Interrupts in 8085

→ The interrupt I/O is a process of data transfer whereby an external device or a peripheral can inform the processor that it is ready for communication and it requests attention.

The process is initiated by an external device and is asynchronous, meaning that it can be initiated at any time without reference to system clock.

→ Interrupt requests are classified in two categories:

(1) Maskable interrupts

(2) Non maskable interrupts.

8085 includes four maskable interrupts and one non maskable interrupt.

→ Among the four maskable interrupts, one is non vectored, which requires external hardware to supply a call location to restart the execution. The other three are vectored to a specific location.

→ The microprocessor can ignore or delay a maskable interrupt request if it is performing some critical task. However, it has to respond to a non maskable interrupt immediately.

The 8085 Interrupt:

→ The 8085 interrupt process is controlled by the Interrupt Enable Flipflop, which is internal to the processor and can be set or reset by using software instructions.

→ If the flipflop is enabled, the input to the interrupt signal INTR (pin 10) goes high, the microprocessor is interrupted. This is a maskable interrupt and can be disabled.

→ 8085 has a non maskable and three additional vectored interrupt signals as well.

8085 interrupt process can be described in eight steps :

(1) The interrupt process should be enabled by writing the instruction EI in the main program. The instruction EI sets the Interrupt Enable flipflop. The instruction DI resets the flip flop and disables the interrupt process.

Instruction EI (Enable Interrupt)

→ 1 byte instruction
→ sets the Interrupt enable flip flop and enables interrupt process
→ System resets or an interrupt disables the interrupt process

Instruction DI (Disable interrupt)

→ 1 byte instruction
→ resets interrupt enable flipflop and disables interrupt.
→ It should be included in a program statement, where an interrupt from an outside source cannot be tolerated.

(2) When the microprocessor is executing a program, it checks the INTR line during the execution of each instruction

(3) If INTR is high and interrupt is enabled, the microprocessor completes the current instruction, disables the interrupt enable flip flop and sends $\overline{INTA}$ signal. The processor cannot accept any interrupt requests until the interrupt enable flip flop is enabled again.

(4) The signal $\overline{INTA}$ is used to insert a restart (RST) instruction through an external hardware.
The RST instruction is a 1-byte call instruction that transfers program control to a specific memory location on page 00H and restarts the execution at that memory location after executing step (5)

(5) When the micro processor receives RST instruction, it saves the memory address of next instruction on the stack. The program is transferred to the CALL instruction.

(6) Assuming the task to be performed is written as a subroutine at the specified locations, the processor performs the tasks. This subroutine is called service routine.

(7) The service routine should include an instruction EI to enable the interrupt again.

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

RST (Restart) Instructions :

→ 8085 includes eight RST instructions

→ These are 1 byte call instructions that transfer the program execution to a specific location on page 00H.

→ They are executed in a similar way to that of CALL instructions.

TABLE 12.1
Restart Instructions

Mnemonics	Binary Code								Hex Code	Call Location in Hex
	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0		
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

Illustration: An implementation of 8085 Interrupt

Problem Statement:

(1) Write a main program to count continuously in binary with a one-second delay between each count.

(2) Write a service routine at XX70H to flash FFH five times when the program is interrupted, with some appropriate delay between each flash.

Sol

Main Program:

Memory Address	Label	Mnemonics	Comments
		LXI SP, XX99H	; Init. SP.
XX00		EI	; enable interrupt process
03			
04		MVI A, 00H	; Initialize count
06	NXTCNT:	OUT PORT1	; display count
08		MVI C, 01H	; parameter for 1-sec delay
0A		CALL DELAY	; wait 1 second
0D		INR A	
0E		JMP NXTCNT	

Delay Routine : Write a delay subroutine for 1 sec.

Service Routine :

Memory Address	Label	Mnemonics	Comments
XX70	SERY:	PUSH	; save contents
XX71		PUSH PSW	
XX72		MVI B, 0AH	; load Reg B for 5 flashes & 5 blanks
XX74		MVI A, 00H	; load 00 to blank display.
XX76	FLASH:	OUT PORT1	
XX78		MVI C, 01H	
XX7A		CALL DELAY	
XX7D		CMA	
XX7E		DCR B	
XX7F		JNZ FLASH	
XX82		POP PSW	
XX83		POP B	
XX84		EI	
XX85		RET	

8085 Vectored Interrupts:

→ The 8085 has five vectored interrupts.

→ (1) INTR
(2) RST 5.5
(3) RST 6.5
(4) RST 7.5
(5) TRAP → Non maskable interrupt.

→

Interrupt	call locations
1. TRAP	0024H
2. RST 7.5	003CH
3. RST 6.5	0034H
4. RST 5.5	002CH

→ TRAP has the highest priority followed by RST 7.5, RST 6.5, RST 5.5 and INTR in that order.

Note: TRAP has a lower priority than the HOLD signal used for DMA.

TRAP:

→ TRAP is a non maskable interrupt (NMI) having the highest priority among the interrupt signals.

→ It cannot be enabled or disabled.

→ It is level and edge sensitive, meaning that TRAP input should go high and stay high to be acknowledged.

→ It cannot be acknowledged again until it makes a transition from high to low to high.

RST 7.5, 6.5, and 5.5 :

→ These are maskable interrupts which can be enabled under program control with two instructions

EI and SIM (Set interrupt mask)

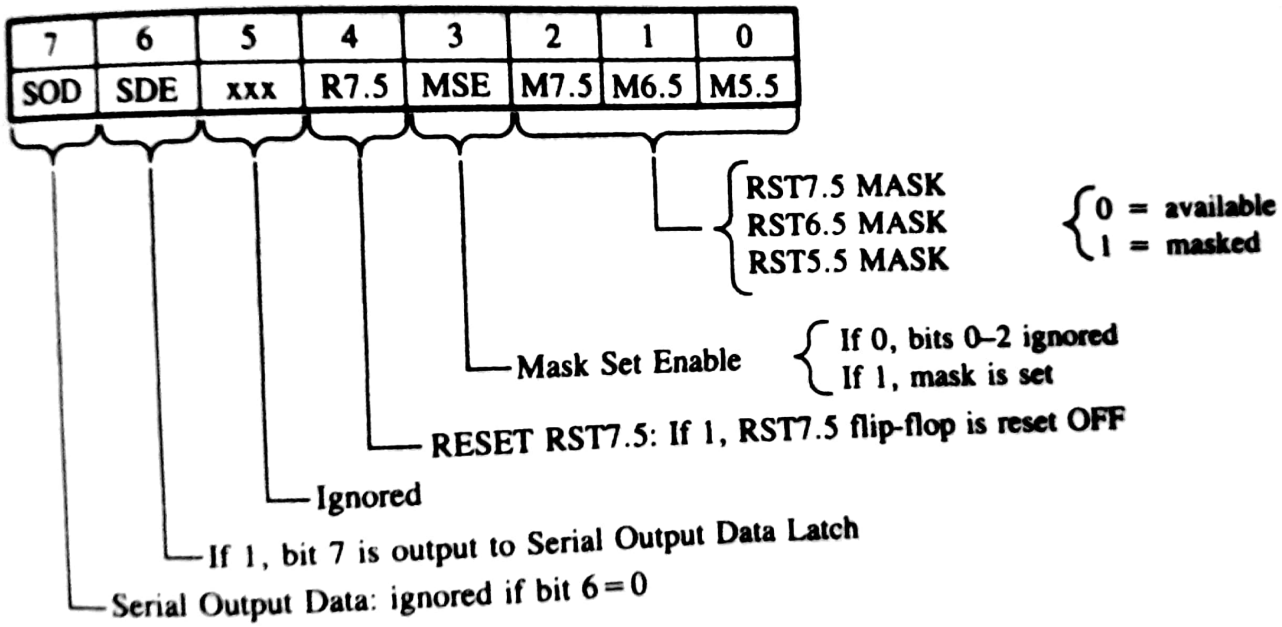

FIGURE 12.6
Interpretation of the Accumulator Bit Pattern for the SIM Instruction

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 mask 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 content of the 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 function is to implement serial I/O (discussed in Chapter 16). Bits D_7 and D_6 of the accumulator are used for serial I/O and do not affect the interrupts. Bit $D_6 = 1$ enables the serial I/O and bit D_7 is used to transmit (output) bits.

Steps in interrupting 8085:

(1) The interrupt process is enabled. The instruction EI sets the interrupt enable flipflop and one of the inputs to the AND gates is set to logic 1.
These AND gates activate the program transfer to various vectored locations.

(2) An appropriate bit pattern is loaded into Accumulator.

(3) If bit $D_3 = 1$, respective interrupts are enabled according to bits $D_2 - D_0$.

(4) RST 7.5, 6.5 and 5.5 are being monitored.

(5) If bit $D_3 = 0$, bits $D_2 - D_0$ have no effect on previous conditions.

(6) Bit $D_4 = 1$; this resets RST 7.5.

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

→ Instruction DI
→ System reset
→ Recognition of an Interrupt Request.

Triggering levels:

(1) RST 7.5 :

This is positive edge sensitive and can be triggered with a short pulse. The request is stored internally by the D flip flop until the microprocessor responds to the request or until it is cleared by Reset or by bit D_4 in the SIM instructions.

(2) RST 6.5 and RST 5.5:

→ These instructions are level sensitive, meaning that triggering level should be on until the microprocessor completes the execution of current instruction. If the microprocessor is unable to respond, they should be stored or held by external hardware.

Ex:-

Enable all the interrupts in an 8085 system.

Sol

```
EI          ; enable interrupts
MVI A, 0BH  ; load bit pattern 0000 1000
SIM         ; Enable RST 7.5, 6.5, 5.5
```

Ex:- Reset the 7.5 interrupt from previous example.

```
MVI A, 18H  ; Set D4 = 1
SIM         ; Reset 7.5 interrupt
              flipflop
```

Pending Interrupts:

→ Because there are several interrupt lines, when one interrupt request is being served, other interrupt requests may occur and remain pending.

→ 8085 has an additional instruction called RIM (Read interrupt Mask) to sense these pending interrupts.

Instruction RIM : Read Interrupt Mask

1 byte instruction that can be used for the following functions :

⇒ To read interrupt masks. This instruction loads the accumulator with 8 bits indicating the current status of interrupt masks.

→ To identify pending interrupts. Bits D_4, D_5 and D_6 identify the pending interrupts.

→ To receive serial Data. Bit D_7 is used to receive serial data.

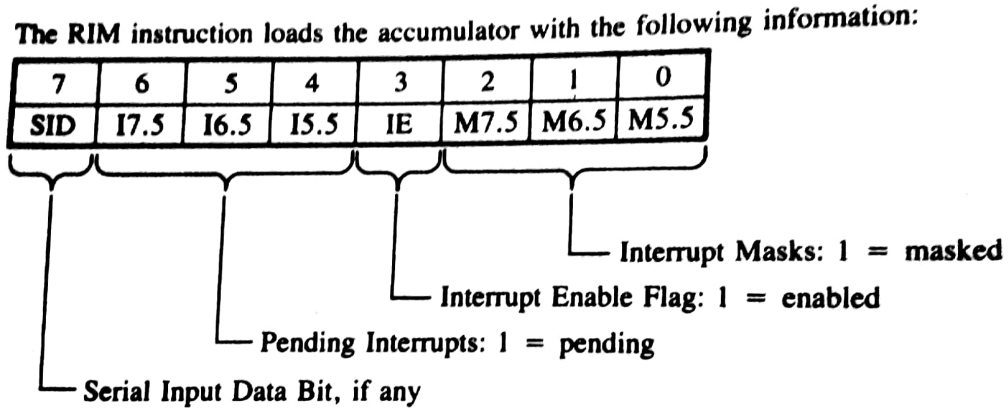

FIGURE 12.7
Interpretation of the Accumulator Bit Pattern for the RIM Instruction

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

Ex:- Assuming 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 main program.

Sol

```
        RIM         ; Read interrupt mask
        MOV B, A    ; Same mask information
        ANI 20H     ; Check whether
                    RST 6.5 is pending
        JNZ NEXT
        EI
        RET         ; RST 6.5 is not pending; return to main
NEXT:   MOV A, B    ; Get bit pattern; RST 6.5 pending
        ANI 0DH     ; enable RST 6.5 by setting D1=0
        ORI 08H     ; Enable SIM by setting D3=1
        SIM
        JMP SERV    ; Jump to service
                    routine RST 6.5.
```