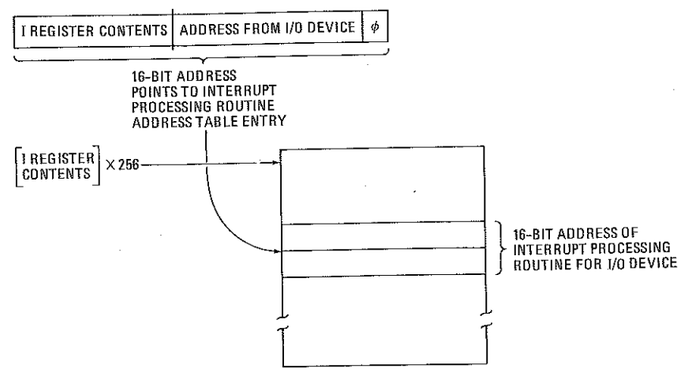
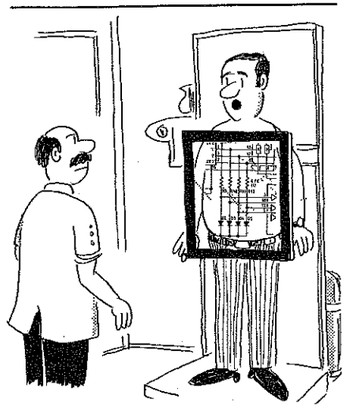


FIG. 2

The remaining interrupt mode, mode 2, is the most powerful interrupt mode of the three. Using this mode, up to 128 interrupt levels can be used in the Z-80 system. A table of addresses representing up to 128 interrupt processing routines is stored anywhere in memory. Figure 1 shows this table, in which each entry consists of two bytes representing the address of the interrupt processing routine. Register I is previously loaded with an eight-bit value representing the address of the start of the table divided by 256_{10}. For example if the table started at 2000H, register I would be loaded with 20H.

With mode 2 previously set by an IM2 instruction and the interrupt enable flip-flop set, an interrupt on the INT pin causes the same IORQ and M1 response as mode 1. The difference is that the interrupting device supplies an eight-bit value representing the lower-order eight bits of the interrupt vector, while register I supplies the eight higher-order bits of the interrupt vector as shown in Fig. 2. The CPU treats the two bytes as a 16-bit memory address and puts the contents of that memory address and that memory address plus one into the program counter, thus effectively transferring control to the interrupt vector address retrieved from the table. Note that the least significant bit of the address from the I/O device must always be a logical 0, so that the resulting address points to the first word of the interrupt vector table entry. Using mode 2, up to 128 external I/O devices could cause 128 unique interrupts with a subsequent transfer of control to 128 different interrupt processing locations. Obviously, external logic is required to properly establish the priority of these devices so that only one interrupt can occur at a time. Mode 2, of course, is not implemented in the 8080.

Next month, we'll discuss how to interface the Z-80 to I/O devices and to additional memory. **R-E**



"That's remarkable, Doc!—How'd you know that I'm an electronics technician?"