



Interrupts in 8085

Course Instructor: Dr. Khuraijam Nelson Singh

Interrupt

- An interrupt is a signal that temporarily suspends the normal execution of a program and redirects the control to a specific interrupt service routine (ISR).
- Interrupts allow the microprocessor to respond to external events, such as user input, system events, or hardware signals, without the need for constant polling.
- The μP checks for interrupts during every instruction.

Software Interrupts vs Hardware Interrupts

SOFTWARE INTERRUPTS	HARDWARE INTERRUPTS
They are caused by writing an instruction.	An external device initiates the hardware interrupts by giving an appropriate signal at the interrupt pin of the processor.
8085 has 8 software interrupt instructions called RST N, where N can be any value from 0...7. Hence we have RST 0 ... RST 7.	8085 has 5 hardware interrupt pins: TRAP, RST 7.5, RST 6.5, RST 5.5, INTR
Cannot be masked or disable.	All Hardware interrupts can be disabled except TRAP.
All Software interrupts have the same priority.	Hardware interrupts priority order: TRAP 1 st (Highest) RST 7.5 2 nd RST 6.5 3 rd RST 5.5 4 th INTR 5 th (Lowest)
All Software interrupts are Vectored	All Hardware interrupts are Vectored except INTR.

Vectored and Non-Vectored Interrupts

- ***Vectored Interrupts:*** The address of the subroutine is already known to the Microprocessor. Vector Addresses are calculated by the formula $8 * \text{TYPE}$.
- ***Non-Vectored Interrupts:*** The device will have to supply the address of the subroutine to the Microprocessor.
- *INTR* is the only non-vectorized interrupt in 8085 microprocessor.

Software interrupts vector addresses

INTERRUPT	VECTOR ADDRESS
RST 0	00 H
RST 1	08 H
RST 2	10 H
RST 3	18 H
RST 4	20 H
RST 5	28 H
RST 6	30 H
RST 7	38 H

- **RST n** is equivalent to a subroutine call, in which the content of the program counter is saved in the stack i.e. the program jumps to the instruction starting at restart location.

Hardware interrupts vector addresses

INTERRUPT	VECTOR ADDRESS
TRAP (RST 4.5)	24 H
RST 5.5	2C H
RST 6.5	34 H
RST 7.5	3C H

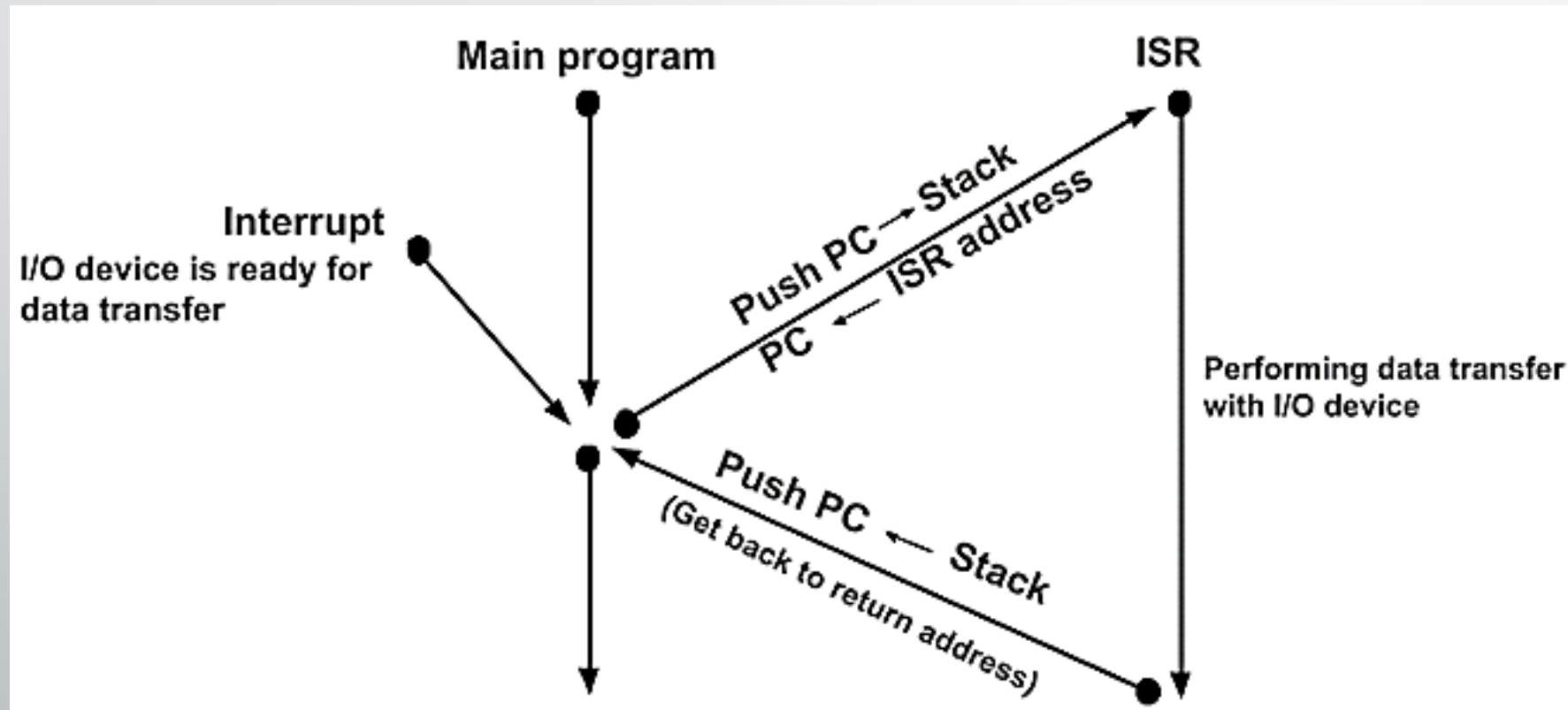
Maskable and Non-Maskable Interrupts

- **Maskable Interrupts** are those which can be disabled or ignored by the microprocessor.
- These interrupts are either edge-triggered or level-triggered.
- *INTR*, *RST 7.5*, *RST 6.5*, *RST 5.5* are maskable interrupts.
- **Non-Maskable** Interrupts are those which cannot be disabled or ignored by microprocessor.
- *TRAP* is a non-maskable interrupt.
- It consists of both level as well as edge triggering and is used in critical power failure conditions.

Interrupt handling mechanism :

1. When an interrupt occurs, the processor first finishes the current instruction.
2. It then suspends the current program and executes an ISR.
3. For this, it pushes the value of PC (address of next instruction) into the stack.
4. Now it loads the ISR address into PC and proceeds to execute the ISR.
5. At the end of the ISR, it pops the return address from the stack and loads it back into PC.

- When an interrupt occurs during the execution of a current program, therefore, after execution of the current instruction, the processor executes ISR. After the execution of ISR, the processor must resume to program exactly as before the interrupt occurred. For this, the content of the PC, content of μP registers and the content of some status conditions is stored. The collection of all status bit conditions in a microprocessor is called PSW(program status word).
- During the interrupt cycle, the contents of PC and PSW are pushed onto the stack. The branch address for the particular interrupt is then passed to PC and a new PSW is loaded into the status register.
- The last instruction in the ISR is the return from interrupted instruction. When this instruction is executed, the old PSW and the return address are popped from the stack.



- Mainly in the microprocessor-based system the interrupts are used for data transfer between the peripheral and the microprocessor.
- The processor will check the interrupts always at the 2nd T-state of last machine cycle.
- If there is any interrupt it accepts the interrupt and send the INTA (active low) signal to the peripheral.

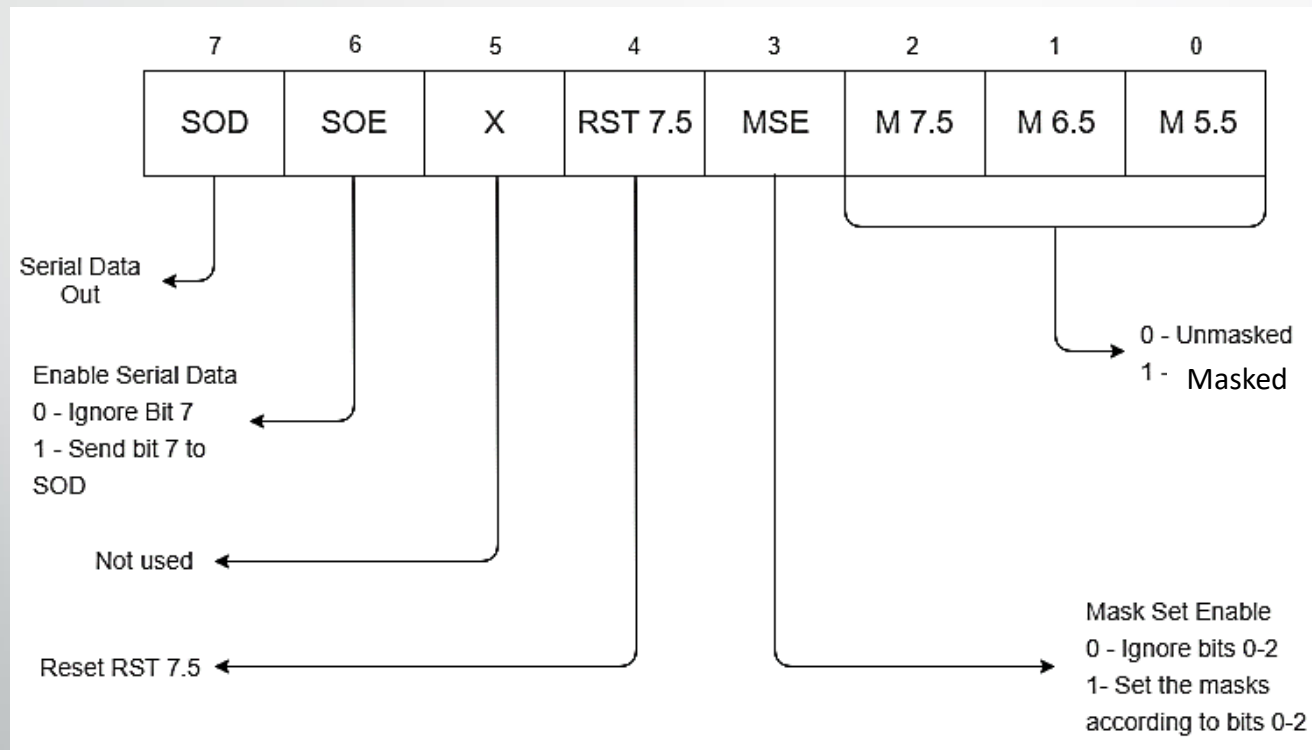
Examples:

- Interrupt request by pressing the keyboard key-Instead of the processor checking all the time, whether a key is pressed, the keyboard interrupts the processor as when we press a key. In the ISR of the keyboard, which is a part of the keyboard driver software, the processor will read the data from the keyboard.

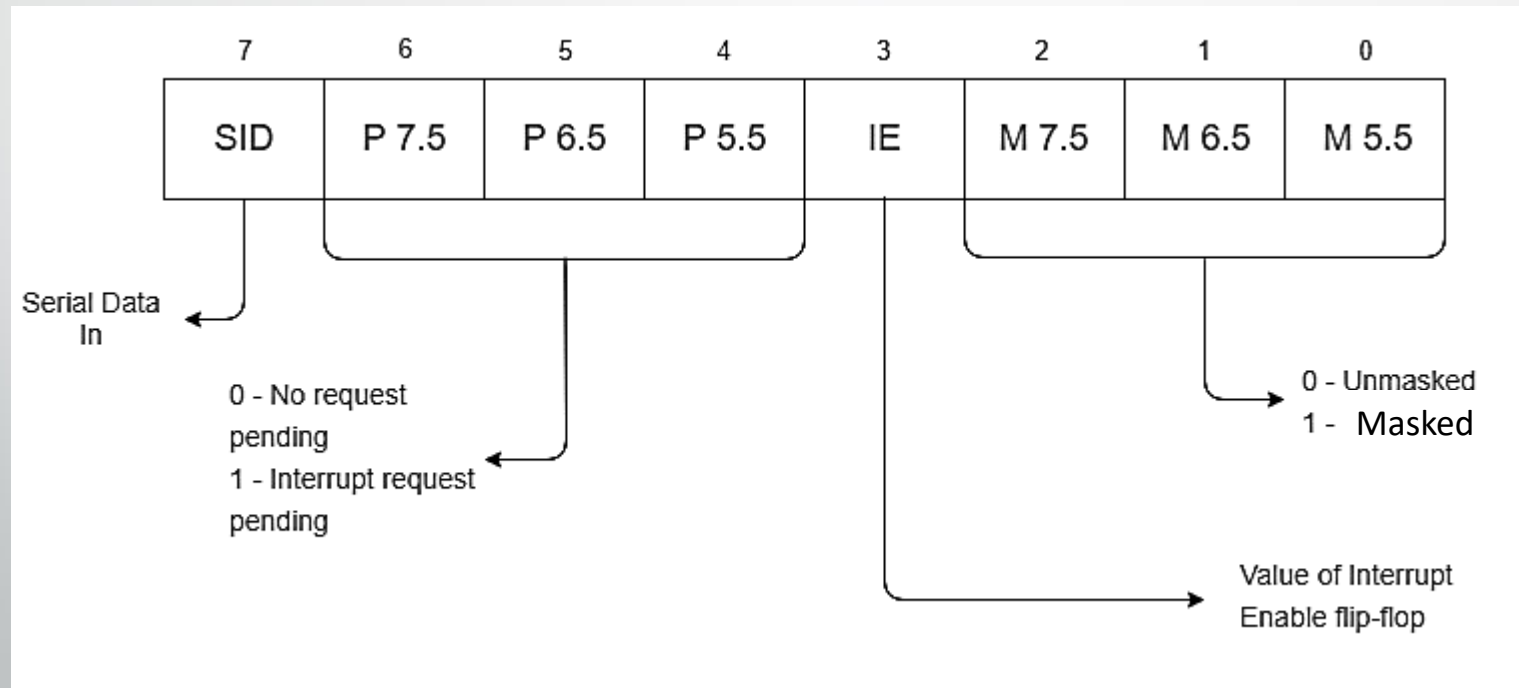
Instruction for Interrupts

- **Enable Interrupt (EI)** – The interrupt enable flip-flop is set and all interrupts are enabled following the execution of next instruction followed by EI. No flags are affected. After a system reset, the interrupt enable flip-flop is reset, thus disabling the interrupts. This instruction is necessary to enable the interrupts again (except TRAP).
- **Disable Interrupt (DI)** – This instruction is used to reset the value of enable flip-flop hence disabling all the interrupts. No flags are affected by this instruction.

- **Set Interrupt Mask (SIM)** – The SIM instruction generally used to mask or unmask RST hardware interrupts. When it executed, the SIM instruction reads the content of accumulator and accordingly mask or unmask the interrupts.



- **Read Interrupt Mask (RIM)** – Used to read the status of the hardware interrupts by loading into the **Accumulator** which defines the condition of the mask bits for the interrupts. It also used to reads the condition of SID (Serial Input Data) bit on the microprocessor.



- The 8085 microprocessor has two dedicated pins for serial data transfer:
 1. Serial Data Output (SOD) ----- Pin No. 4
 2. Serial Data Input (SID) ----- Pin No. 5
- They both are specially made for Input/Output which is further controlled by software.
- The transfer of data is controlled with the help of two instructions, i.e, SIM and RIM.

Methods of preventing an interrupt from occurring

- MASK Individual Bits through SIM Instruction
- Disable all Interrupts through DI Instruction




Masking

- We can prevent an interrupt from occurring by MASKING its individual bit through SIM Instruction. If an interrupt is masked it will not be serviced.
- One of the main advantages of masking as opposed to disabling interrupts is that by masking we can selectively disable a particular interrupt while keeping other interrupts active, whereas through DI instruction all interrupts are disabled.
- ONLY RST 7.5, RST 6.5 and RST 5.5 can be masked by this method.

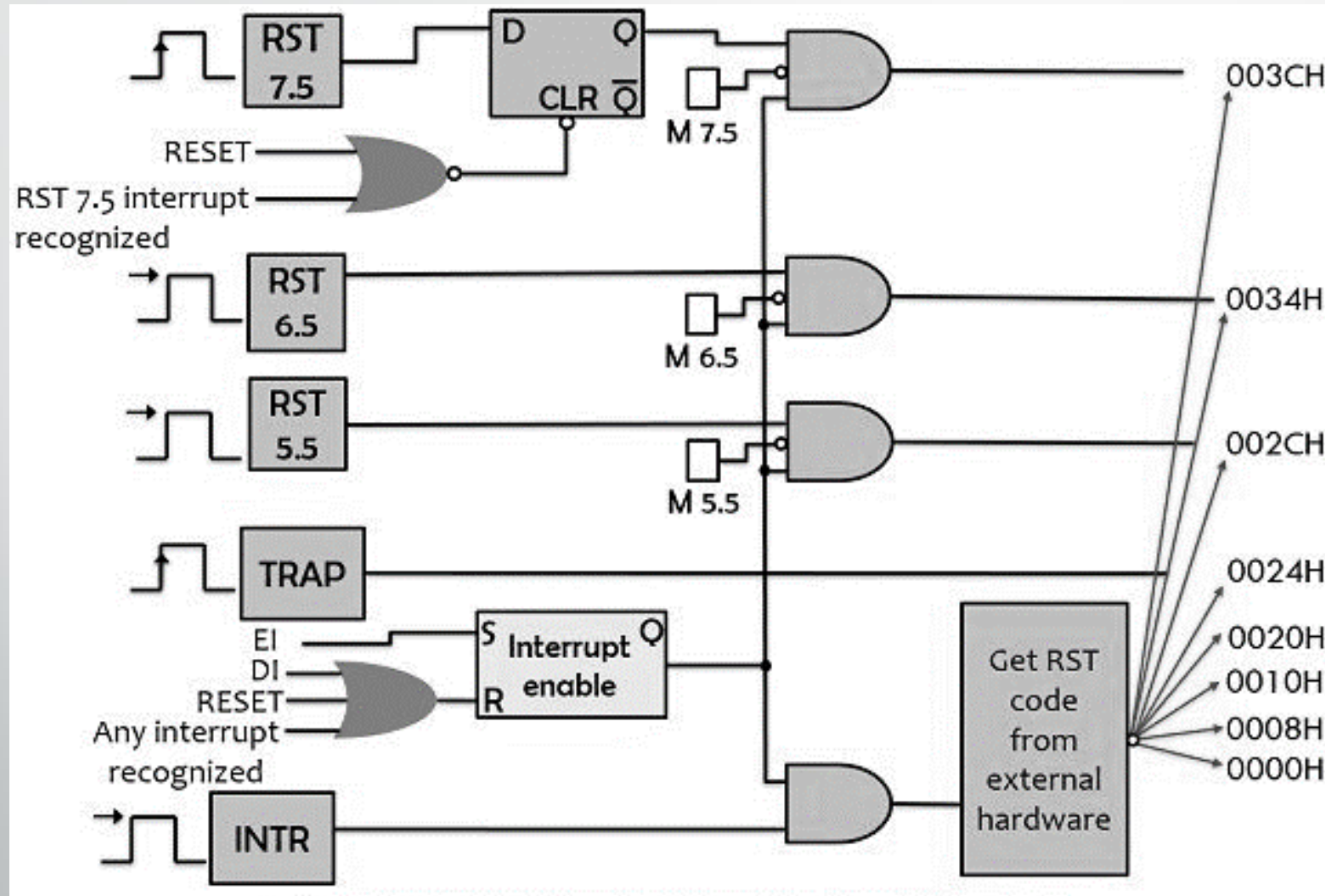
Disabling Interrupts

- Interrupts can be disabled through the DI Instruction.
- This instruction resets the INTE Flip Flop and hence none of the interrupts can occur (Except TRAP) i.e. INTE F/F - 0.
- Once disabled, these interrupts can be re-enabled through EI instruction, which sets the INTE Flip Flop i.e. INTE F/F -1.
- INTE-Interrupt Enable

Hardware Interrupt Triggering

Interrupt	Priority	Trigger	Mask by SIM	Disable by DI	Vector Address
TRAP	1	E/L	Nope	Nope	0024H
RST 7.5	2	E	Yes	Yes	003CH
RST 6.5	3	L	Yes	Yes	0034H
RST 5.5	4	L	Yes	Yes	002CH
INTR	5	L	Nope	Yes	Given by External Hardware
<div></div>					

Interrupt Structure of 8085



TRAP

- TRAP has the **highest priority**.
- It is **Edge as well as Level triggered** hence the signal must go High and also Remain high for some time for it to be recognized. This prevents any noise signal from being accepted.
- It is a Non-Maskable Interrupt i.e. it can **neither be masked nor be disabled**.
- It is a vectored interrupt and has a **vector address of 0024H**.

RST 7.5

- RST 7.5 has the **priority lower than TRAP**.
- It is **Edge triggered**.
- It is a **Maskable Interrupt** i.e. it can be masked through the **SIM Instruction**.
- It can also be disabled through the **DI Instruction**.
- It is a vectored interrupt and has a **vector address of 003CH**.
- RST 7.5 can also be **reset** through the **R 7.5** bit in the **SIM** Instruction irrespective of whether it is Masked or not.

RST 6.5

- RST 6.5 has the **priority lower than RST 7.5**.
- It is **Level triggered**.
- It is a **Maskable Interrupt** i.e. it can be masked through the **SIM Instruction**.
- It can also be disabled through the **DI Instruction**.
- It is a vectored interrupt and has a **vector address of 0034H**.

RST 5.5

- RST 5.5 has the **priority lower than RST 6.5.**
- It is **Level triggered.**
- It is a **Maskable Interrupt** i.e. it can be masked through the **SIM Instruction.**
- It can also be disabled through the **DI Instruction.**
- It is a vectored interrupt and has a **vector address of 002CH.**

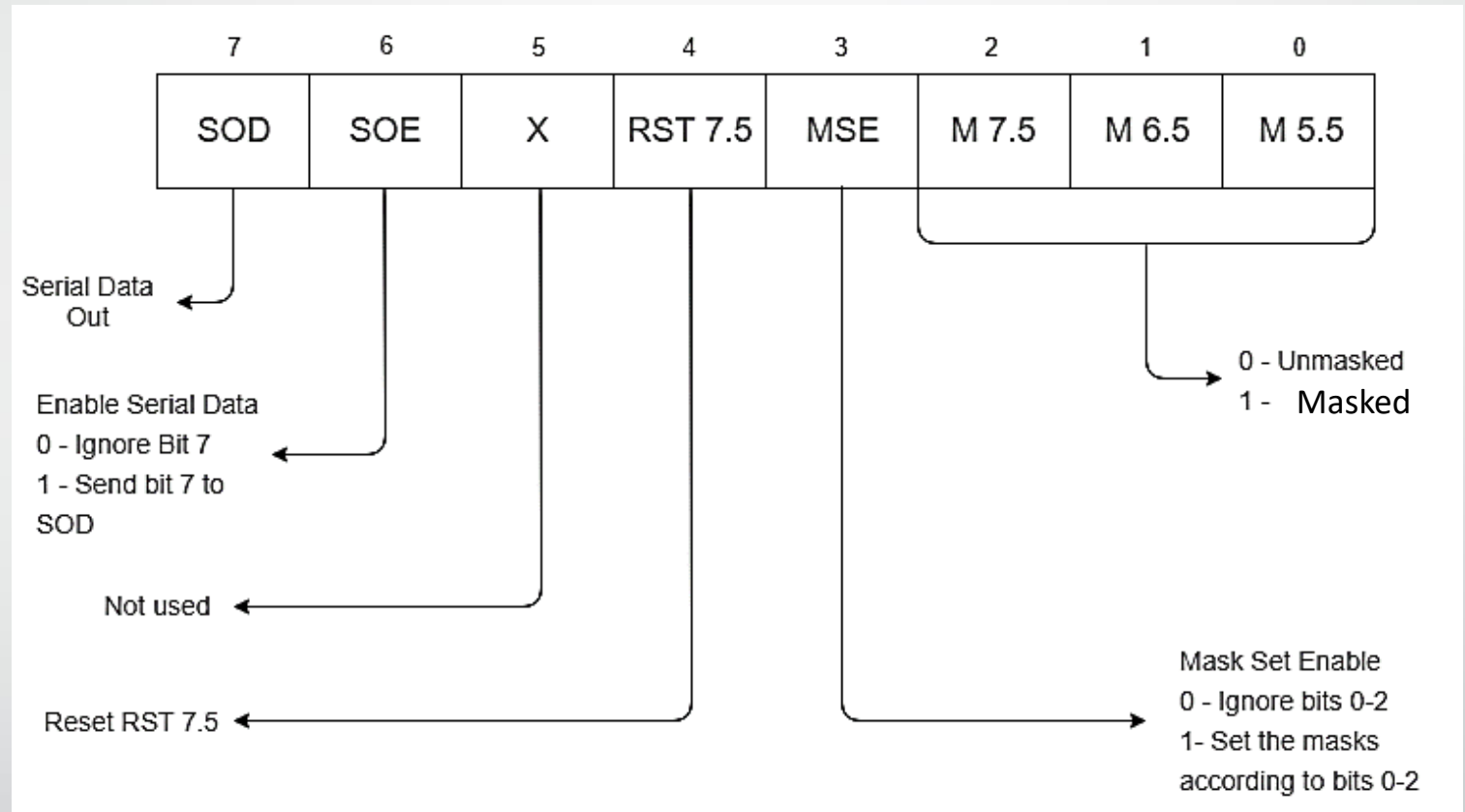
INTR

- INTR has the **priority lower than RST 5.5.**
- It is **Level triggered.**
- It can only be disabled through the **DI Instruction.**
- **It cannot be masked through the SIM Instruction.**
- It is a **Non-Vectored** interrupt.

SIM (Set Interrupt Mask)

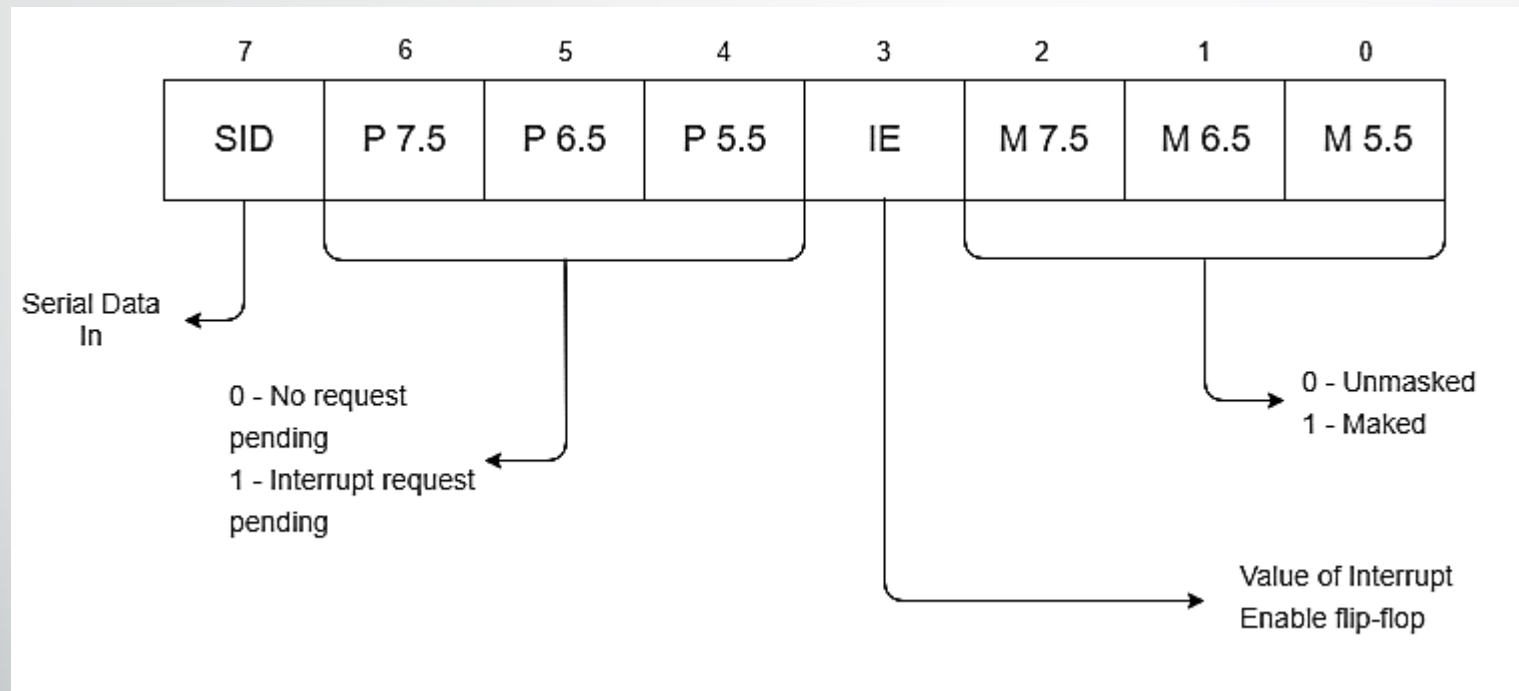
- Purpose of SIM instruction:
 - To Mask or Un-Mask the RST7.5, RST6.5 and RST5.5 interrupts.
 - To send the data out serially (bit - by - bit) through the SOD line of the μ P.
 - To reset RST7.5 interrupt irrespective of whether it is masked or not.
- Method of execution:
 - The appropriate byte is formed and loaded into the Accumulator. Then the SIM Instruction is executed.
 - The UP reads the contents of the accumulator in the above order.
- Note
 - If We have disabled interrupts using DI, then whatever masking we do in SIM is of no use. First, we must enable interrupts using EI instruction. Only Then the masking pattern we give in SIM will come into effect.

Explain the following:
MVI A,C9H
SIM



RIM (Read Interrupt Mask)

- Transfers the contents of the interrupt control logic and serial control logic to accumulator.
- At a time, when more than 1 interrupt requests may occur, higher priority interrupt will be serviced and for lower priority interrupts (pending interrupts), μP stores its information.
- Programmer can monitor the status of pending interrupts.
- Purpose of RIM instruction:
 - 1) Read Interrupt Mask status
 - 2) Accept data serially through SID pin
 - 3) To see pending interrupts of μP .
- Pending Interrupts:
 - Pending interrupts are those interrupts, which are waiting to be serviced.
 - An interrupt becomes pending as a higher priority interrupt is currently being serviced. RIM Instruction indicates the Pending Status of RST7.5, RST6.5 and RST5.5.
- Method of execution:
 - Then the RIM Instruction is executed.
 - The μP loads the appropriate byte into the Accumulator.
 - The programmer reads the contents of the Accumulator.



SQUARE WAVE GENERATION USING SIM

- Write an ALP to generate a square wave of 1kHz using 8085

```
BACK: MVI    A, 40H ; SIM Command = 0100 0000
      SIM
      CALL   DLAY
      MVI    A, C0H ; SIM Command = 1100 0000
      SIM
      CALL   DLAY
      JMP     BACK
```

For a square wave of 1 KHz, the time period is 1 msec.
Hence the required delay is of 0.5 msec.

Assume 8085 is working at 3 MHZ

```

DLAY: MVI    B, XXH        ; 7 T-states ... .. Count is calculated later
BACK: DCR    B              ; 4 T-states ... .. Decrement Count
      JNZ    BACK          ; 10T (true) / 7T (false)
      RET                  ; 10T-states

```

$$T_D = MT + [(Count)_d \times NT] - 3T$$

Here $MT = \text{Time outside the loop} = 17T$
 $NT = \text{Time inside the loop} = 14T$

$$T_D = 17T + [(Count)_d \times 14T] - 3T$$

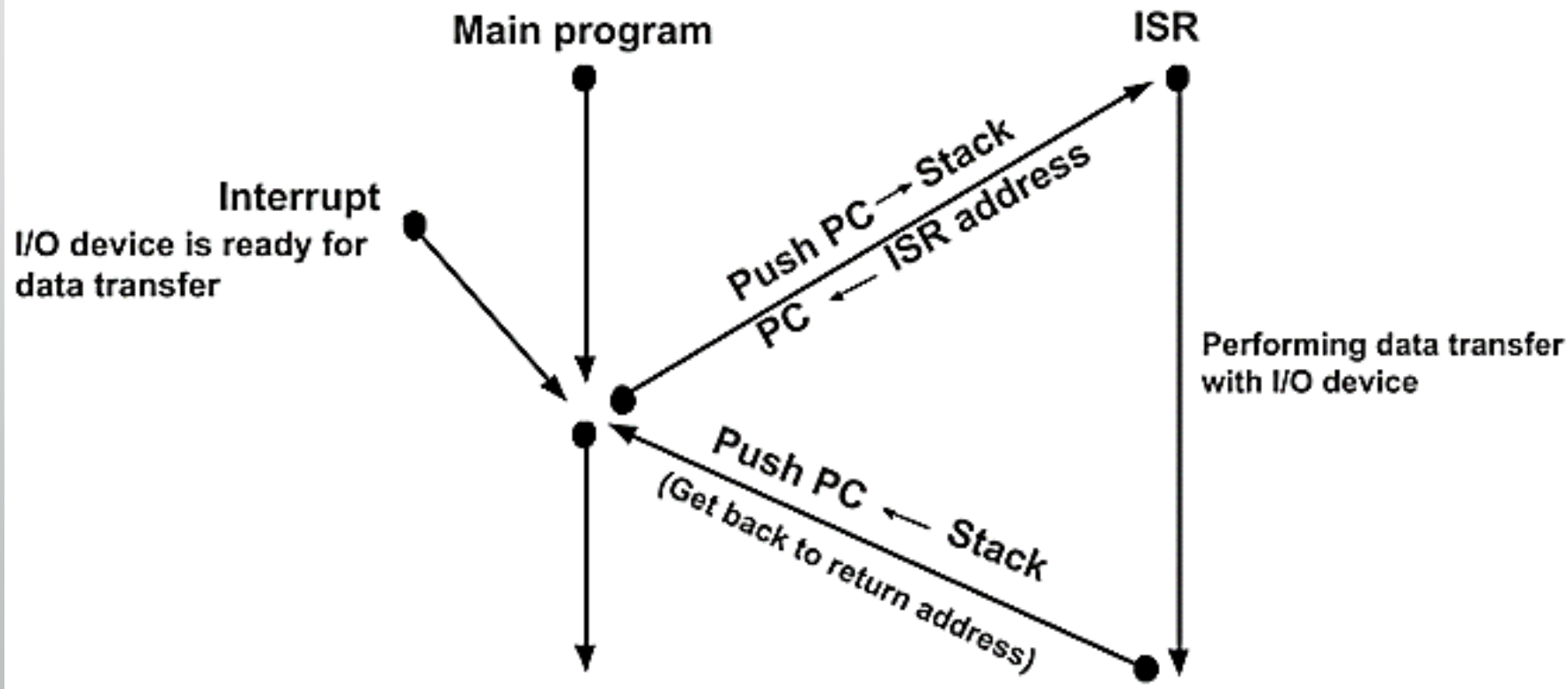
Required $T_D = 0.5 \text{ msec} = 0.5 \times 10^{-3} \text{ sec}$
 $1T = 0.333 \mu\text{sec} = 0.333 \times 10^{-6} \text{ sec}$

Substituting the above values we get:

$$0.5 \times 10^{-3} = 17 \times (0.333 \times 10^{-6}) + [(Count)_d \times 14 \times (0.333 \times 10^{-6})] - 3 \times (0.333 \times 10^{-6})$$

Count = 6AH

Additional Information on Interrupt



Note: “EI” should be written before receiving the interrupt.

Once the ISR starts, the INTE gets disabled automatically. So, if you want to give 2nd interrupt during the ISR of the first interrupt then, you should write “EI” to enable the INTE Flip-flop or the interrupt simple language.

Cont..

Q. WAP to enable all the interrupts of 8085. Mask all the interrupts of 8085 except RST 6.5.

SOD	SDE	X	R7.5	MSE	M7.5	M6.5	M5.5
-----	-----	---	------	-----	------	------	------

$SOD = 0$, $SDE = 0$, $R\ 7.5 = 0$, $MSE = 1$, $M\ 7.5 = 1$, $M\ 6.5 = 0$, $M\ 5.5 = 1$

SIM Word is $00X01101 = 0DH$

LXI SP, 2500H

MVI A, 0DH

SIM

EI

NOP

RET

Cont..

Q. Two bytes of reserved memory are used to store a 16 bit count of the number of times an external event occurs. The least significant of the two bytes is stored in location CNT and the most significant is stored in location CNT + 1. An ISR increments the 16 bit events counter each time it is called. Use RST 6.5 interrupt. WAP to implement such type of event counter.

LXI SP, 2500H	JMP L1
LXI H, 0000H	ISR 6.5: LHLD CNT
SHLD CNT	INX H
MVI A, 0DH	SHLD CNT
SIM	RET
L1: EI	
NOP	
NOP	
HLT	

Cont..

Q. Assuming the microprocessor is completing 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.

SDI	P7.5	P6.5	P5.5	IE	M7.5	M6.5	M5.5
-----	------	------	------	----	------	------	------

RIM

MOV B, A

ANI 20H

RZ

MOV A, B

ANI 0DH

ORI 08H

SIM

EI

JMP ISR ROUTINE/RET

Cont..

Q.WAP to implement a real time clock using RST 7.5 interrupt.

LXI SP, 2500H	NOP	DAA
L4: MVI H, 00H	HLT	MOV H, A
L3: MVI L, 00H	CPI 60H	CPI 24H
L2: MVI C, 00H	JNZ L1	JNZ L3
L1: CALL Display	MOV A, L	JMP L4
MVI A, 0BH	ADI 01H	ISR 7.5: MOV A, C
SIM	DAA	ADI 01H
EI	MOV L, A	DAA
NOP	CPI 60H	MOV C, A
	JNZ L2	RET
	MOV A, H	Display: MOV A, C
	ADI 01H	OUT PORT 1
		MOV A, L
		OUT PORT 2
		MOV A, H
		OUT PORT 3