Interrupt →

- 8085 µp processes interrupt only when Interrupt Enable (EI) flip flop is high & interrupting device sends a INTR signal that is (interrupt Request) signal.

The software instruction used for setting Interrupt enable flip flop is:

Instruction – EI (Enable Interrupt).

- This is 1 Byte instruction
- This sets the interrupt Enable flip flop & enable interrupt process.
- Only system reset or an interrupt disables the interrupt process.

Instruction – DI (Disable Interrupt).

- This is 1 byte instruction.
- This instruction resets interrupt enable F/F.
- It should used in a program segment where an interrupt from an outside source can not be tolerated.

- When µp receives INTR signal, µp completes current instruction, disable Interrupt Enable F/F & sends signal $\overline{INTA}$ to interrupting device. (**) [PC (Prog. Counter) value is stored at Stack)]

Then the Program sequence transfer to the location of Interrupt Service Routine (ISR) of that interrupt.

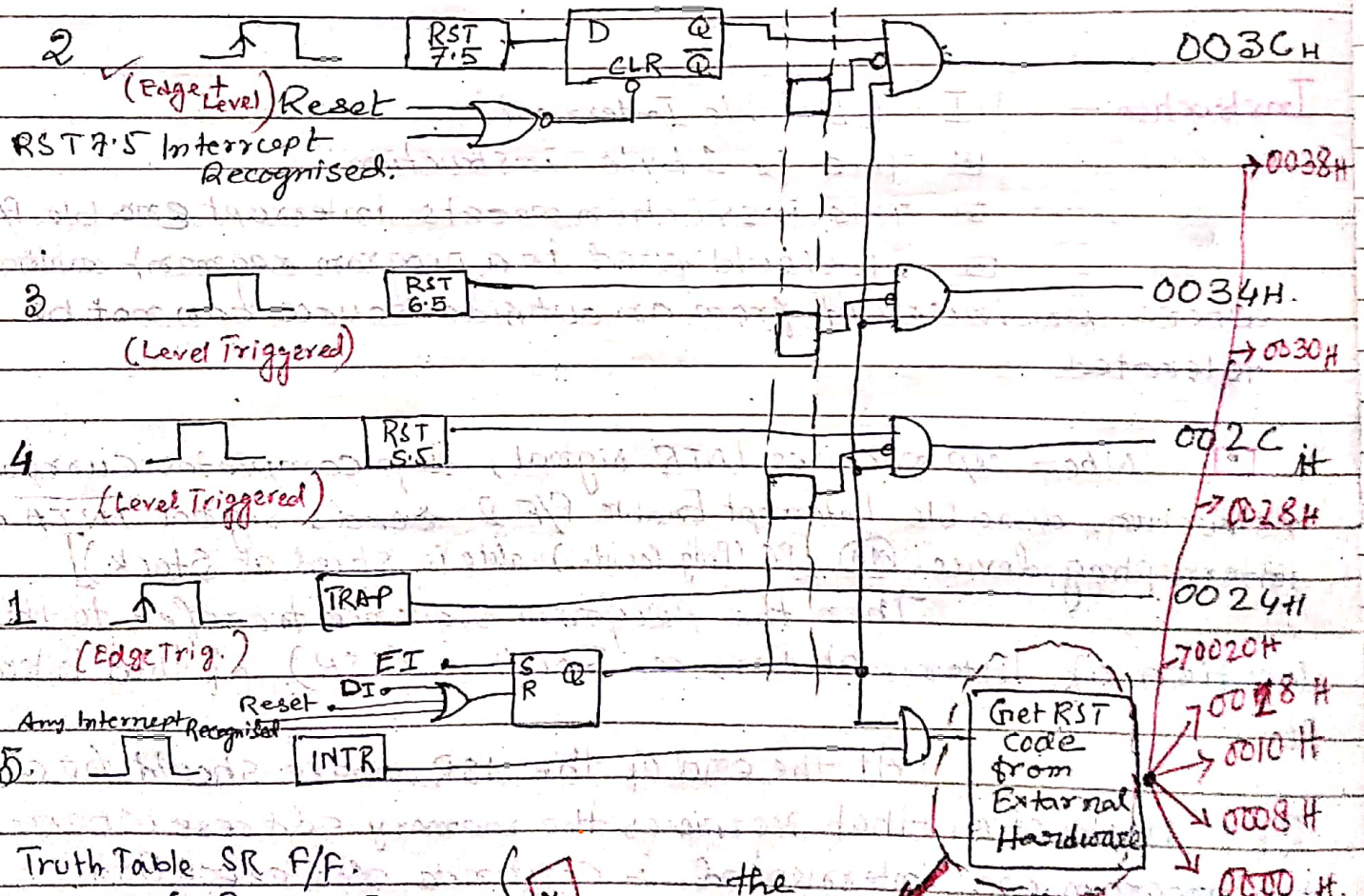
At the end of the ISR there should be a 'RET' instruction that retrieves the memory address where the program was interrupted. & continue the execution. ISR should include EI instruction so that further interrupt can be occured.

(**) The signal $\overline{INTA}$ is used to insert a restart (RST) instruction through external hardware. This instruction transfer the program control to a specific memory location on Page 00H.

8085 Vectored Interrupts :- For vectored interrupts, µp will know the address from where the interrupts are coming & their corresponding starting address of ISR.

Sl. No.	Interrupt	Call Location
1.	TRAP	0024 H.
2.	RST 7.5	003C H
3.	RST 6.5	0034 H.
4.	RST 5.5	002C H.

Trap has highest priority, followed by RST 7.5, 6.5, 5.5 and INTR. However TRAP has lower priority than the HOLD signal used for DMA.



Truth Table SR F/F.

S	R	Q
EI	DI/Reset	
0	0	0
0	1	0
1	0	1
1	1	Undefined

* the INTR is an only one non-vectored interrupt as µp does not know it's ISR location & the corresponding address. So it depends on external hardware.

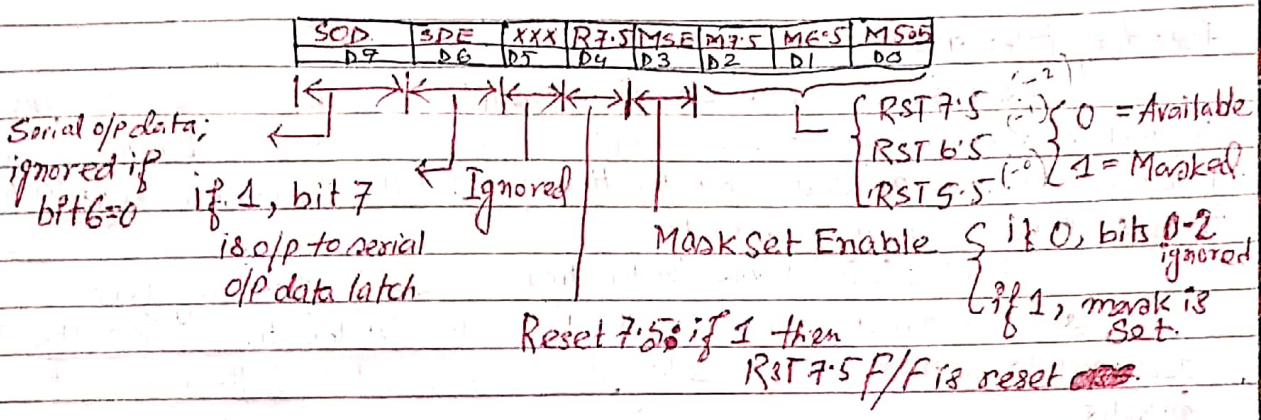
← ONLY FOR THIS CONDITION THE AND GATE WILL BE ACTIVATED.

- Software Interrupt: RST 0, 1, 2, 3, 4, 5, 6, 7. [These are can be generated by writing program (s/w)]
- Hardware Interrupt: RST 7.5, 6.5, 5.5, TRAP, INTR [These are generated externally by peripheral Device]

RST 7.5, 6.5 & 5.5 →

These are maskable interrupts and enabled under program control with two instruction: EI & SIM.

SIM → (SET INTERRUPT MASK)



Three different functions of SIM instruction →

ⓐ Set the mask for RST 7.5, 6.5, 5.5 interrupts. This instruction reads the content of the accumulator and enable/disables the interrupts according to the content of AC.

Bit D3 is a control bit and should = 1 for bits D0, D1 & D2 to be effective. Logic '0' on D0, D1 and D2 will enable the corresponding interrupts & logic '1' will disable the interrupts.

ⓑ 2nd function is reset RST 7.5 flip flop. D4 bit is Contl bit for RST 7.5. If D4 = 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. Bit D7 and D6 of the accumulator are used for serial I/O & don't effect interrupts. Bit D6 = 1 enables the serial I/O and bit D7 is used to transmit (output) bits.

WAP to enable all the interrupts in an 8085 system.

Ans.

Instruction	Comment
EI	; Enable interrupts
MVI A, 08H	; Load bit pattern to enable RST 7.5, 6.5 & 5.5. ✓
SIM	; Enable RST 7.5, 6.5, 5.5

AC ← 0000 1000 (D7 D0)

* RIM interrupt loads the AC
* SIM - AC content need to be loaded ✓

1010 –
1011 –
1100 –
1101 –
1110
1111

WAP to reset the 7.5 interrupt.

Instruction -

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

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.

RIM is one byte instruction & used for following functions-

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

ⓑ To identify pending interrupts. Bits D4, D5 and D6 identify the pending interrupts.

ⓒ To receive serial data. Bit D7 is used for this purpose.

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

↓ Serial i/p data bit.

↓ Pending Interrupts 1 = Pending

↳ Interrupt Enable Flag: 1 = Enabled.

Prob: Assuming µp is completing an RST 7.5 interrupt request, Check to see if RST6.5 is pending. If it is pending, enable RST 6.5 with out affecting any other interrupts; otherwise, return to main prog.

Ans. Instruction: Comment:

```
RIM          ; Read Interrupt Mask.
MOV B, A     ; Save Mask information.
ANI 20H      ; Check whether RST6.5 is pending.
JNZ NEXT
EI
RET          ; RST6.5 is not pending, return to main Prog.
NEXT: MOV A, B   ; Get bit Pattern; RST6.5 is pending
      ANI 0DH    ; Enables RST6.5 by setting D1 = 0
      ORI 08H    ; Enable SIM by setting D3 = 1
      SIM
      JMP SERV   ; Jump to ISR (Interrupt Service Routine) of RST 6.5
```

RST6.5
x x 0 x x x x x
0 0 1 0 0 0 0 0
1: Pending
0: Not pending

Not required

Scanned with CamScanner