Chapter-12

Interrupts

12.1 THE 8085 INTERRUPTS

The 8085 interrupt process is controlled by the Interrupt Enable flip-flop, which is internal to the processor and can be set or reset by using software instructions. If the flip-flop is enabled and the input to the interrupt signal INTR (pin 10) goes high, the microprocessor is interrupted. This is a maskable interrupt and can be disabled. The 8085 a nonmaskable and three additional vectored interrupt signal as well.

The 8085 interrupt process can be described in terms of eight steps:

Step 1: The interrupt process should be enabled by writing the instruction EI in the main program.

Instruction EI (Enable Interrupt)

- This is a 1-byte instruction.
- The instruction sets the Interrupt Enable flip-flop and enables the interrupt process.
- System reset or an interrupt disables the interrupt process.

Instruction DI (Disable Interrupt)

- This is a 1-byte instruction.
- The instruction resets the Interrupt Enable flip-flop and disables the interrupt process.
- It should be included in a program segment where an interrupt from an outside source cannot be tolerated.

Step 2: When the microprocessor is executing a program, it checks the INTR line during the execution of each instruction.

Step 3: If the line INTR is high and the interrupt is enabled, the microprocessor completes the current instruction, disables the Interrupt Enable flip-flop and sends a signal $\overline{\text{INTA}}$ – Interrupt Acknowledge (active low). The processor cannot accept any interrupt request until the interrupt flip-flop is enabled again.

Step 4: The signal $\overline{\text{INTA}}$ is used to insert a restart (RST) instruction (or a Call instruction) through external hardware. The RST instruction is a 1-byte call instruction that transfers the program control to a specific memory location.

Step 5: When the microprocessor receives an RST instruction, it saves the memory address of the next instruction on the stack. The program is transferred to the CALL location.

Step 6: Assuming that the task to be performed is written as a subroutine at the specified location, the processor performs the task. The subroutine is known as service routine.

Step 7: The service routine should include the instruction EI to enable the interrupt again.

Step 8: At the end of the subroutine, the RET instruction retrieves the memory address where the program was interrupted and continues the execution.

12.11 RST (Restart) Instructions

These are 1-byte Call instructions that transfer the program execution to a specific location. The RST instructions are executed in a similar way to that of Call instructions as listed in Table 12.1.

TABLE 12.1

Restart Instructions

	Binary Code									
Mnemonics	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Hex Code	Call Locations
RST 0	1	1	0	0	0	1	1	1	C7	0000
RST 1	1	1	0	0	1	1	1	1	CF	0008
RST 2	1	1	0	1	0	1	1	1	D7	0010
RST 3	1	1	0	1	1	1	1	1	DF	0018
RST 4	1	1	1	0	0	1	1	1	E7	0020
RST 5	1	1	1	0	1	1	1	1	EF	0028
RST 6	1	1	1	1	0	1	1	1	F7	0030
RST 7	1	1	1	1	1	1	1	1	FF	0038

In Figure 12.1, the instruction RST 5 is built using resistors and tri-state buffer.

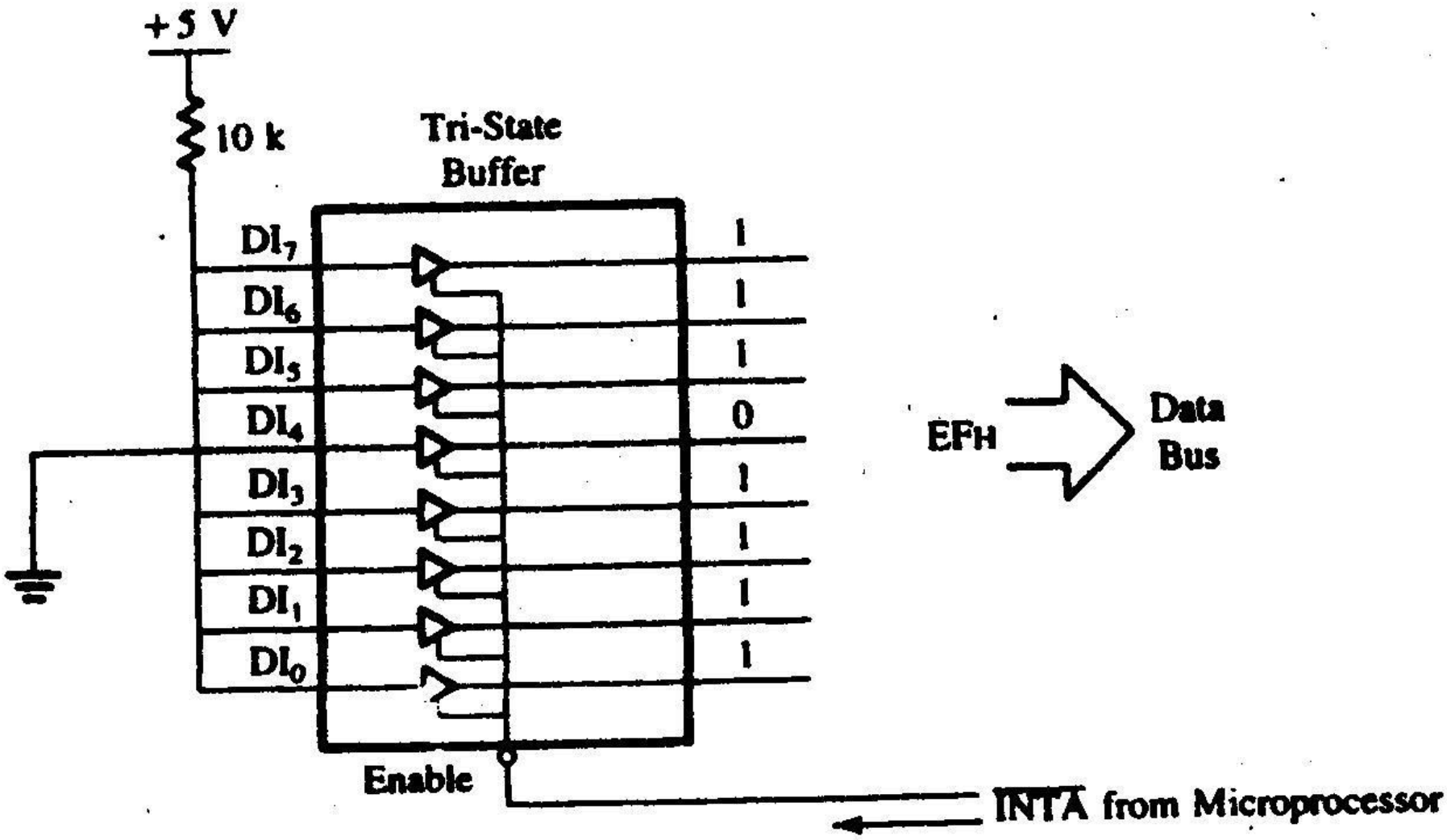

FIGURE 12.1
A Circuit to Implement the Instruction RST 5

12.1.2 Illustration: An Implementation of the 8085 Interrupt

PROBLEM SATEMENT

1. Write a main program to count continuously in binary with a one second delay between each count.
2. Write service routine at XX70H to flash FFH five times when the program is interrupted, with some appropriate delay between each count.

MAIN PROGRAM

```
            LXI SP, XX99H
            EI

            MVI A, 00H
NXTCNT:     OUT PORT1

            MVI C, 01H
            CALL DELAY

            INR A
            JMP NXTCNT
```

Service Routine

```
SERV:   PUSH B
        PUSH PSW

        MVI B, 0AH

        MVI A, 00H
FLASH:  OUT PORT1

        MVI C,01H
        CALL DELAY

        CMA

        DCR B
        JNZ FLASH

        POP PSW
        POP B
        EI
        RET
```

DESCRIPTION OF THE INTERRUPT PROCESS

1. The program will count continuously from 00H to FFH with a one second delay between each count.
2. To interrupt the processor, push the switch, the INTR line goes high.
3. Assuming the switch is pushed when the processor is executing the instruction OUT.

a.
b.
c.
d.
e. The program is transferred to memory location 0028H. The locations 0028-29-2AH should have the following Jump instructions to transfer the program to the service routine

JMP XX70H

(However, you do not have access to write at 0028H in the monitor program. See the next section)

4. The programs jumps to the service routine at XX70H.
5.
6.
7.

TESTING INTERRUPT ON A SINGLE-BOARD COMPUTER SYSTEM

In single-board microcomputers, some restart locations are usually reserved for users, and the system designer provides a Jump instruction at a restart location to jump somewhere in R/W memory. For example, in Intel's SDK-85 system, R/W memory begins at page 20H, and you may find the following instruction in the monitor program at memory location 0028H:

```
0028    JMP 20C2H
```

If instruction RST 5 is inserted as shown in Figure 12.3 , it transfers the program to location 0028H, and the monitor transfers the program from 0028H to location 20C2H. To implement the interrupt shown in Figure 12.3, you need to store the Jump instruction as shown below:

```
20C2    C3      JMP SERV
20C3    70
20C4    20
```

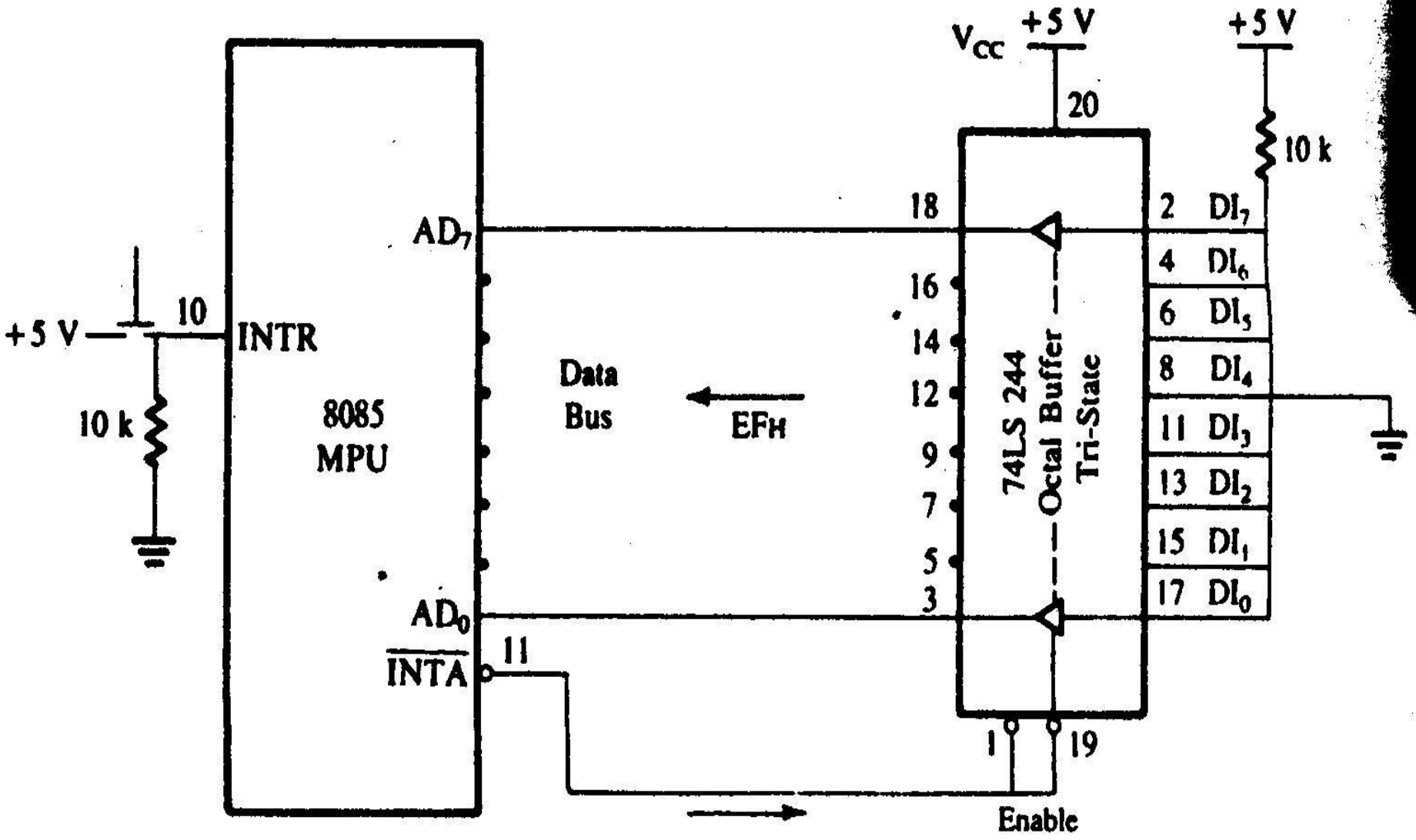

FIGURE 12.3
Schematic to Implement the 8085 Interrupt

ISSUES IN IMPLETING INTERRUPTS

In the above illustration, some questions remain unanswered:

1. *Is there a minimum pulse width required for the INTR signal?*

The microprocessor checks INTR, one clock period before the last T-states of an instruction cycle. In the 8085, the Call instructions require 18 T-states; therefore, the INTR pulse should be high at least for 17.5 T-states. In a system with 3 MHz clock frequency (such as the SDK-85 system), the input pulse to INTR should be at leas 5.8 μs long.

2. *How long can the INTR pulse stay high?*

The INTR pulse can remain high until the interrupt flip-flop is set by the EI instruction in the service routine. If it remains high after the execution of the EI instruction, the processor will be interrupted again, as if it were a new interrupt. In Figure 12.3, the manual push button will keep the INTR high for more than 20 ms; however, the service routine has a delay of 1 second, and the EI instruction is executed at the end of the service routine.

3. *Can the microprocessor be interrupted again before the completion of the first interrupt service routine?*

The answer to this question is determined by the programmer. After the first interrupt, the interrupt process is automatically disabled. In the Illustrative program in section 12.12, the service routine enables the interrupt at the end of the service routine; in this case, the microprocessor cannot be interrupted before the completion of this routine. If the instruction EI is written at the beginning of the routine, the microprocessor can be interrupted again during the service routine.

12.2 8085 VECTORED INTERRUPTS

The 8085 has five interrupt inputs (Figure 12.5). One is called INTR, three are called RST 5.5, RST 6.5, and RST 7.5 respectively, and the fifth is called TRAP, a nonmaskable interrupt. These last four (RSTs and TRAP) are automatically vectored (transferred) to a specific locations on memory page 00H without any external hardware. The do not require the INTA signal or an input port; the necessary hardware is already implemented inside the 8085. These interrupts and their call locations are as follows:

Interrupts		Call Locations
1. TRAP	→	0024H
2. RST 7.5	→	003CH
3. RST 6.5	→	0034H
4. RST 5.5		002CH

The TRAP has highest priority, followed by RST 7.5, 6.5, 5.5, and INTR, in that order; however, the TRAP has lower priority than the Hold signal used for DMA.

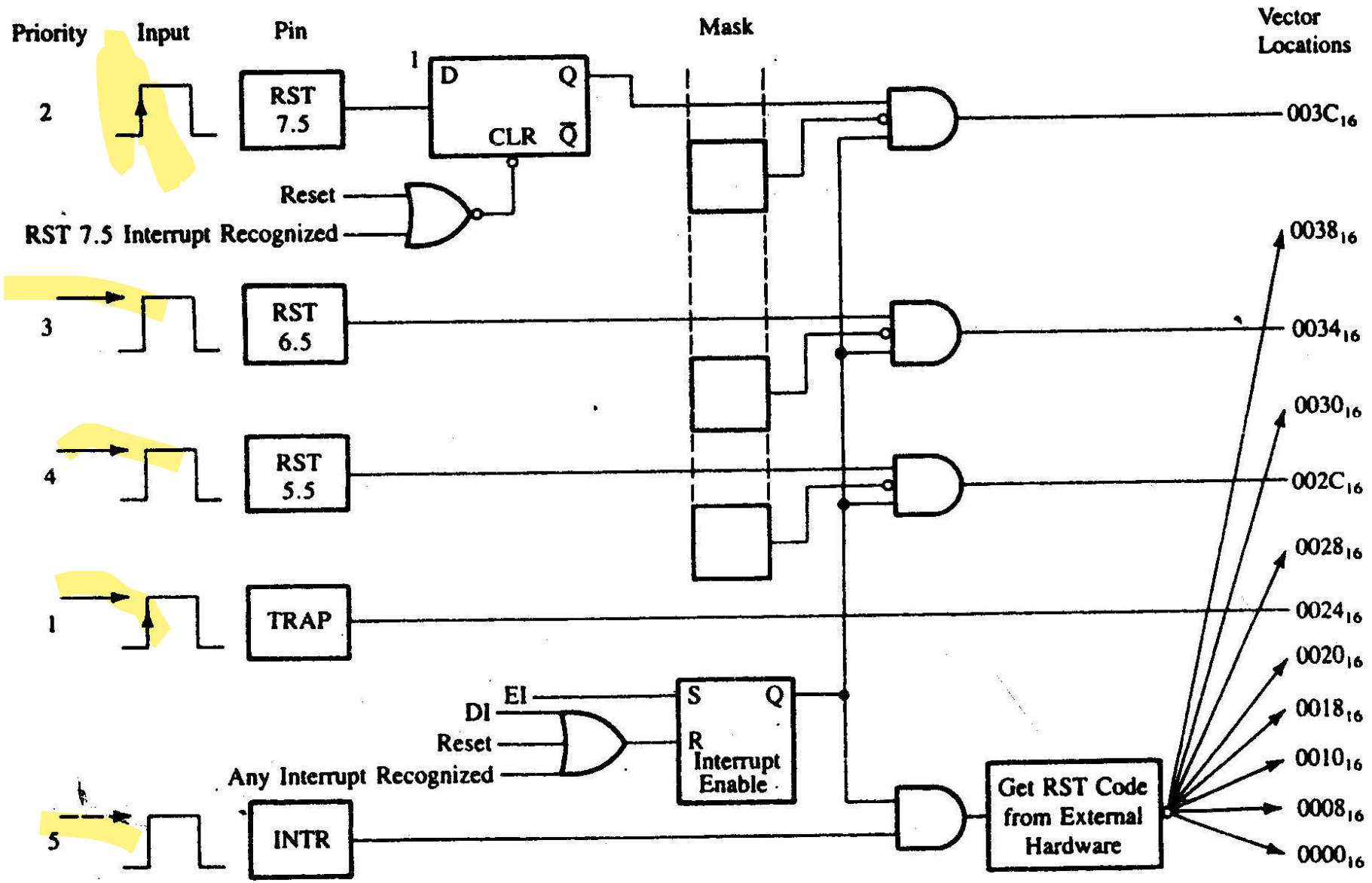

FIGURE 12.5
The 8085 Interrupts and Vector Locations

SOURCE: Intel Corporation, *MCS'80/85 Student Study Guide* (Santa Clara, Calif.: Author, 1979).

12.2.1 TRAP

TRAP, a nonmaskable interrupt known as NMI, is analogous to smoke detector. It has the highest priority among the interrupt signals, it need not to be enabled, and cannot be disabled. It is level-and edge-sensitive, meaning that the input should go high to be acknowledged. It cannot be acknowledged again until it makes transition from high to low to high.

Figure 12.5 shows that when this interrupt is triggered, the program control is transferred to location 0024H without any external hardware or the interrupt enable instruction EI. TRAP is generally used for such critical events as power failure and emergency shut-off.

12.2.2 RST 7.5, 6.5, and 5.5

These maskable interrupt are enabled under program control with two instructions: EI (Enable Interrupt), and SIM (Set Interrupt Mask) described below:

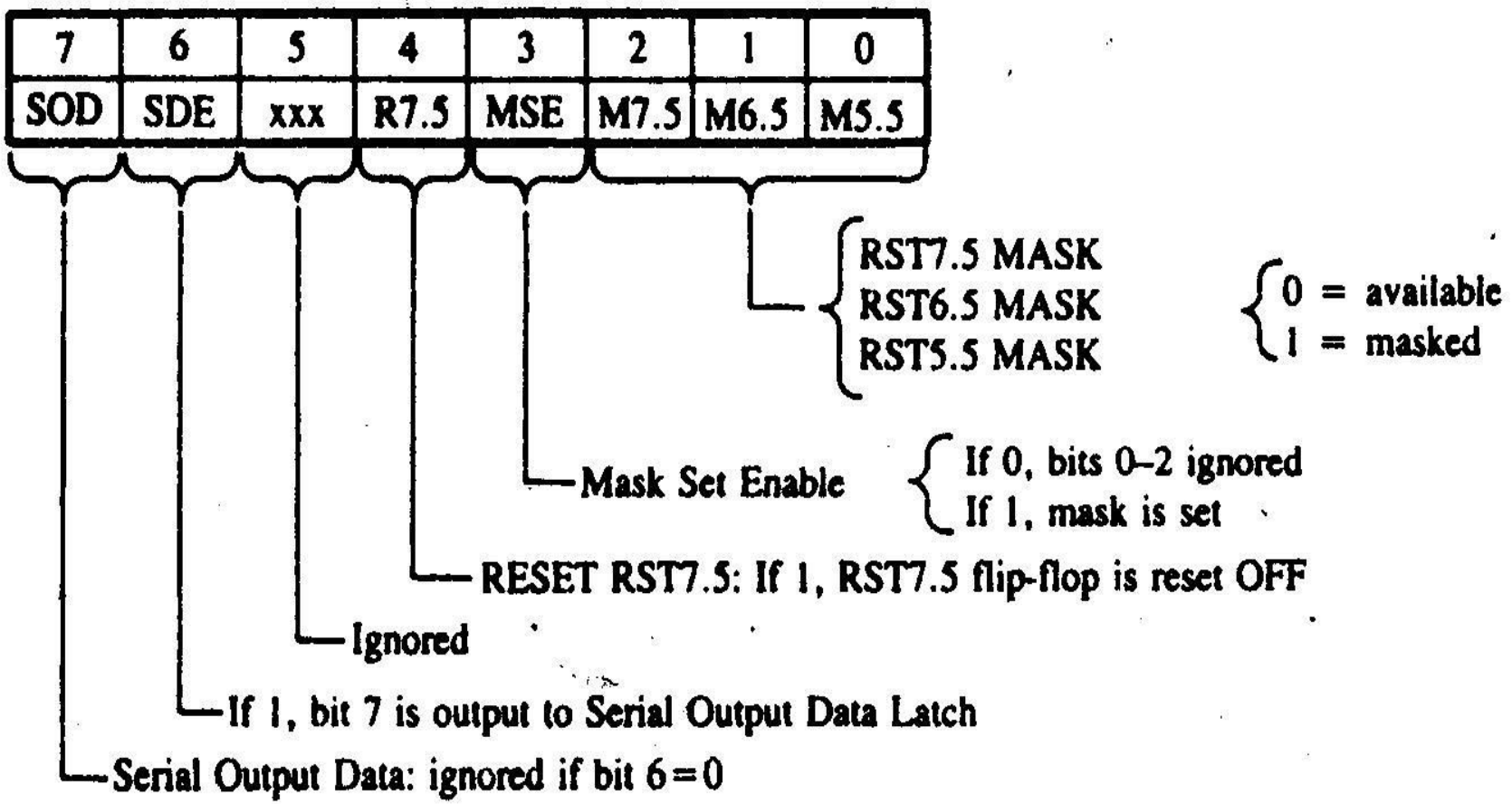

FIGURE 12.6

Interpretation of the Accumulator Bit Pattern for the SIM Instruction

SOURCE: Intel Corporation, *Assembly Language Programming Manual* (Santa Clara, Calif.: Author, 1979), pp. 3–59.

Instruction SIM:

Set Interrupt Mask. This is a 1–byte instruction and can be used for three different functions (Figure 12.6).

- One function is to set for RST 7.5, 6.5, and 5.5 interrupts. This instruction reads the content of the accumulator and enables or disables the interrupts according to the contents of accumulator. Bit D_3 is a control bit and should = 1 for bits D_0, D_1, and D_2 to be effective. Logic 0 on D_0, D_1, and D_2 will enable the corresponding interrupts, and logic 1 will disable the interrupts.

- The second function is to reset RST 7.5 flip-flop (Figure 12.6). Bit D_4 is additional control for RST 7.5. If D_4=1, RST 7.5 is reset. This is used to override (or ignore) RST 7.5 without servicing it.

- The third is to implement serial I/O (Bits D_7 and D_6) and do not affect the interrupts.

The entire interrupt process (except TRAP) is disabled by resetting the Interrupt Enable flip-flop (Figure 12.5). The flip-flop can be reset in one of the three ways:

- Instruction DI
- System Reset
- Recognition of an Interrupt Request

Example: 12.1: Enable all the interrupts in an 8085 system.

```
Instruction:    EI              ; Enable interrupts
                MVI A, 08H      ; To enable RST 7.5, 6.5, and 5.5
                SIM             ; Enable RST 7.5, 6.5, and 5.5
```

Example: 12.2: Reset the 7.5 interrupt from Example 12.1.

```
Instruction:    MVI A, 18H      ; Set D4 = 1
                SIM             ; Reset 7.5 interrupt flip-flop
```

PENDING INTERRUPTS

When one interrupt request is being served, other interrupt request may occur and remain pending. The 8085 has an additional instruction call RIM (Read Interrupt Mask) to sense these pending interrupts (Figure 12.7).

Instruction RIM:

Read Interrupt Mask. This is a 1–byte instruction that can be used for the following functions:

The RIM instruction loads the accumulator with the following information:

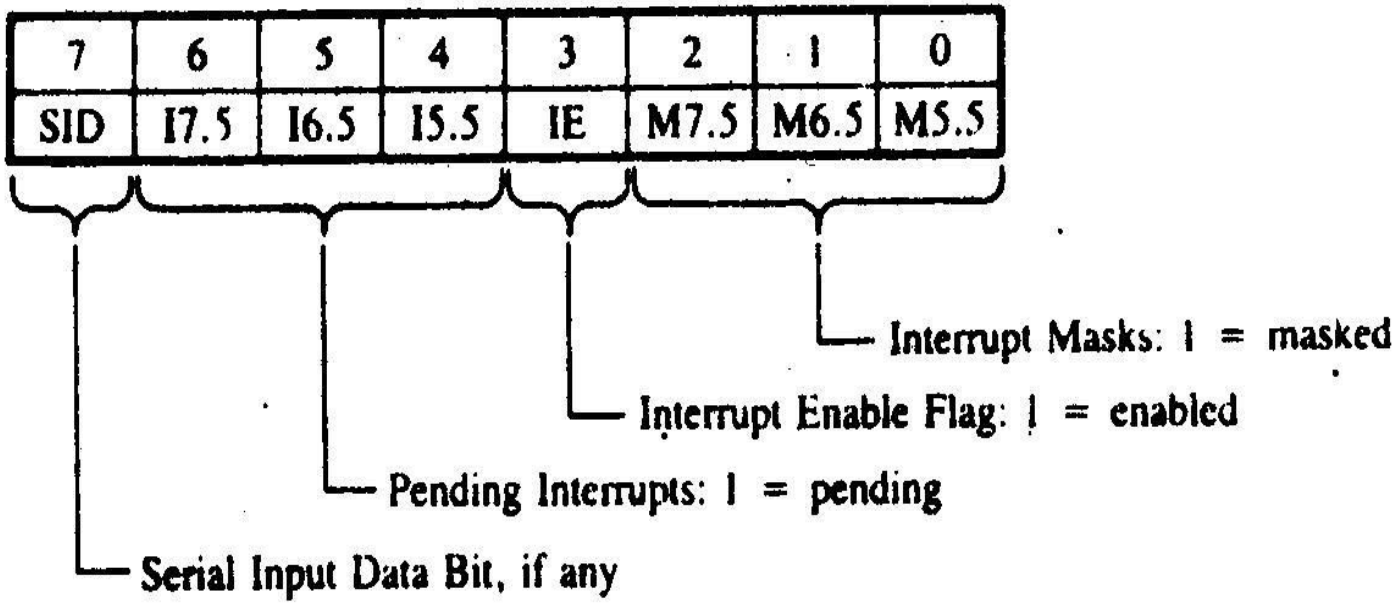

FIGURE 12.7

Interpretation of the Accumulator Bit Pattern for the RIM Instruction

SOURCE: Intel Corporation. *Assembly Language Programming Manual* (Santa Clara, Calif.: Author, 1979), pp. 3-49.

- To read interrupt masks, This instruction loads the accumulator with 8 bits indicating the current status of the interrupt masks (Figure 12.7).
- To identify pending interrupts. Bits D_4, D_5, and D_6 (Figure 12.7) identify the pending interrupts.
- To receive serial data bit D_7 is used (Figure 12.7).

Example: 12.3: Assuming the microprocessor is completing an RST 7.5 interrupt request, check to see if RST 6.5 is pending. If it is pending, enable RST 6.5 without affecting any other interrupts; otherwise return to the main program.

Instruction:

```
RIM
MOV B, A
ANI 20H          ; Check whether RST 6.5 is pending
JNZ NEXT
EI
RET
MOV A,B
ANI 0DH          ; Enable RST 6.5 by sending D1=0
ORI 08H          ; Enable SIM by setting D3=1
SIM
JMP SERV         ; Jump to service routine for RST 6.5
```

12.2.3 Illustration: Interrupt-Driven Clock

PROBLEM STATEMENT

Design a 1-minute timer using a 60 H_z power line as an interrupting source. The output ports should display minutes and seconds in BCD. At the end of the minute, the output ports should continue displaying one minute and zero seconds.

HARDWARE DESCRIPTION

This 1-minute timer is designed with a 60 H_z AC line. The circuit (Figure 12.8) uses a step-down transformer; the 74121 monostable multivibrator, and interrupt pin RST 6.5. After the interrupt the program control is transferred to memory location 0034H in the monitor program.

The AC line with 60 H_z frequency has a period of 16.6 ms; that means it can provide a pulse every sixtieth of a second with 8.3 ms pulse width, which is too long for the interrupt. The interrupt flip-flop is enabled again before 6 μs in the service routine, therefore the pulse should be turned off before the EI instruction in service routine is executed.

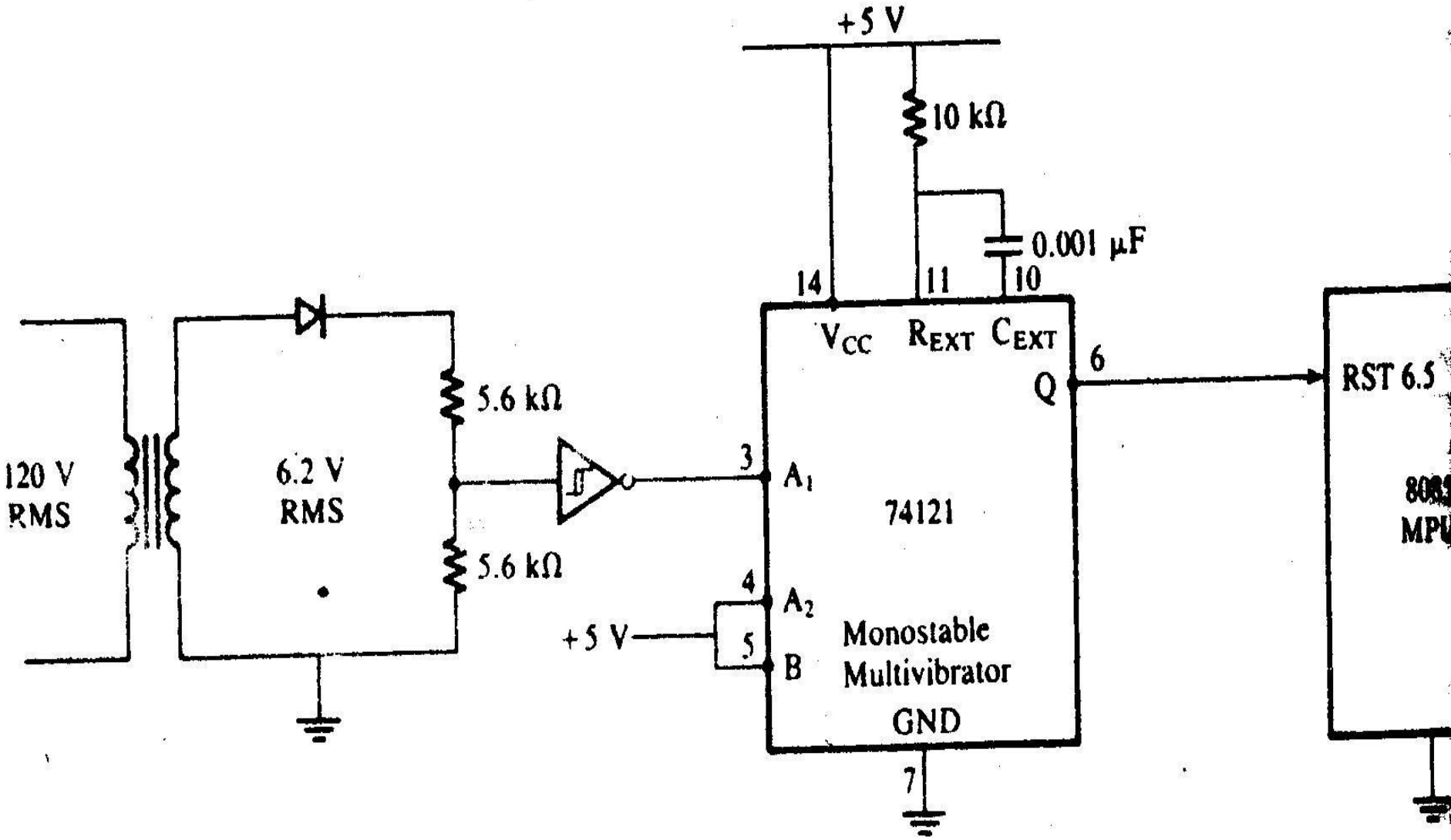

FIGURE 12.8
Schematic of Interrupt-Driven Timer Clock

Monitor Program

```
0034        JMP RWM
```

Main Program

```
            LXI SP, STACK
            RIM
            ORI 08H
            SIM
            LXI B,0000H     ; Set up B for minutes and C for seconds

            MVI D, 3CH      ; Set up D to count 60(10) interrupts
            EI
DISPLAY:    MOV A, B
            OUT PORT1
            MOV A, C
            OUT PORT2
            JMP DISPLAY

RWM:        JMP TIMER       ; This is RST 6.5 vector location 0034H
```

Interrupt Service Routine

```
; Section I
TIMER:      DCR D          ;One interrupt occurred
            EI
            RNZ            ; Has 1 second elapsed? If not, return
;Section II
            DI
            MVI D,3CH      ;1 second is complete; load D to count again
            MOV A,C
            ADD 01H
            DAA            ; Decimal-adjust “Second”
            MOV C,A        ;Save “BCD” seconds
            CPI 60H
            EI
            RNZ

;Section III
            DI
            MVI C,00H      ;60 seconds complete; clear “Seconds” register
            INR B          ; Increment “Minutes” register
            RET
```