Interrupts

Interrupts

- Interrupt is a process where an external device can get the attention of the microprocessor
  - The process starts from the I/O device
  - The process is asynchronous, means can occur at any time during execution of program.
- In order to communicate with microprocessor and I/O devices either Polling or interrupt method is used.
- An interrupt is considered to be an emergency signal.
- The microprocessor should respond to it as soon as possible.

1. Polling Method

- In polling, μP polls i.e. ask each device in sequence whether it is ready for communication (data transfer).
- If device is ready, then data transfer takes place between device & μP.
- If device is not ready or completed its data transfer, then μP asks the next device in chain.
- Main disadvantage of this method is that most of the time μP remains busy in polling. So some useful tasks get less time to execute.
- This method is useful only if μP has contains few I/O devices.

2. Interrupt Method

- ✓ Interrupt is signal send by an external device to the microprocessor to request the processor to perform a particular task or work.
- ✓ It is a simple routine program that keeps a check for the occurrence of the interrupt.
- ✓ Mainly in the microprocessor based system the interrupts are used for data transfer between the peripheral (I/O) and the microprocessor.
- ✓ If the µP accept the interrupt and send the INTA (active low) signal to the peripheral.
- ✓ When interrupt is received, µP suspends its current activity and upon completion, it resumes the suspended activity.
- ✓ The processor executes an interrupt service routine (ISR) addressed in program counter.
- ✓ It returned to main program by RET instruction.
- ✓ Advantage is that µP need not waste time in polling the devices.

Interrupt Process

1. When the MPU is executing a program it checks all the interrupt lines during the execution of each instruction.

2. If any Interrupt line is enabled, the processor completes the current going instruction execution.

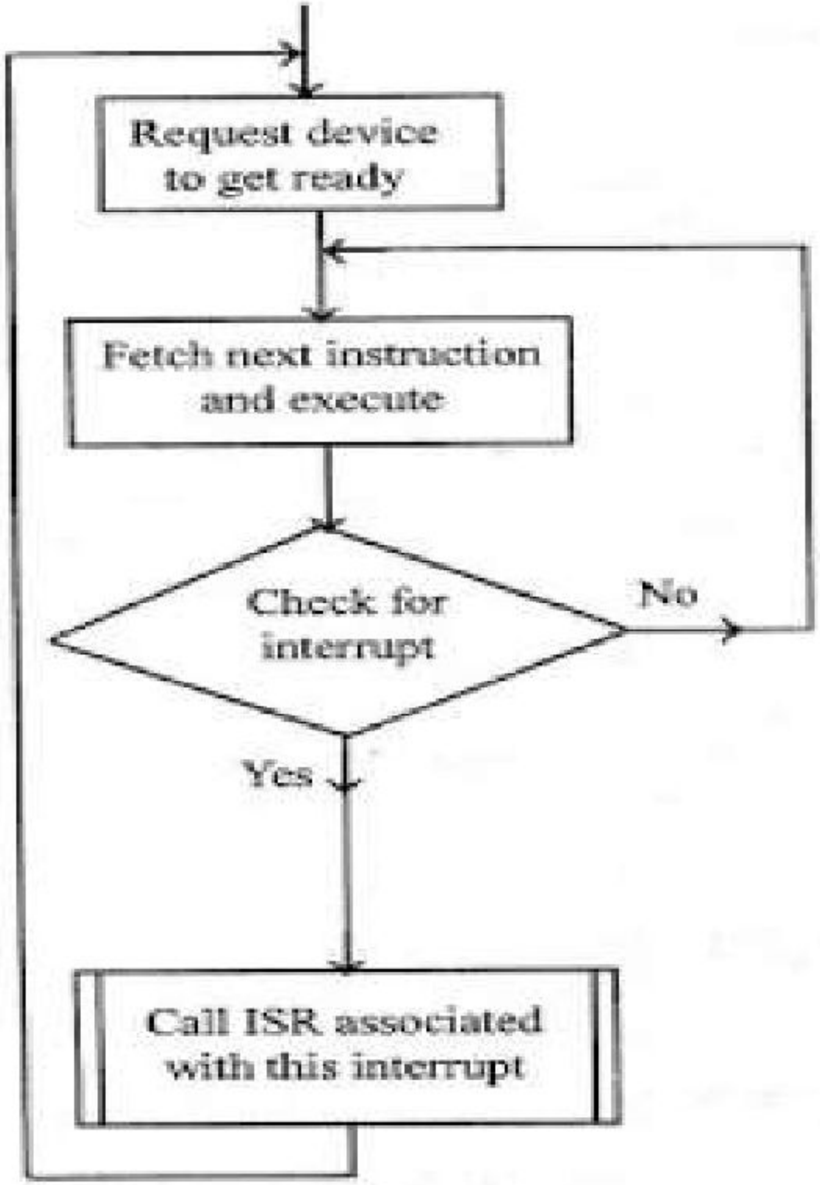
3. If more than one lines are enabled simultaneously then the processor pick up the request which have the highest priority and all others are discarded.

4. After completion of the current instruction execution, processor checks for the respective conditions for the activated interrupt or selected interrupt in case of more than one.

5. If condition are not favorable then request is discarded or stored or if the condition are favorable then the processor generates an external INTA or internal acknowledges signal to insert a RST(restart) instruction or the vector location respectively.

6. Now the processor save the address of the next instruction (program counter value) on to stack and switch to the related RST location or vector location.

7. Service routine written on the location is completed which have RET as its last instruction which returns the program control to the main program by retrieving the return address from the stack.


Request device to get ready
Fetch next instruction and execute
Check for interrupt
No
Yes
Call ISR associated with this interrupt

(a) : Main program execution sequence

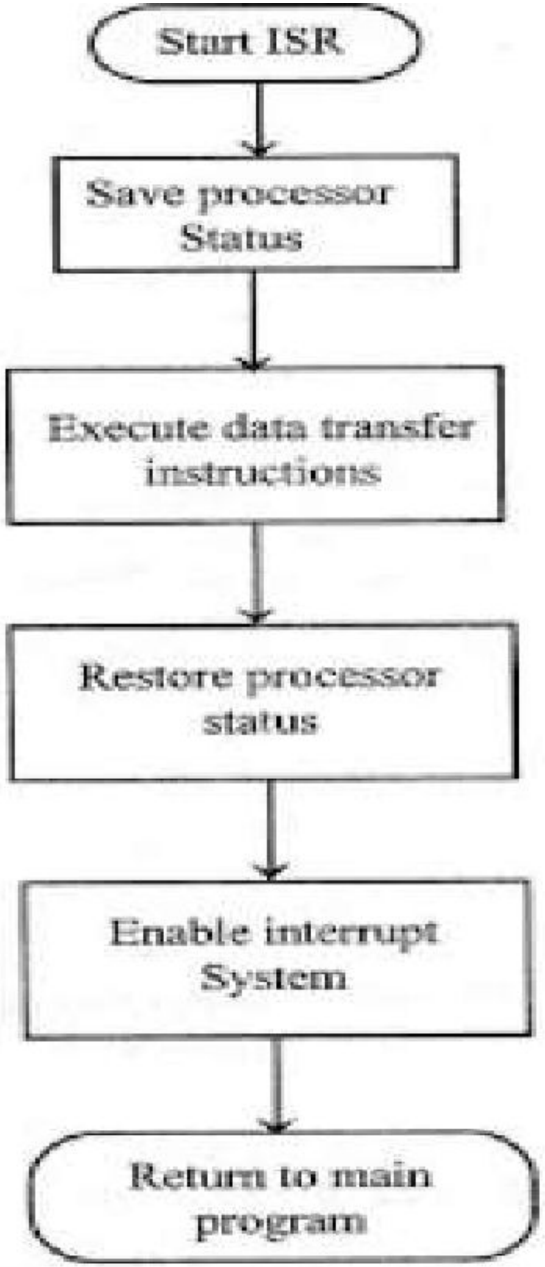
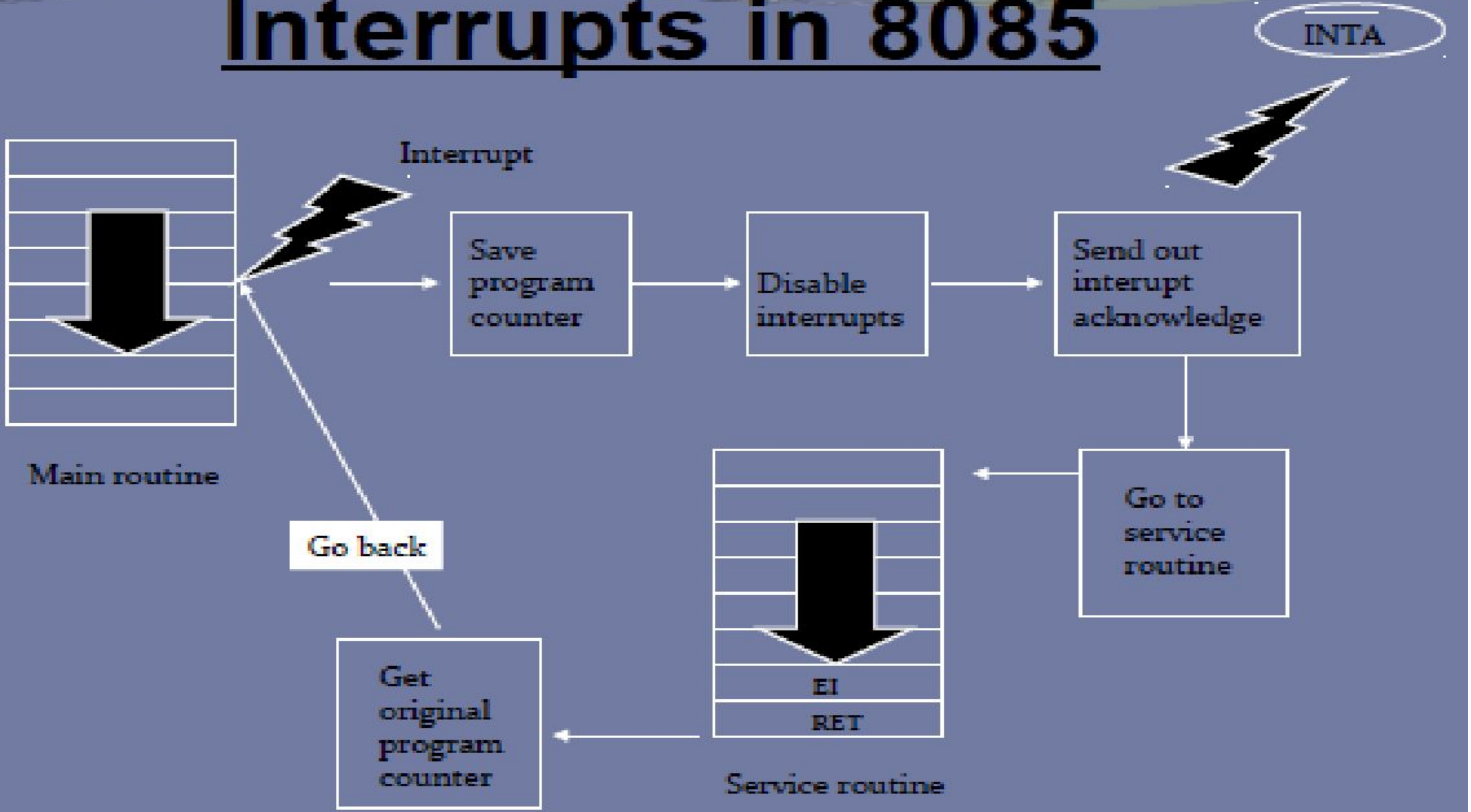


Fig (b) : ISR execution sequence

Interrupts in 8085



Classification of Interrupts

Interrupts can be classified into two types:

- Maskable Interrupts (Can be delayed or Rejected)(a telephone call can be avoided or attended)
- Non-Maskable Interrupts (Can not be delayed or Rejected)(like a smoke detector)

- Interrupts can also be classified into:

- Vectored (the address of the service routine is hard-wired)
- Non-vectored (the address of the service routine needs to be supplied externally by the device)

Types of Interrupt

- Software Interrupt [RST-restart]

- Hardware Interrupt [TRAP, RST7.5,
  RST6.5, RST5.5,INTR]

Software Interrupt

- It is a instruction based Interrupt which is completely controlled by software.
- That means programmer can use this instruction to execute interrupt in main program.
- There are eight software interrupt available in µP that are **RST0 to RST7.**

The vector address for these interrupts can be calculated as

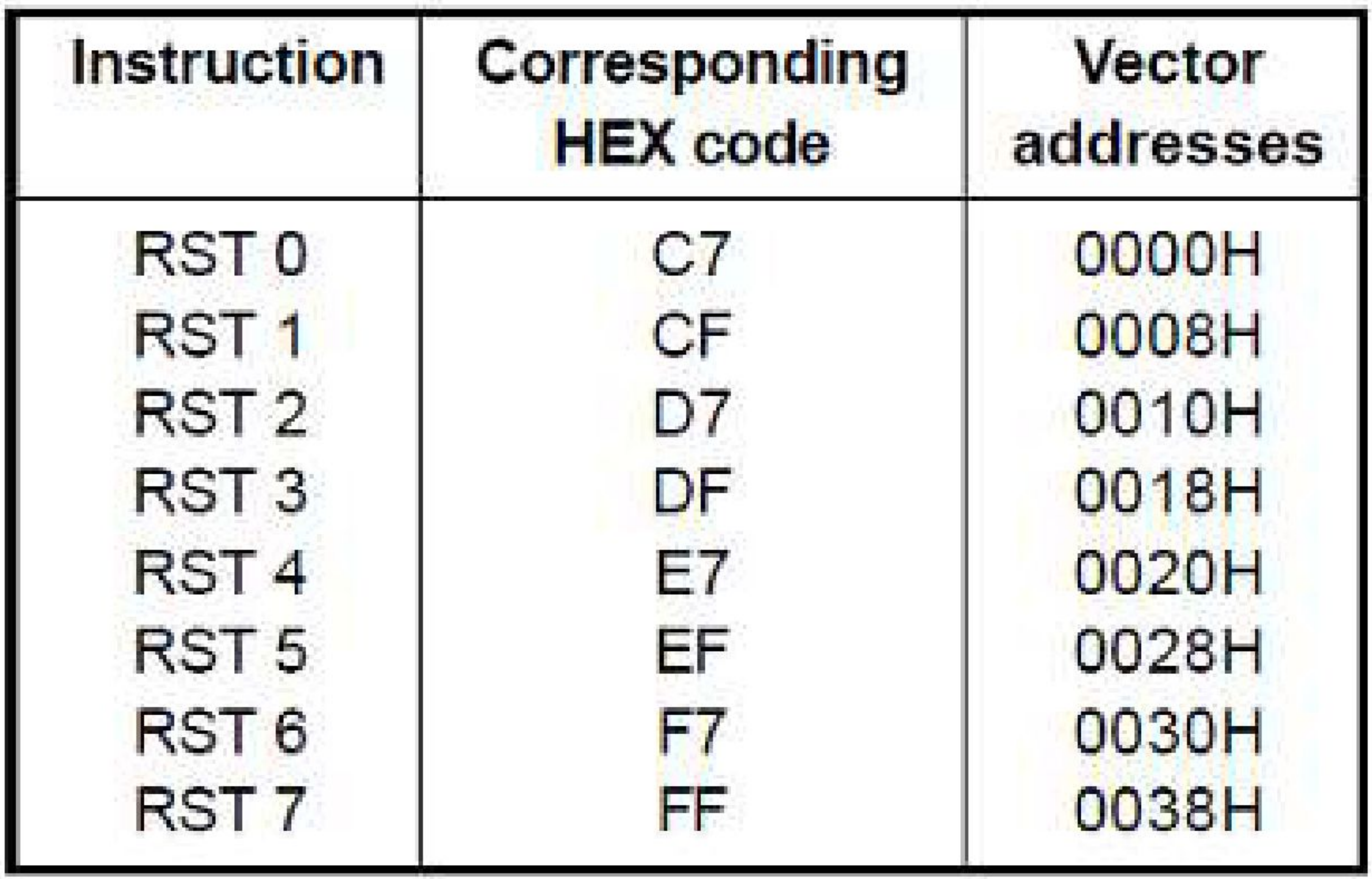
Interrupt number * 8 = vector address

For RST 5 5*8 = 40(in decimal) =28H (in Hexa)

Vector address for interrupt RST5 is 0028H. This vector address is stored in Program Counter(PC).

- These instruction allow transfer of program control from the main program to predefined service routine is also referred to as ISR(Interrupt Service Routine).

Software Interrupt.. cont



Instruction	Corresponding HEX code	Vector addresses
RST 0	C7	0000H
RST 1	CF	0008H
RST 2	D7	0010H
RST 3	DF	0018H
RST 4	E7	0020H
RST 5	EF	0028H
RST 6	F7	0030H
RST 7	FF	0038H

Hardware Interrupt

- ✓ This interrupt is caused by sending a signal on one of the interrupt pins of the microprocessor.
- ✓ An external device initiates the hardware interrupts and placing an appropriate signal at the interrupt pin of the processor.
- ✓ If the interrupt is accepted then the process or executes an interrupt service routine (ISR).
- ✓ Hardware interrupt is Asynchronous(it can occur at any time).
- ✓ The 8085 has five hardware interrupts

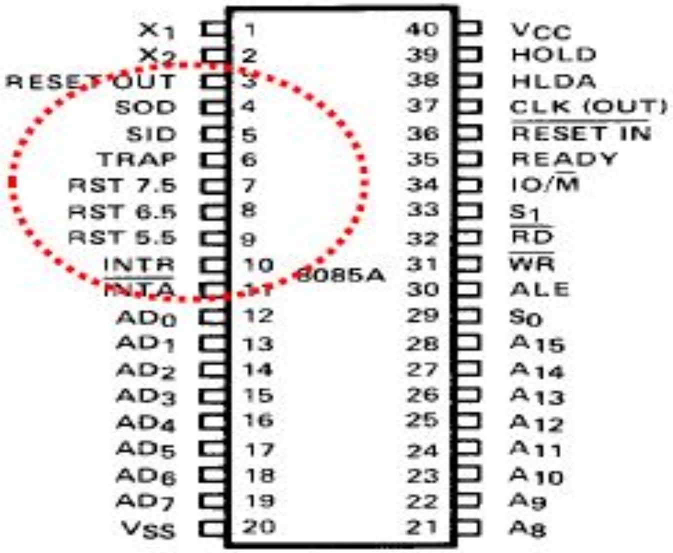
  (1)TRAP

  (2)RST7.5

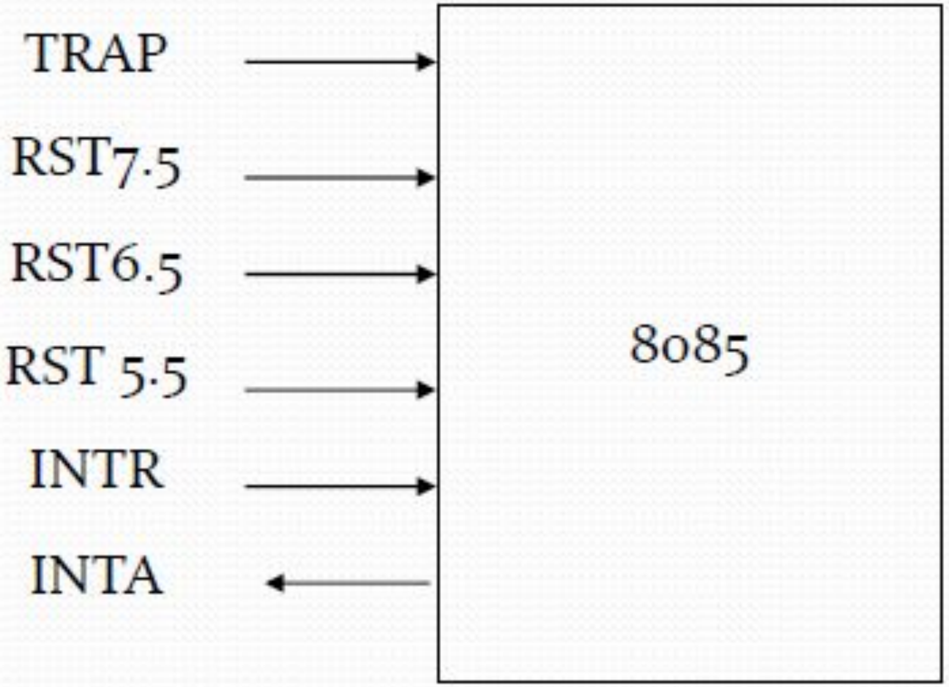
  (3)RST6.5

  (4)RST5.5

  (5)INTR(address is supplied externally)



8085 Interrupts



✓The hardware interrupts are classified Two types–

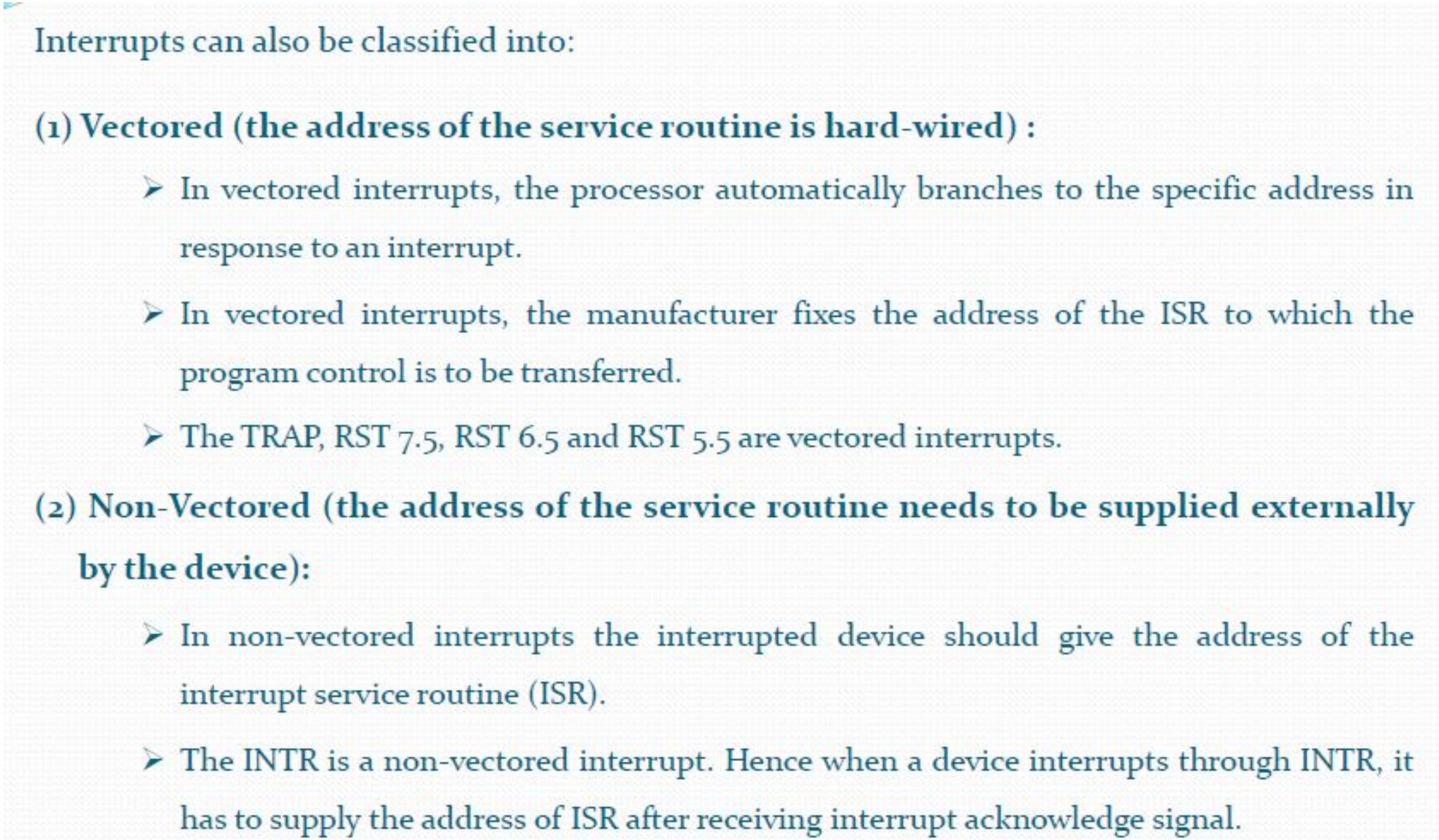
(1)Maskable Interrupts (Can be delayed or Rejected) :

- An interrupt which can be disabled by software that means we can disable the interrupt by sending appropriate instruction, is called a maskable interrupt.
- RST 7.5, RST 6, RST 5.5 , INT R are the example of Maskable Interrupt.

(2)Non-Maskable Interrupts (Can not be delayed or Rejected):

- Cannot disable the interrupt by sending any instruction is called Non Maskable Interrupt.
- TRAP interrupt is the non-maskable interrupt for 8085. It means that if an interrupt comes via TRAP, 8085 will have to recognize the interrupt we cannot mask it.

Hardware interrupts … cont



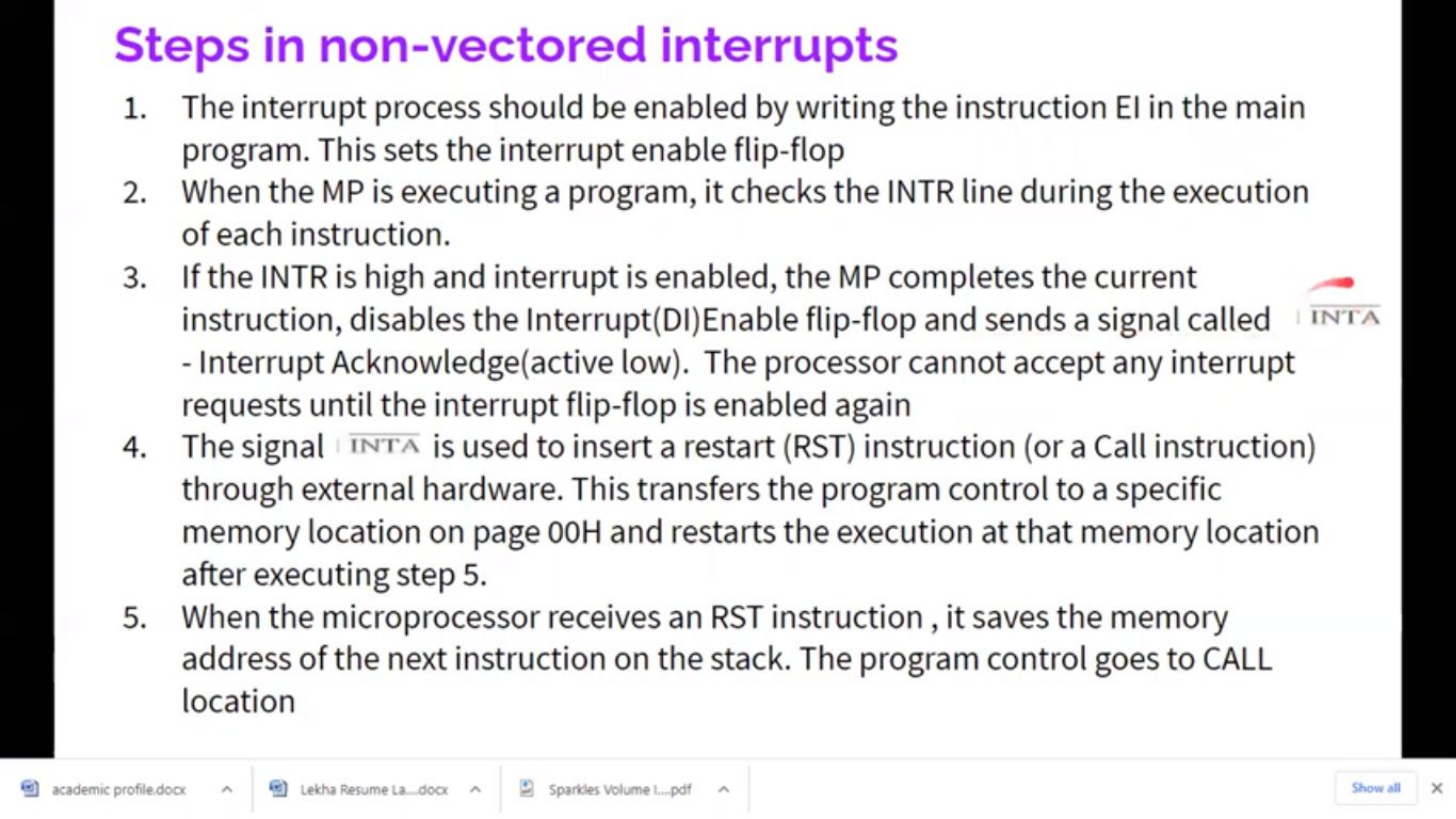
Interrupts can also be classified into:

(1) Vectored (the address of the service routine is hard-wired) :

- In vectored interrupts, the processor automatically branches to the specific address in response to an interrupt.
- In vectored interrupts, the manufacturer fixes the address of the ISR to which the program control is to be transferred.
- The TRAP, RST 7.5, RST 6.5 and RST 5.5 are vectored interrupts.

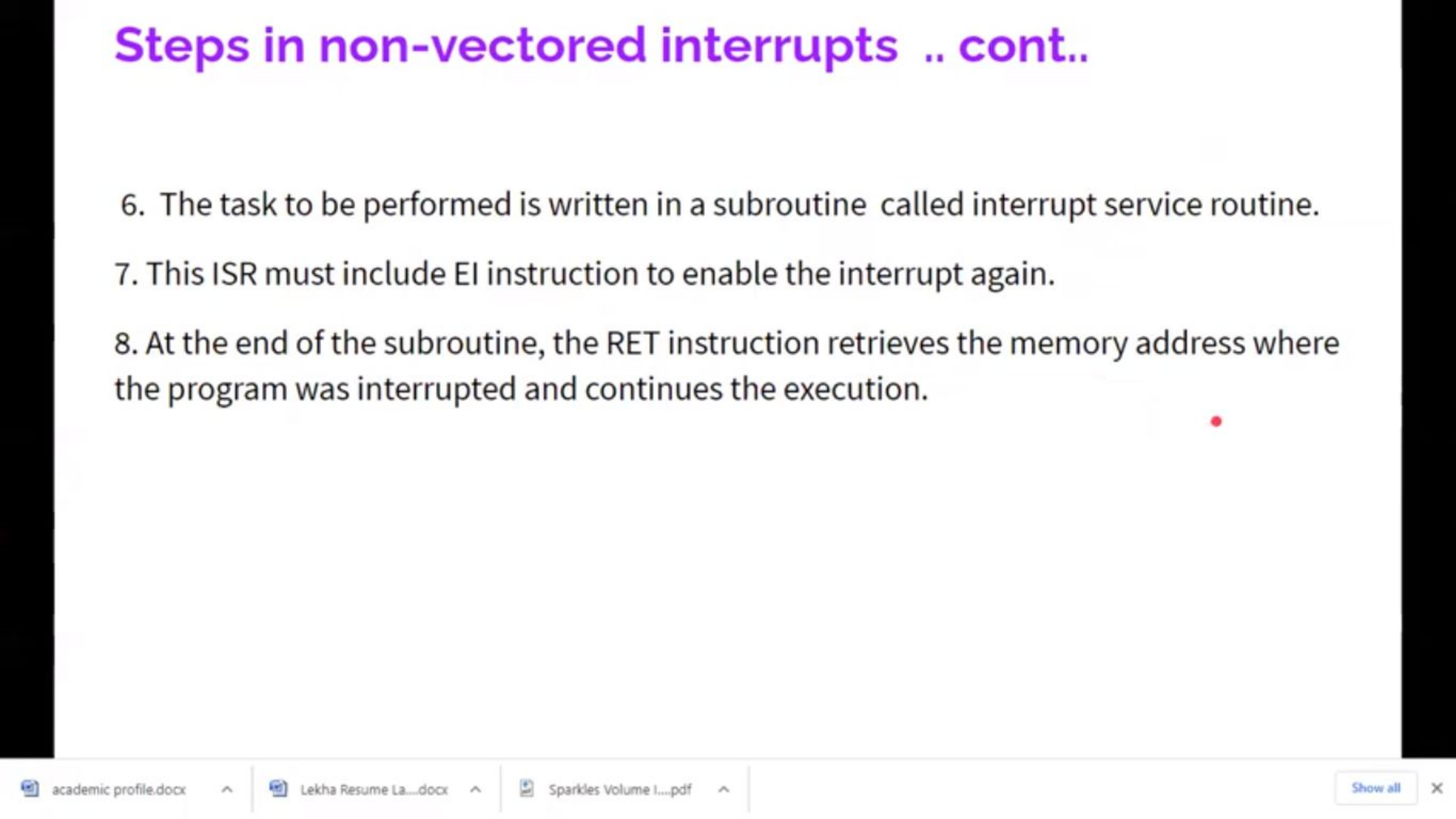
(2) Non-Vectored (the address of the service routine needs to be supplied externally by the device):

- In non-vectored interrupts the interrupted device should give the address of the interrupt service routine (ISR).
- The INTR is a non-vectored interrupt. Hence when a device interrupts through INTR, it has to supply the address of ISR after receiving interrupt acknowledge signal.



Steps in non-vectored interrupts

1. The interrupt process should be enabled by writing the instruction EI in the main program. This sets the interrupt enable flip-flop
2. When the MP is executing a program, it checks the INTR line during the execution of each instruction.
3. If the INTR is high and interrupt is enabled, the MP completes the current instruction, disables the Interrupt(DI)Enable flip-flop and sends a signal called $\overline{INTA}$ - Interrupt Acknowledge(active low). The processor cannot accept any interrupt requests until the interrupt flip-flop is enabled again
4. The signal $\overline{INTA}$ is used to insert a restart (RST) instruction (or a Call instruction) through external hardware. This transfers the 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 microprocessor receives an RST instruction , it saves the memory address of the next instruction on the stack. The program control goes to CALL location



Steps in non-vectored interrupts .. cont..

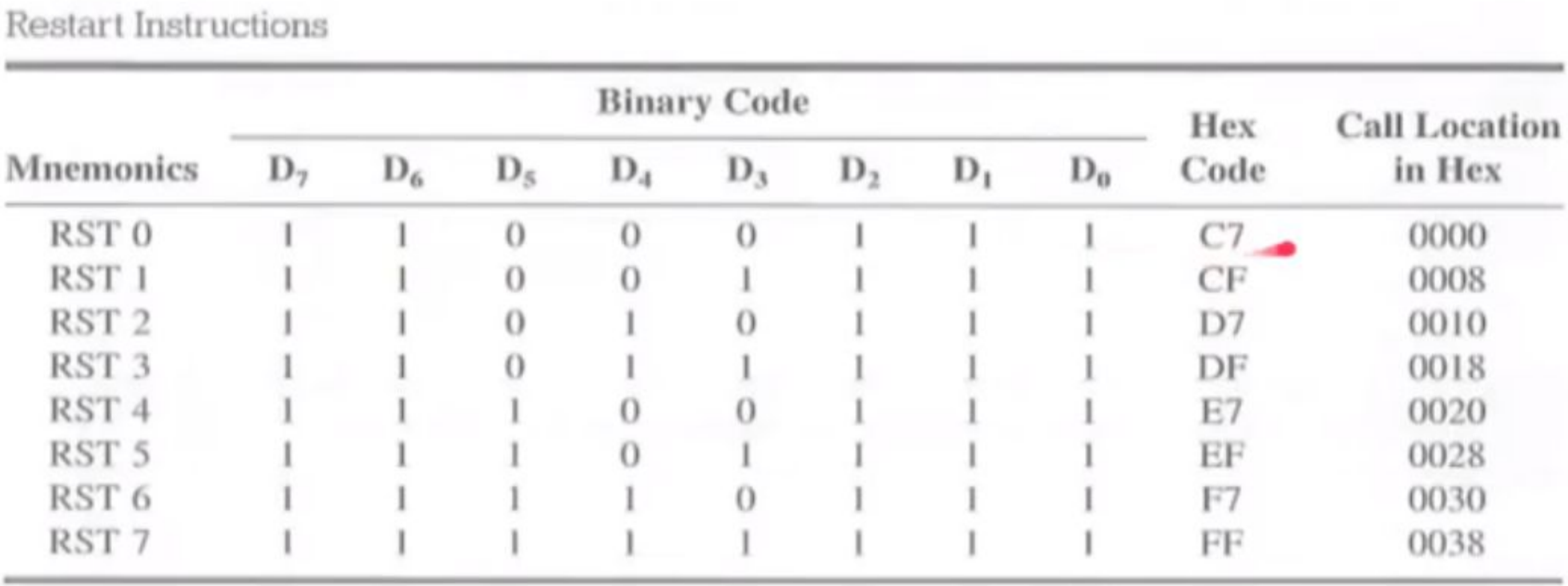
6. The task to be performed is written in a subroutine called interrupt service routine.

7. This ISR must include EI instruction to enable the interrupt again.

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

8085 non-vectored interrupts

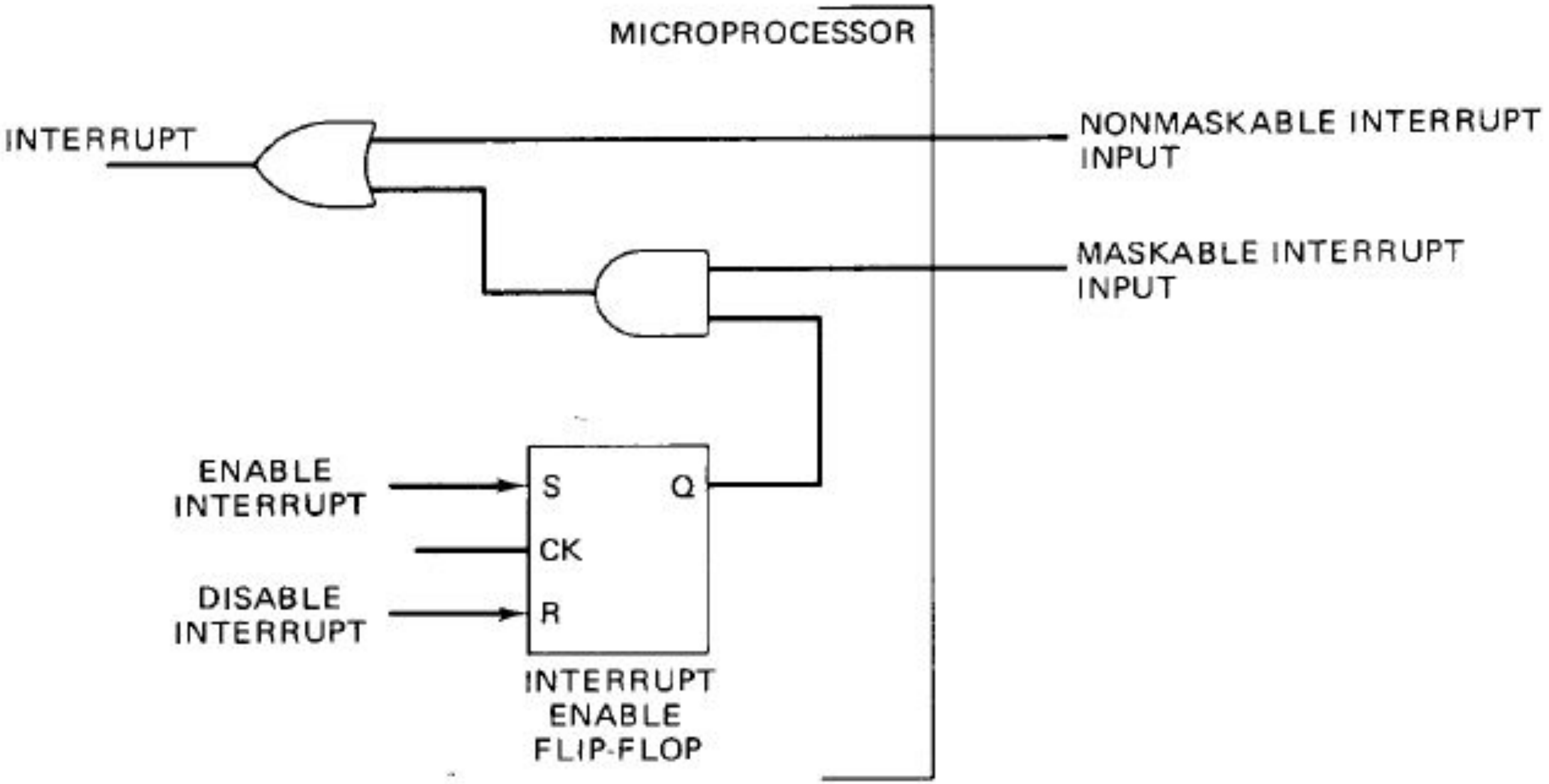
8085 recognizes 8 RESTART instructions: each of these send the execution to a predetermined hard-wired memory location.



Restart Instructions

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

Maskable & Nonmaskable INT

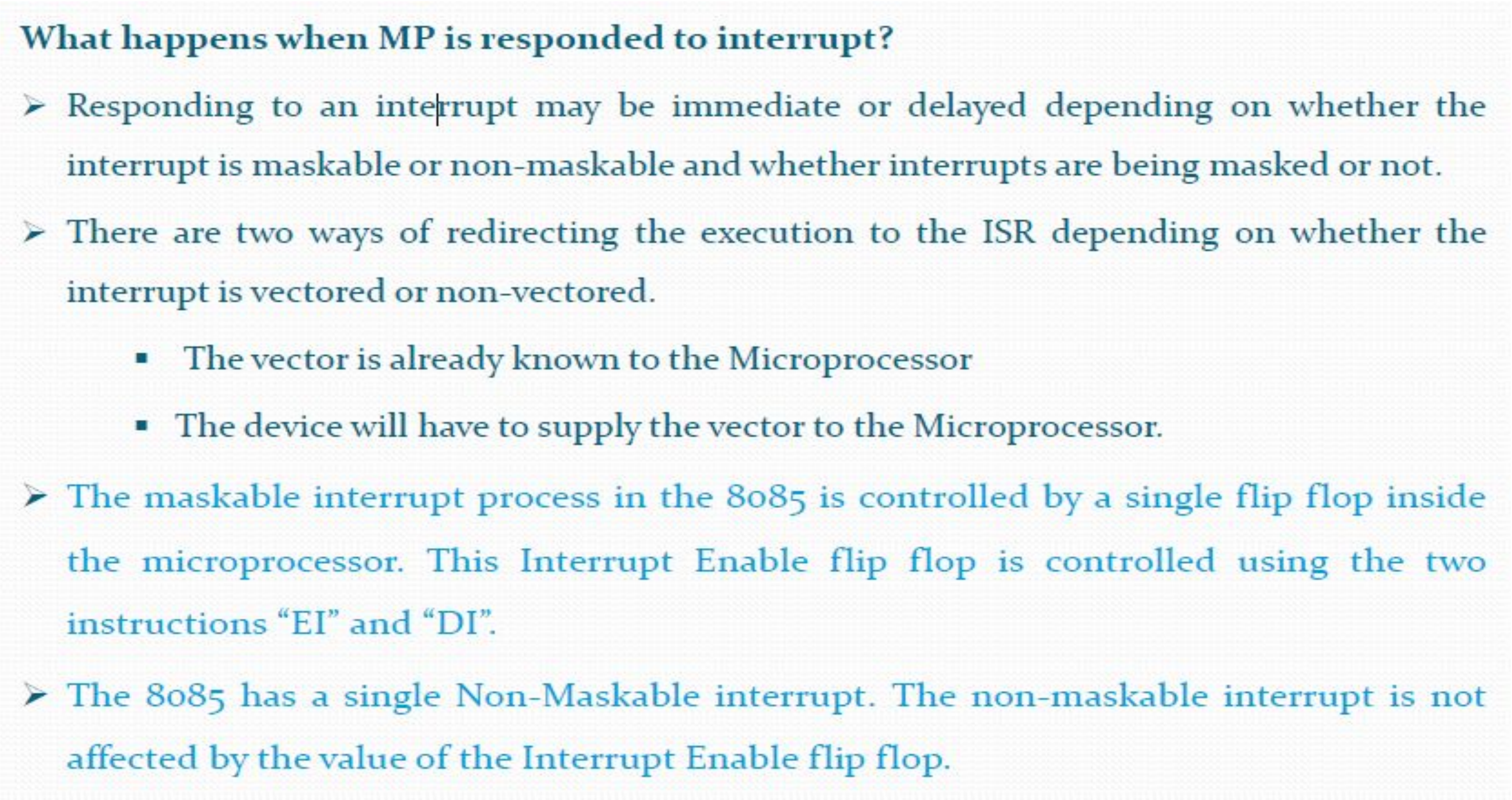


What happens when MP is interrupted?

- When the MP receives an interrupt signal, it suspends the currently executing program and jumps to an Interrupt Service Routine (ISR) to respond to the incoming interrupt.

- Each interrupt will most probably have its own ISR.

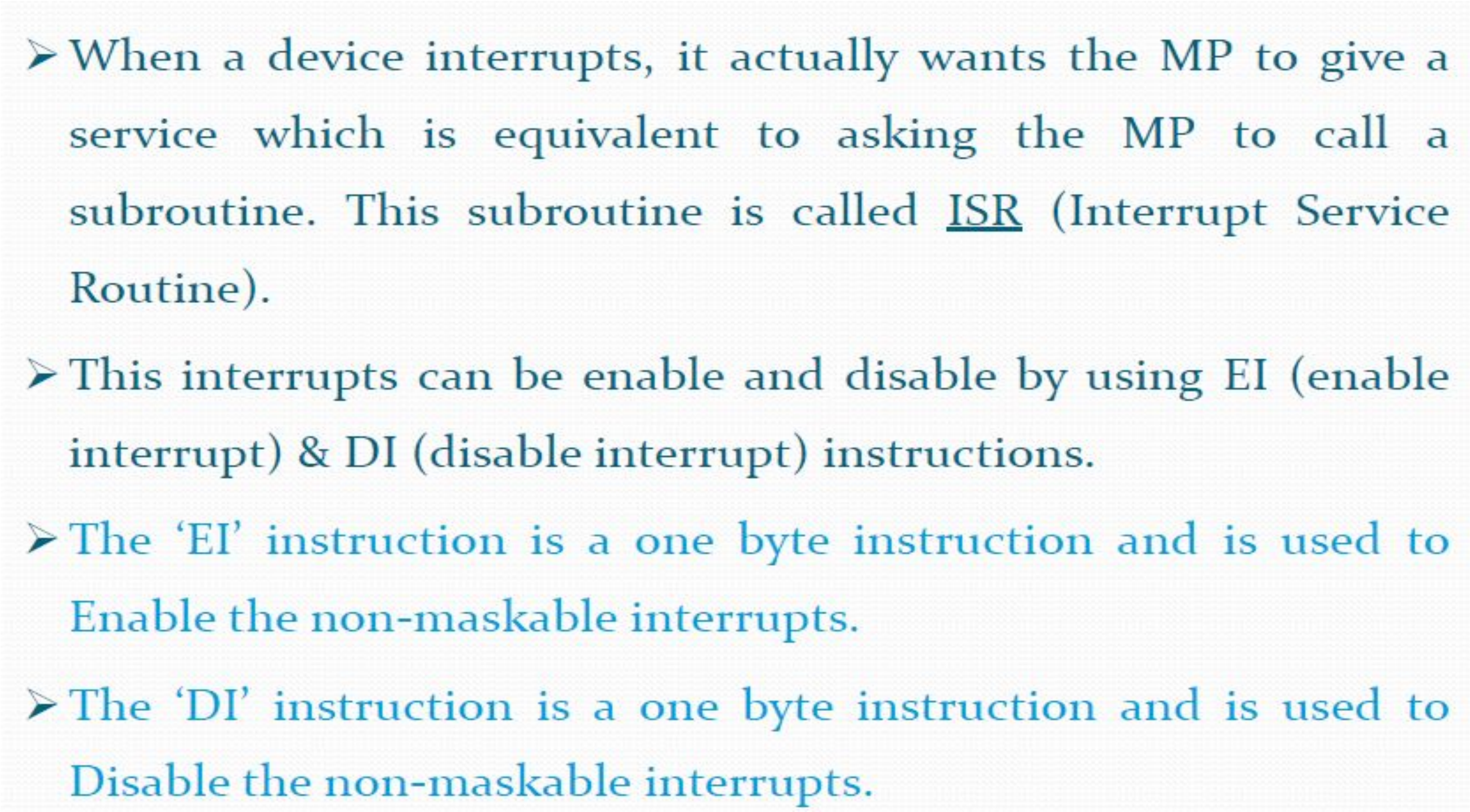
Interrupts in 8085



What happens when MP is responded to interrupt?

- Responding to an interrupt may be immediate or delayed depending on whether the interrupt is maskable or non-maskable and whether interrupts are being masked or not.
- There are two ways of redirecting the execution to the ISR depending on whether the interrupt is vectored or non-vectored.
  - The vector is already known to the Microprocessor
  - The device will have to supply the vector to the Microprocessor.
- The maskable interrupt process in the 8085 is controlled by a single flip flop inside the microprocessor. This Interrupt Enable flip flop is controlled using the two instructions "EI" and "DI".
- The 8085 has a single Non-Maskable interrupt. The non-maskable interrupt is not affected by the value of the Interrupt Enable flip flop.

Interrupts in 8085



- When a device interrupts, it actually wants the MP to give a service which is equivalent to asking the MP to call a subroutine. This subroutine is called ISR (Interrupt Service Routine).
- This interrupts can be enable and disable by using EI (enable interrupt) & DI (disable interrupt) instructions.
- The ‘EI’ instruction is a one byte instruction and is used to Enable the non-maskable interrupts.
- The ‘DI’ instruction is a one byte instruction and is used to Disable the non-maskable interrupts.

Enable Interrupt(EI)

- ✓ The interrupt process is enable by using EI instruction in the main program.
- ✓ It is 1-byte instruction.
- ✓ It enables the interrupt process.
- ✓ Enabling will save the current status and jumps to an interrupt service routine (ISR). After completion it will return back to the main program again.

Disable Interrupt(DI)

✓ This DI instruction is used to disable the interrupt.

✓ It is 1-byte instruction.

✓ This instruction reset the interrupt enable and disables the interrupt.

- ➢ Both EI & DI are used to enable and disable the interrupts. If the interrupt is masked (disabled), they will not be recognized by microprocessor.
- ➢ To enable It again they must be unmasked (enabled) by using EI.

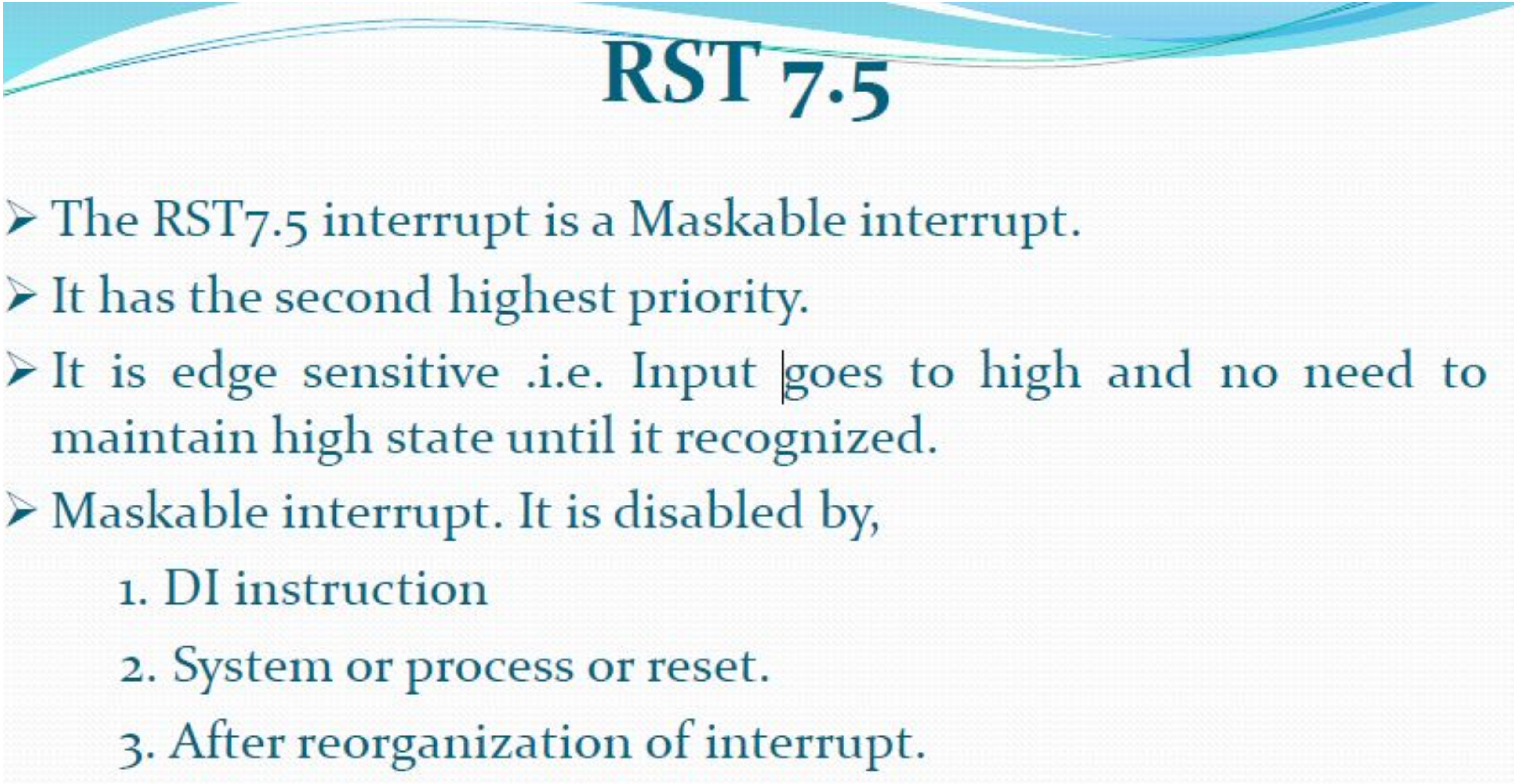
8085 Interrupt

Interrupt name	Maskable	Vectored
INTR	Yes	No
RST 5.5	Yes	Yes
RST 6.5	Yes	Yes
RST 7.5	Yes	Yes
TRAP	No	Yes

TRAP

- This interrupt is a Non-Maskable interrupt. It is unaffected by any mask or interrupt enable.
- TRAP is the highest priority and vectored interrupt (as vector address is fixed i.e. memory location where to transfer control).
- TRAP interrupt is edge and level triggered. This means that the TRAP must go high and remain high until it is acknowledged.
- In sudden power failure, it executes a ISR and send the data from main memory to back up memory.
- The signal, which over rides the TRAP, is HOLD signal. (i.e., If the process or receives HOLD and TRAP at the same time then HOLD is recognized first and then TRAP is recognized).However, TRAP has lower priority than the HLD signal used for DMA.
- There are two ways to clear TRAP interrupt.

  1. By resetting microprocessor (External signal)
  2. By giving a high TRAP ACKNOWLEDGE (Internal signal)



RST 7.5

- The RST7.5 interrupt is a Maskable interrupt.
- It has the second highest priority.
- It is edge sensitive .i.e. Input goes to high and no need to maintain high state until it recognized.
- Maskable interrupt. It is disabled by,

  1. DI instruction
  2. System or process or reset.
  3. After reorganization of interrupt.

RST 6.5 and 5.5

- The RST 6.5 AND 5.5 interrupt is a Maskable interrupt.
- It RST 6.5 has the third and RST 5.5 has forth highest priority.
- It is level triggered. i.e. Input goes to high stay high state until it recognized.
- Enable by EI instruction.
- Maskable interrupt. It is disabled by,

  1. DI, SIM instruction
  2. System or process or reset.
  3. After reorganization of interrupt.

INTR

- The INTR interrupt is a Maskable interrupt. It is disabled by,

  1. DI, SIM instruction
  2. System or process or reset.
  3. After reorganization of interrupt.

- Enable by EI instruction. Has lowest Priority.
- Non-Vectored interrupt After receiving INTA(active low) Signal, It has to supply the address of ISR. It is a level sensitive interrupts .i.e. Input goes to high and it is necessary to maintain high state until it recognized.
- The following sequence of events occurs when INTR signal goes high.

  1. The 8085 checks the status of INTR signal during execution of each instruction.
  2. If INTR signal is high , then 8085 complete its current instruction and sends active low interrupt acknowledge signal, if the interrupt is enabled
  3. In response to the acknowledge signal, external logic places an instruction OPCODE on the data bus. In the case of multi byte instruction, additional interrupt acknowledge machine cycles are generated by the 8085 to transfer the additional bytes in to the microprocessor.
  4. On receiving the instruction, the 8085 save the address of next instruction on stack and execute received instruction.

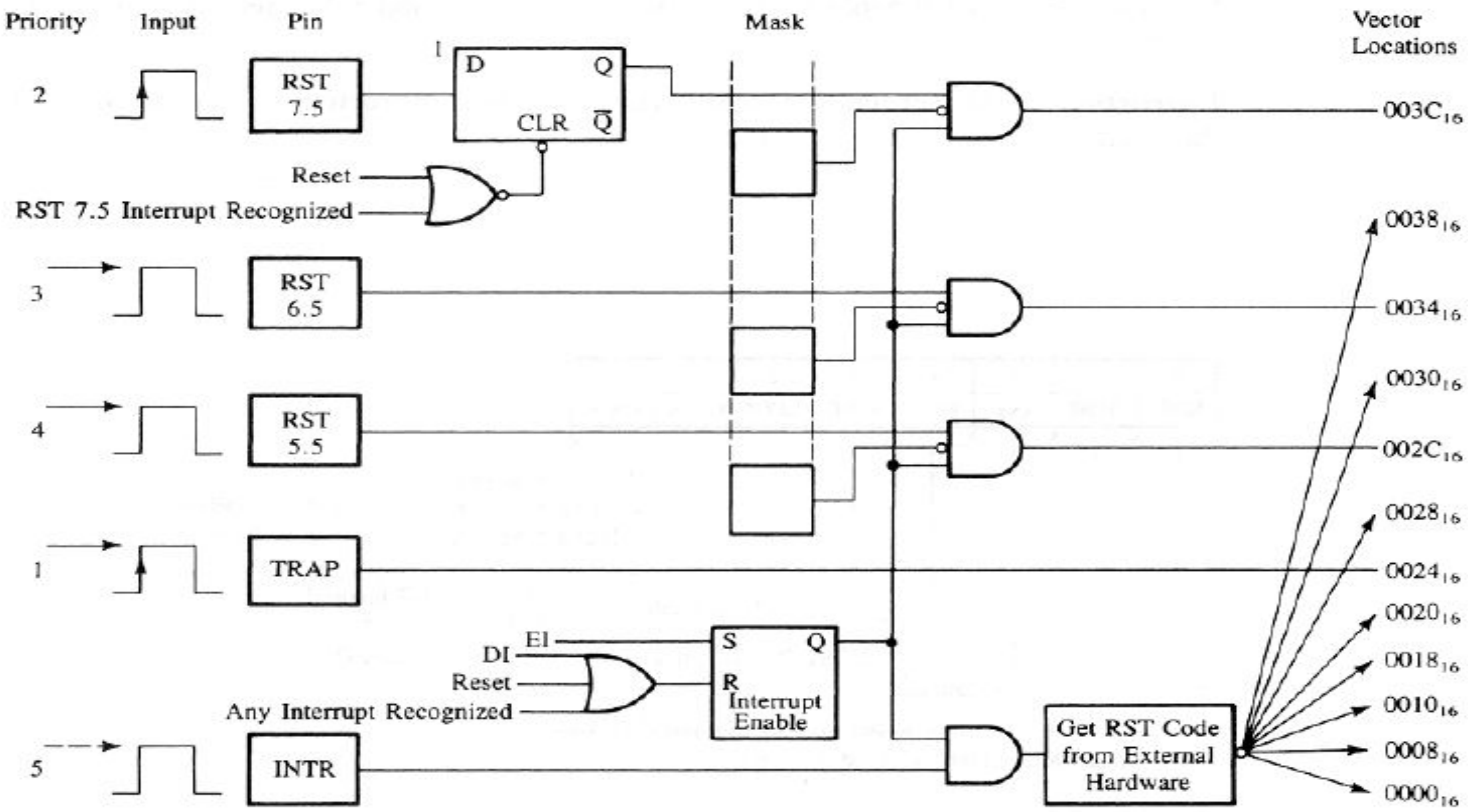
Interrupt Vectors & the Vector Table

- An **interrupt vector** is a pointer to where the ISR is stored in memory.
- All interrupts (vectored or otherwise) are mapped onto a memory area called the **Interrupt Vector Table (IVT).**
  - The IVT is usually located in (0000H - 00FFH).

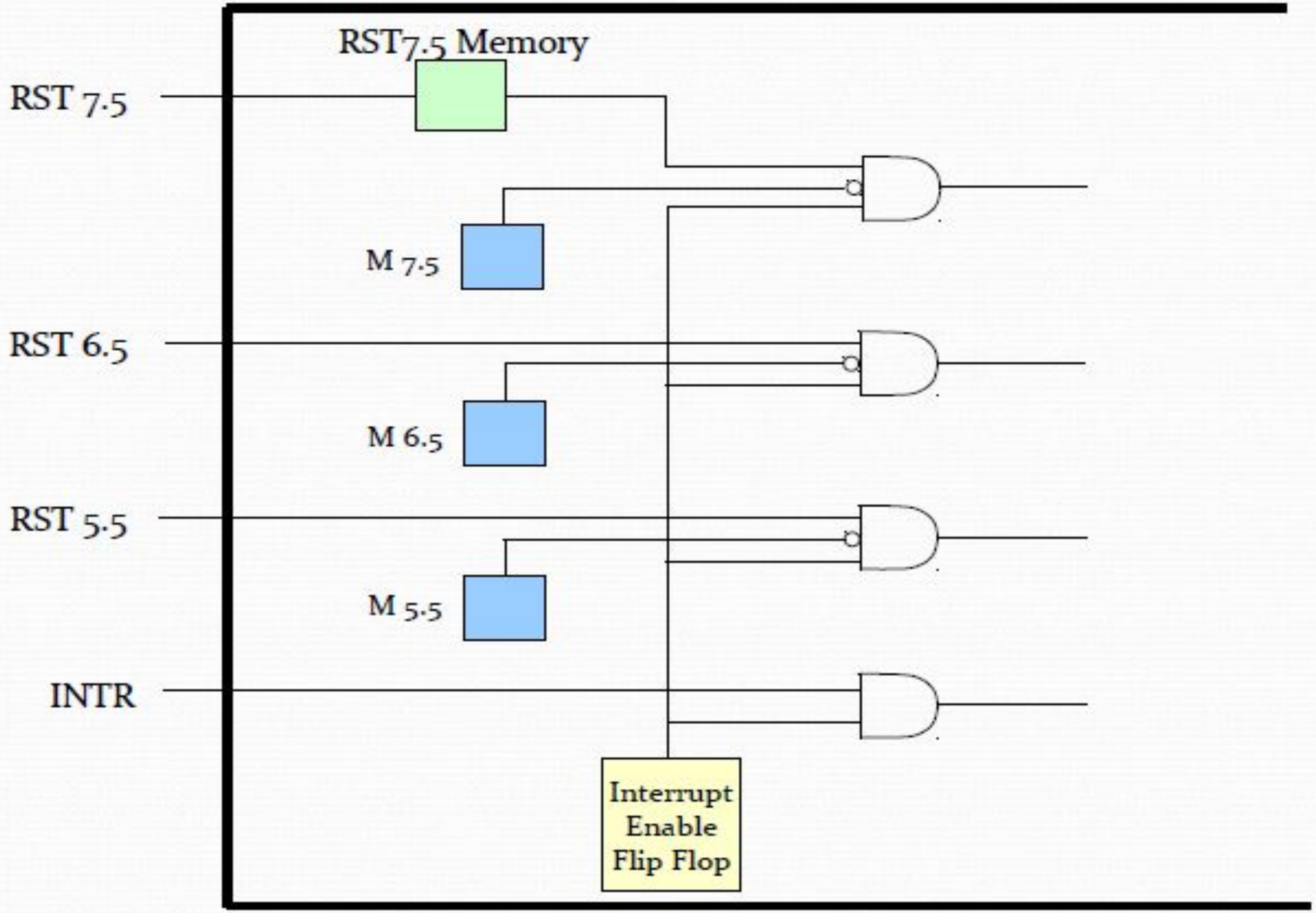
Vector Address = Interrupt number * 8

Interrupt Name	Calculation	Vector Address
INTR	--	--
TRAP (RST 4.5)	4.5x8=36(Decimal))	0024H(Hexa)
RST 5.5	5.5x8=44	002CH
RST 6.5	6.5x8=52	0034H
RST 7.5	7.5x8=60	003CH

Vectored Interrupts



MASKABLE INTERRUPTS



The 8085 Interrupts

Interrupt Name	Maskable	Masking Method	Vectored	Priority	ISR address	Triggering Method
TRAP	No	None	Yes	1st (Highest)	0024H	Level & Edge Sensitive
RST 7.5	Yes	DI / EI SIM	Yes	2nd	003CH	Positive Edge Sensitive
RST 6.5	Yes	DI / EI SIM	Yes	3rd	0034H	Level Sensitive
RST 5.5	Yes	DI / EI SIM	Yes	4th	002CH	Level Sensitive
INTR	Yes	DI / EI	No	5th (lowest)	No specific location	Level Sensitive

Level Triggered:- The signal at these pins must be maintained until the interrupt is acknowledged. External interrupt request flip-flops are required.

Edge Triggered: - Only a pulse is required to set the interrupt request → this request is remembered until the 8085A responds to the interrupt or until the request is reset by the **SIM** instruction or a /RESET IN signal. The interrupt request flip-flops for RST7.5 is internal to the microprocessor.

The 8085 Non-Vectored Interrupt Process

1. The interrupt process should be enabled using the EI instruction.
2. The 8085 checks for an interrupt during the execution of every instruction.
3. If INTR is high, MP completes current instruction, disables the interrupt and sends INTA (Interrupt acknowledge) signal to the device that interrupted
4. INTA allows the I/O device to send a RST instruction through data bus.
5. Upon receiving the INTA signal, MP saves the memory location of the next instruction on the stack and the program is transferred to 'call' location (ISR Call) specified by the RST instruction.
6. Microprocessor Performs the ISR.
7. ISR must include the 'EI' instruction to enable the further interrupt within the program.
8. RET instruction at the end of the ISR allows the MP to retrieve the return address from the stack and the program is transferred back to where the program was interrupted.

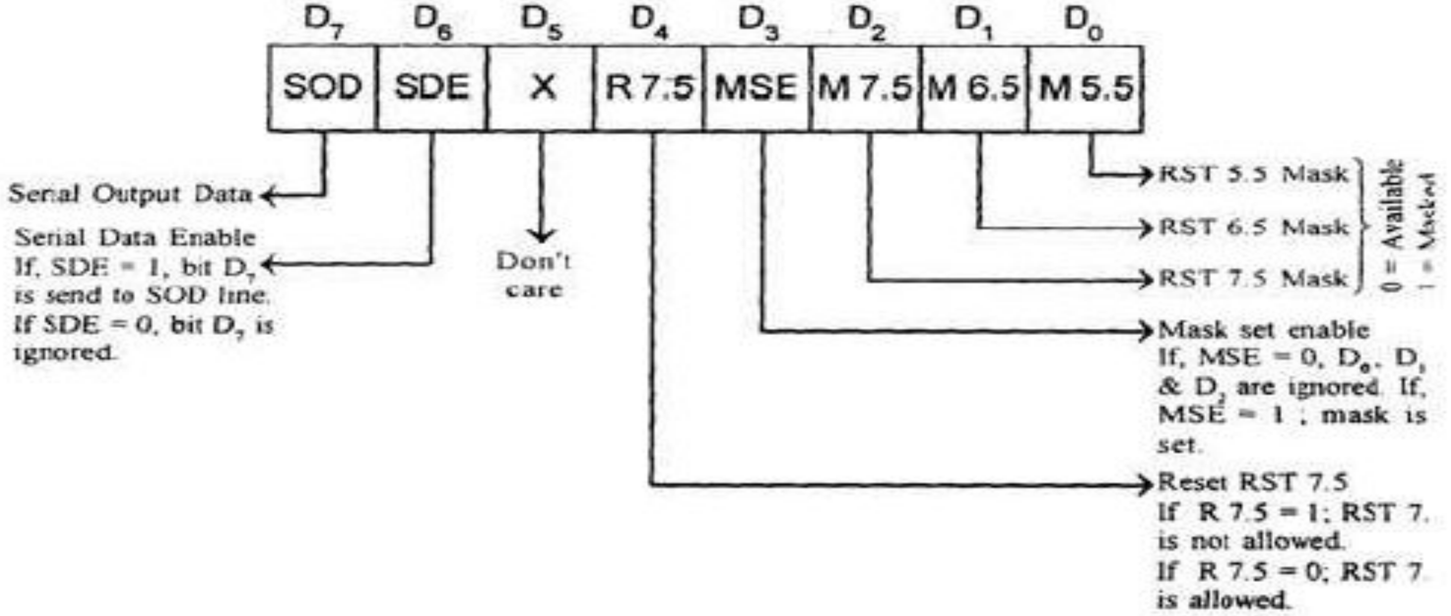
The 8085 Maskable/Vectored Interrupt Process

1. The interrupt process should be enabled using the EI instruction.
2. The 8085 checks for an interrupt during the execution of every instruction.
3. If there is an interrupt, and if the interrupt is enabled using the interrupt mask, the microprocessor will complete the executing instruction, and reset the interrupt flip flop.
4. The microprocessor then executes a call instruction that sends the execution to the appropriate location in the interrupt vector table.
5. When the microprocessor executes the call instruction, it saves the address of the next instruction on the stack.
6. The microprocessor jumps to the specific service routine.
7. The service routine must include the instruction EI to re-enable the interrupt process.
8. At the end of the service routine, the RET instruction returns the execution to where the program was interrupted.

SIM for interrupt

- ✓ 8085 provide the additional masking facility for RST 7.5, RST 6.5 and RST 5.5 using SIM instruction.
- ✓ The status of these interrupts can be read by executing RIM instruction.
- ✓ The masking or unmasking of RST 7.5, RST 6.5 and RST 5.5 interrupts can be performed by moving an 8-bit data to accumulator and then executing SIM instruction.
- ✓ The format of the 8 bit data is shown below

SIM Instruction



D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SOD	SDE	X	R 7.5	MSE	M 7.5	M 6.5	M 5.5

Serial Output Data

Serial Data Enable
If, SDE = 1, bit D_7 is send to SOD line.
If SDE = 0, bit D_7 is ignored.

Don't care

RST 5.5 Mask
RST 6.5 Mask
RST 7.5 Mask
0 = Available
1 = Masked

Mask set enable
If, MSE = 0, D_0, D_1 & D_2 are ignored. If, MSE = 1 ; mask is set.

Reset RST 7.5
If R 7.5 = 1; RST 7. is not allowed.
If R 7.5 = 0; RST 7 is allowed.

SIM Instruction

- Bit 0 is the mask for RST 5.5, bit 1 is the mask for RST 6.5 and bit 2 is the mask for RST 7.5.
  - If the mask bit is 0, the interrupt is available.
  - If the mask bit is 1, the interrupt is masked.
- Bit 3 (Mask Set Enable - MSE) is an enable for setting the mask.
  - If it is set to 0 the mask is ignored and the old settings remain.
  - If it is set to 1, the new setting are applied.

SIM Instruction

- Bit 4 of the accumulator in the SIM instruction allows explicitly resetting the RST 7.5 memory even if the microprocessor did not respond to it.
- Bit 5 is not used by the SIM instruction
- Bit 6 & Bit 7 is used for extra functionality such as serial data transmission.

SIM Instruction

- Example: Set the interrupt masks so that RST5.5 is enabled, RST6.5 is masked, and RST7.5 is enabled.
- First, determine the contents of the accumulator

- Enable 5.5	bit 0 = 0
- Disable 6.5	bit 1 = 1
- Enable 7.5	bit 2 = 0
- Allow setting the masks	bit 3 = 1
- Don’t reset the flip flop	bit 4 = 0
- Bit 5 is not used	bit 5 = 0
- Don’t use serial data	bit 6 = 0
- Serial data is ignored	bit 7 = 0

0	0	0	0	1	0	1	0

Contents of accumulator are: 0A H

```
EI              ; Enable interrupts including INTR
MVI A, 0A       ; Prepare the mask to enable RST 7.5, and 5.5, disable 6.5
SIM             ; Apply the settings RST masks
```

Enable all the interrupts in an 8085 system.

Instructions

```
EI              ;Enable interrupts
MVI A,08H       ;Load bit pattern to enable RST 7.5, 6.5, and 5.5
SIM             ;Enable RST 7.5, 6.5, and 5.5
```

Bit $D_3 = 1$ in the accumulator makes the instruction SIM functional, and bits D_2, D_1, and $D_0 = 0$ enable the interrupts 7.5, 6.5, and 5.5.

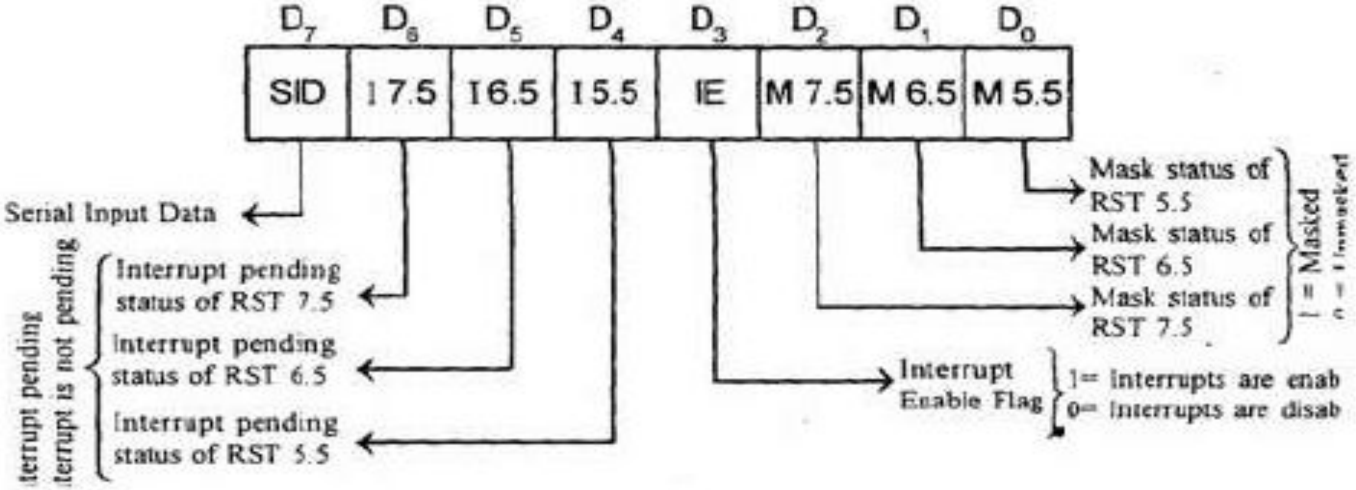
RIM for interrupt

- ✓ The status of pending interrupts can be read from accumulator after executing RIM instruction.
- ✓ Actually RIM does the following three tasks:

  1 Read the interrupt mask (bit 2, 1, 0). 2 Identify pending interrupts (bit 6, 5, 4).

  3 Receive serial input data bit (bit 7).
- ✓ When RIM instruction is executed an 8-bit data is loaded in accumulator, which can be interpreted as shown in fig.

RIM Instruction



The RIM Instruction and the Masks

- Bits 0-2 show the current setting of the mask for each of RST 7.5, RST 6.5 and RST 5.5
  - They return the contents of the three mask flip flops.
  - They can be used by a program to read the mask settings in order to modify only the right mask.

- Bit 3 shows whether the maskable interrupt process is enabled or not.
  - It returns the contents of the Interrupt Enable Flip Flop.
  - It can be used by a program to determine whether or not interrupts are enabled.

The RIM Instruction and the Masks

- Bits 4-6 show whether or not there are pending interrupts on RST 7.5, RST 6.5, and RST 5.5
  - Bits 4 and 5 return the current value of the RST5.5 and RST6.5 pins.
  - Bit 6 returns the current value of the RST7.5 memory flip flop.

- Bit 7 is used for Serial Data Input.
  - The RIM instruction reads the value of the SID pin on the microprocessor and returns it in this bit.

PENDING INTERRUPTS

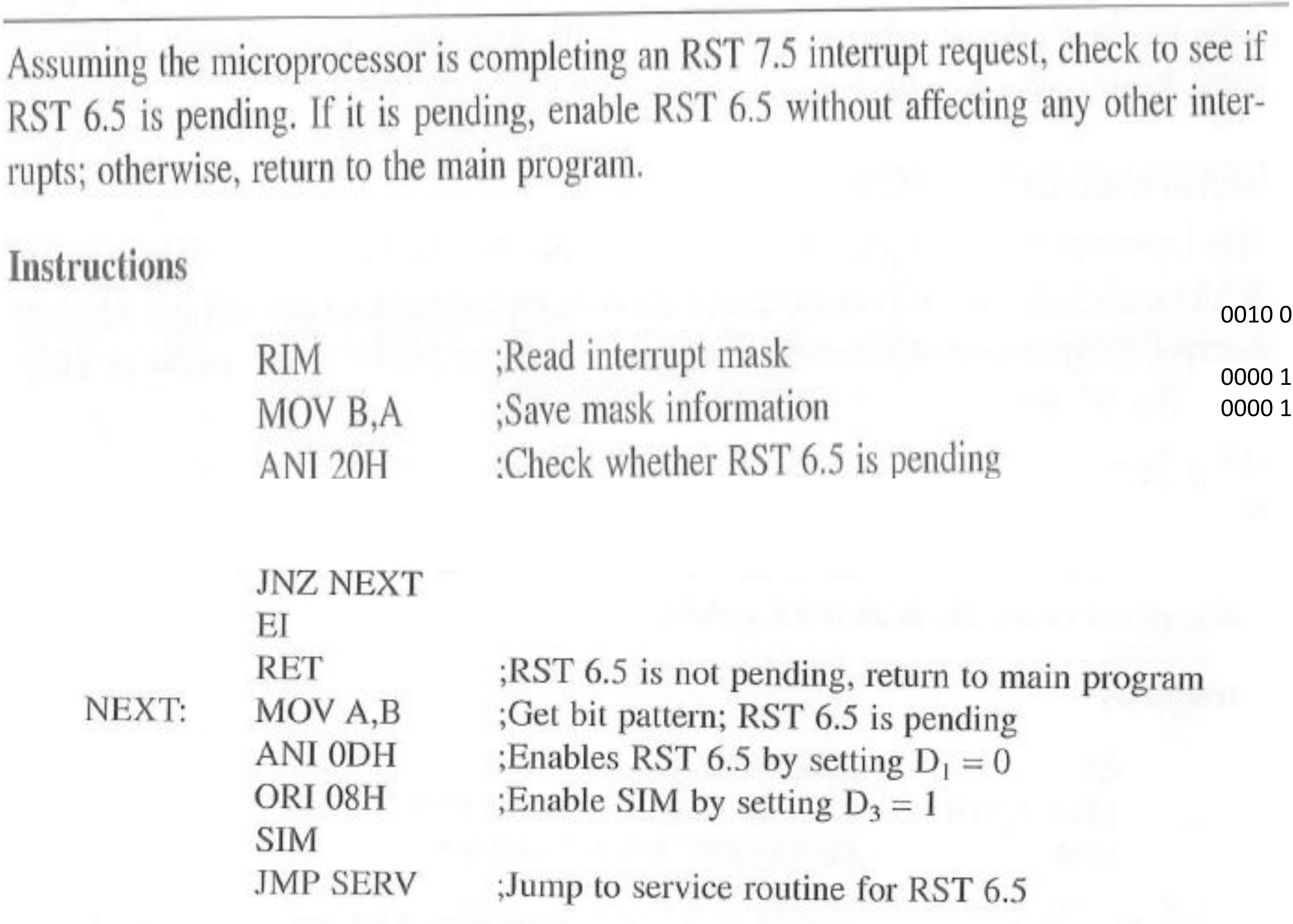
Because there are several interrupt lines, when one interrupt request is being served, other interrupt requests may occur and remain pending. The 8085 has an additional instruction called RIM (Read Interrupt Mask) to sense these pending interrupts.

Instruction RIM: Read Interrupt Mask. This is a 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 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 (Figure 12.7) is used to receive serial data.



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.

Instructions

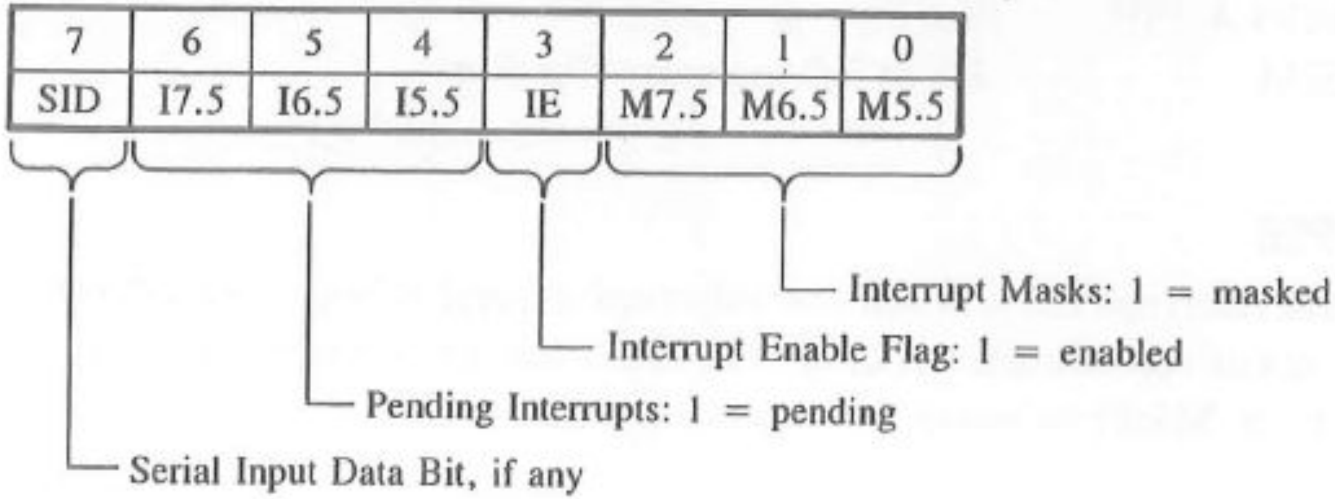
```
        RIM           ;Read interrupt mask
        MOV B,A       ;Save mask information
        ANI 20H       ;Check whether RST 6.5 is pending


        JNZ NEXT
        EI
        RET           ;RST 6.5 is not pending, return to main program
NEXT:   MOV A,B       ;Get bit pattern; RST 6.5 is pending
        ANI 0DH       ;Enables RST 6.5 by setting D1 = 0
        ORI 08H       ;Enable SIM by setting D3 = 1
        SIM
        JMP SERV      ;Jump to service routine for RST 6.5
```

0010 0000 (20H)

0000 1101 (0D)
0000 1000 (08)

The instruction RIM checks for a pending interrupt. Instruction ANI 20H masks all the bits except D_5 to check pending RST 6.5. If $D_5 = 0$, the program control is transferred to the main program. $D_5 = 1$ indicates that RST 6.5 is pending. Instruction ANI 0DH sets $D_1 = 0$ (RST 6.5 bit for SIM), instruction ORI sets $D_3 = 1$ (this is necessary for SIM to be effective), and instruction SIM enables RST 6.5 without affecting any other interrupts. The JMP instruction transfers the program to the service routine (SERV) written for RST 6.5.

The RIM instruction loads the accumulator with the following information:



7	6	5	4	3	2	1	0
SID	I7.5	I6.5	I5.5	IE	M7.5	M6.5	M5.5

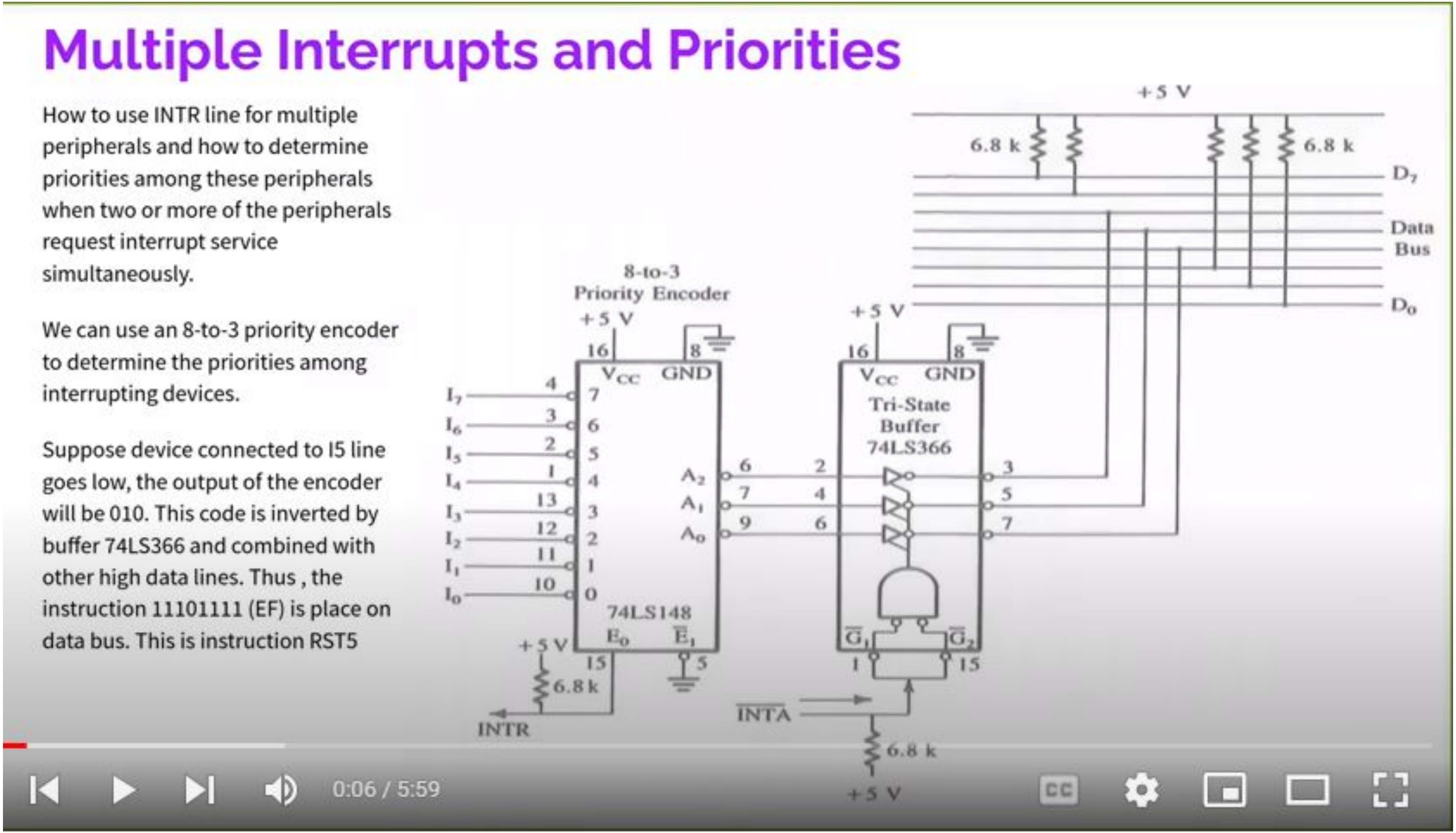
Interrupt Masks: 1 = masked
Interrupt Enable Flag: 1 = enabled
Pending Interrupts: 1 = pending
Serial Input Data Bit, if any

FIGURE 12.7

Interpretation of the Accumulator Bit Pattern for the RIM Instruction

Comparison of Hardware interrupts and Software interrupts

Software Interrupt	Hardware Interrupt
It is a synchronous event.	It is an asynchronous event.
This interrupt is requested by executing instruction.	This interrupt is requested by an external device on a pin.
PC in incremented.	PC is not incremented.
The microprocessor does not execute any interrupt acknowledge cycle to acknowledge this interrupt. The microprocessor executes a normal instruction cycle.	The microprocessor executes either interrupt acknowledge cycle bus or idle machine cycle to acknowledge this interrupt.
It cannot be ignored or masked.	It can be masked except for TRAP.
It has the highest priority among all interrupts.	The priority is lower than that of a software interrupt.
It does not affect interrupt control logic.	It affects interrupt control logic.
It is not used to interface peripherals that means it does not improve the throughput of the system. It is used in program debugging.	It is used to interface peripherals in interrupt-driven I/O. It improves the throughput of the system.



Multiple Interrupts and Priorities

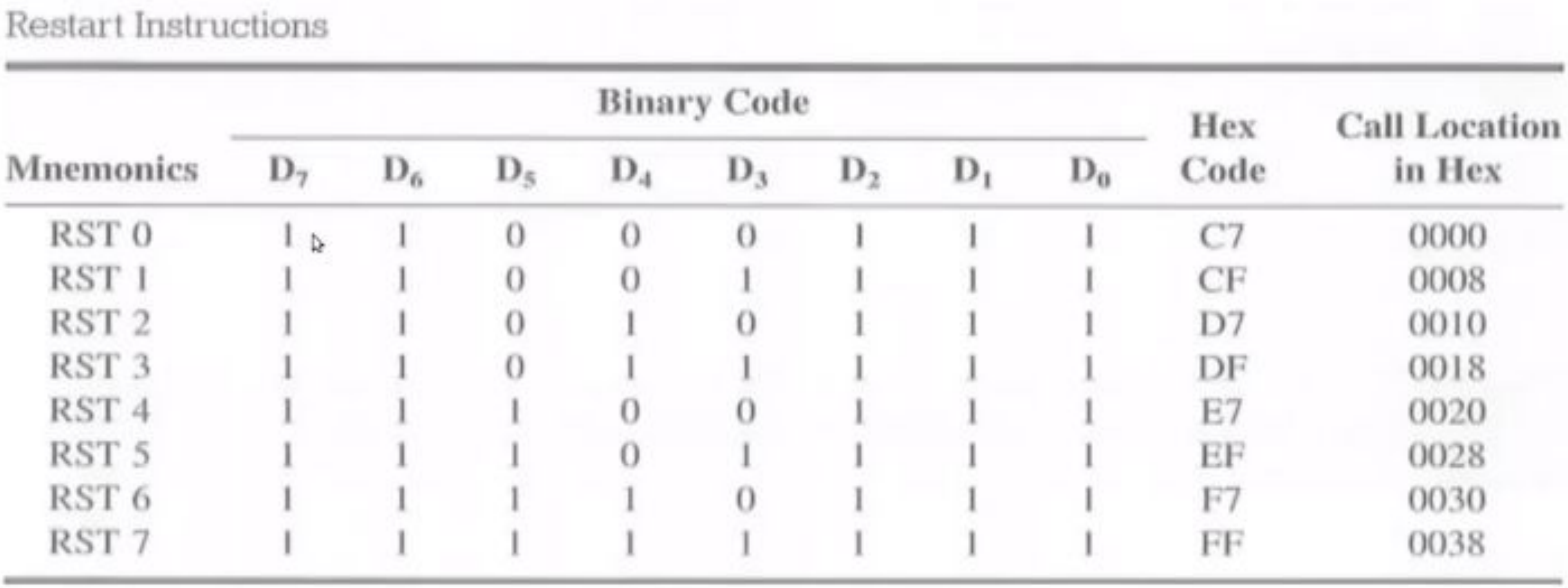
How to use INTR line for multiple peripherals and how to determine priorities among these peripherals when two or more of the peripherals request interrupt service simultaneously.

We can use an 8-to-3 priority encoder to determine the priorities among interrupting devices.

Suppose device connected to I5 line goes low, the output of the encoder will be 010. This code is inverted by buffer 74LS366 and combined with other high data lines. Thus , the instruction 11101111 (EF) is place on data bus. This is instruction RST5

8085 non-vectored interrupts

8085 recognizes 8 RESTART instructions: each of these send the execution to a predetermined hard-wired memory location.



Restart Instructions

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