CHAPTER

4

8085 Interrupt Structure

4.1 Polling and Interrupts

- Whenever more than one I/O devices are connected to a microprocessor based system, any one of the I/O devices may ask service at any time. There are two methods in which the microprocessor can service these I/O devices. One method is to use the **polling routine**, while the other method employs **interrupt**.
- In the polling routine the microprocessor checks whether any of the I/O devices is requesting for service.
- The polling routine is a simple program that keeps a check for the occurrences of interrupt. For e.g. : Let us assume that our polling routine is servicing I/O ports 1, 2, 3,.....8. The polling routine will check the status of the I/O ports in a proper sequence.
- The polling routine will first transfer the status of the I/O port 1 to the accumulator. It then checks the contents of accumulator to determine if the service request bit is set. If the bit is set then I/O port 1 service routine is called, otherwise the polling routine will move forward to check if port 2 is requesting service. On completion of the service to port1, the polling routine will test port2. The process is repeated till all the 8 ports are tested and all the I/O ports those are demanding service are processed. On completion of the polling routine, the microprocessor will resume with the execution of the program. Fig. 4.1.1 shows the sequence for polling routine.
- The polling routine has priorities assigned to the different I/O devices. Once the routine begins port1 will always be checked first, then port2 and so on.
- Another way that allows the microprocessor to stop the execution of the program and give service to the I/O devices is **interrupt**. It is an external asynchronous input that informs the microprocessor to complete the instruction that it is currently executing and fetch a new routine in order to offer service to the I/O device. Once the I/O device is serviced, the microprocessor will continue with the execution of its normal program.

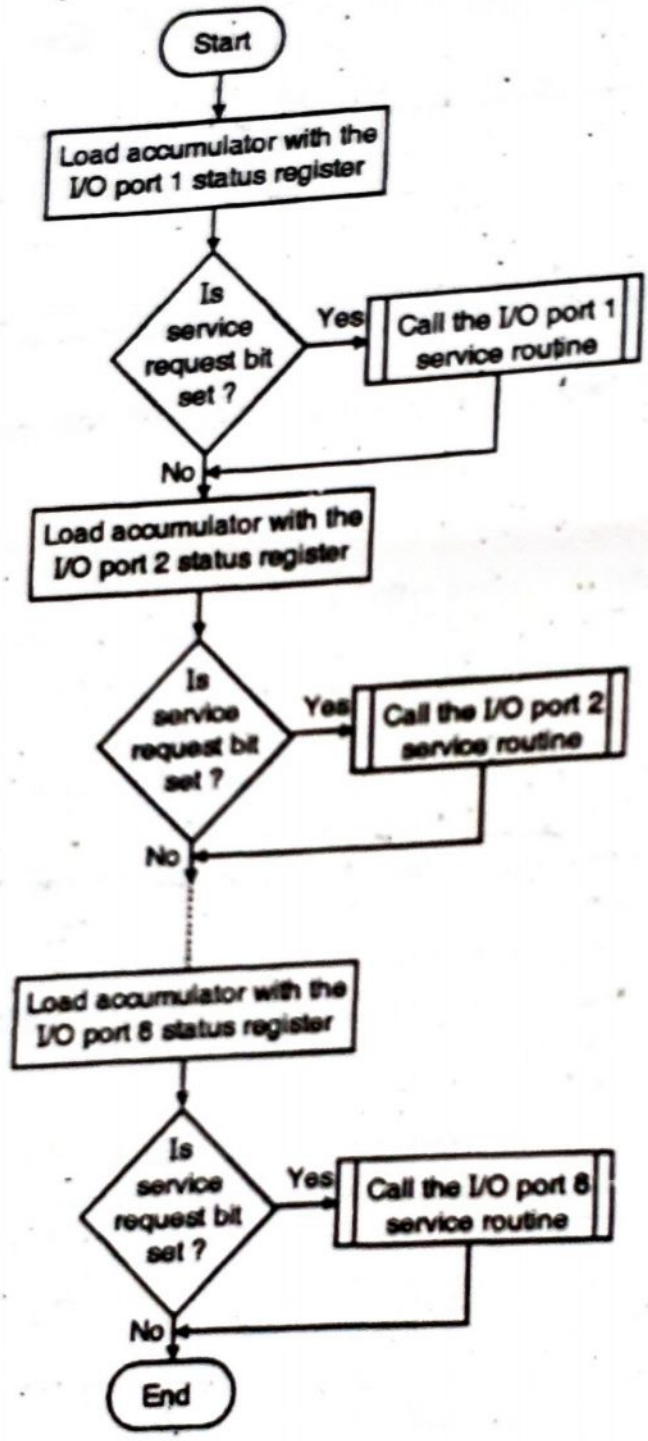

Fig 4.1.1 : Polling sequence

4.2 Basic Definitions of Interrupts

Definitions

1. **Interrupt :** It is a mechanism by which an I/O device (Hardware interrupt) or an instruction (Software interrupt) can suspend the normal execution of the processor and get itself serviced.
2. **Interrupt service routine (ISR) :** A small program or a routine that when executed services the corresponding interrupting source is called as an ISR.

3. Vectored/Non-vectored interrupt

Q. What are vectored and non-vectored interrupts ?

If the ISR address of an interrupt is to be taken from the interrupting sour it is called as a **non-vectored interrupt;** else it is a vectored interrupt. When interrupt request is activated the microprocessor control logic execut machine cycle. During this cycle, it generates starting address of interrupt routine. These addresses are fixed for different interrupts. This memory add interrupt service routine is called as vector location and interrupts which us vector locations are called as *vectored interrupts.*

4. Maskable/Non-maskable interrupt

Interrupt that can be masked (disabled) or unmasked (enabled) by the progr is called as maskable interrupt; else it is a non maskable interrupt.

The maskable and nonmaskable interrupts are enabled and disabled und program control. If the particular flip-flops are set/reset the interrupts masked/unmasked.

In cases where an interrupt is masked, the microprocessor will not respond interrupt even if the interrupt is activated. The interrupts which can be m under the software control are called as **maskable interrupts.**

The interrupts which cannot be masked under the software control are ca **nonmaskable interrupts.**

Q. What is the difference between Non-maskable and maskable interrupt ?

Sr. No.	Non-maskable (NMI)	Maskable
1.	It cannot be masked off or made pending.	It can be masked off or pending.
2.	This interrupt does not disable non-maskable interrupts.	This interrupt disables maskable interrupts.
3.	It is used for emergency purpose, like power failure, smoke detector etc.	It is used to interface periphe
4.	Higher priority.	Lower priority.
5.	Response time is low.	Response time is high.

4.3 8085 Interrupt Structure

- 8085 supports two types of interrupts. They are :
  - Hardware interrupts
  - Software interrupts

4.3.1 Hardware Interrupts

- Peripheral device activates interrupt by activating the respective pin.
- In response to the interrupt request microprocessor completes the c instruction execution in main program and transfer program con interrupt service routine.

- In ISR routine, required task is completed. Task may be to read data, to write data, to update the status, to update the counter etc.
- After completing the task, the program control is transferred back to the main program.
- These types of interrupts where the microprocessor pins are used to receive interrupt requests, are called **hardware interrupts.**
- The microprocessor 8085 has five hardware interrupts they are :

• TRAP	• RST 7.5	• RST 6.5
• RST 5.5	• INTR	

- The interrupt structure is a 5 level structure.
- Fig. 4.3.1 shows the interrupt structure.

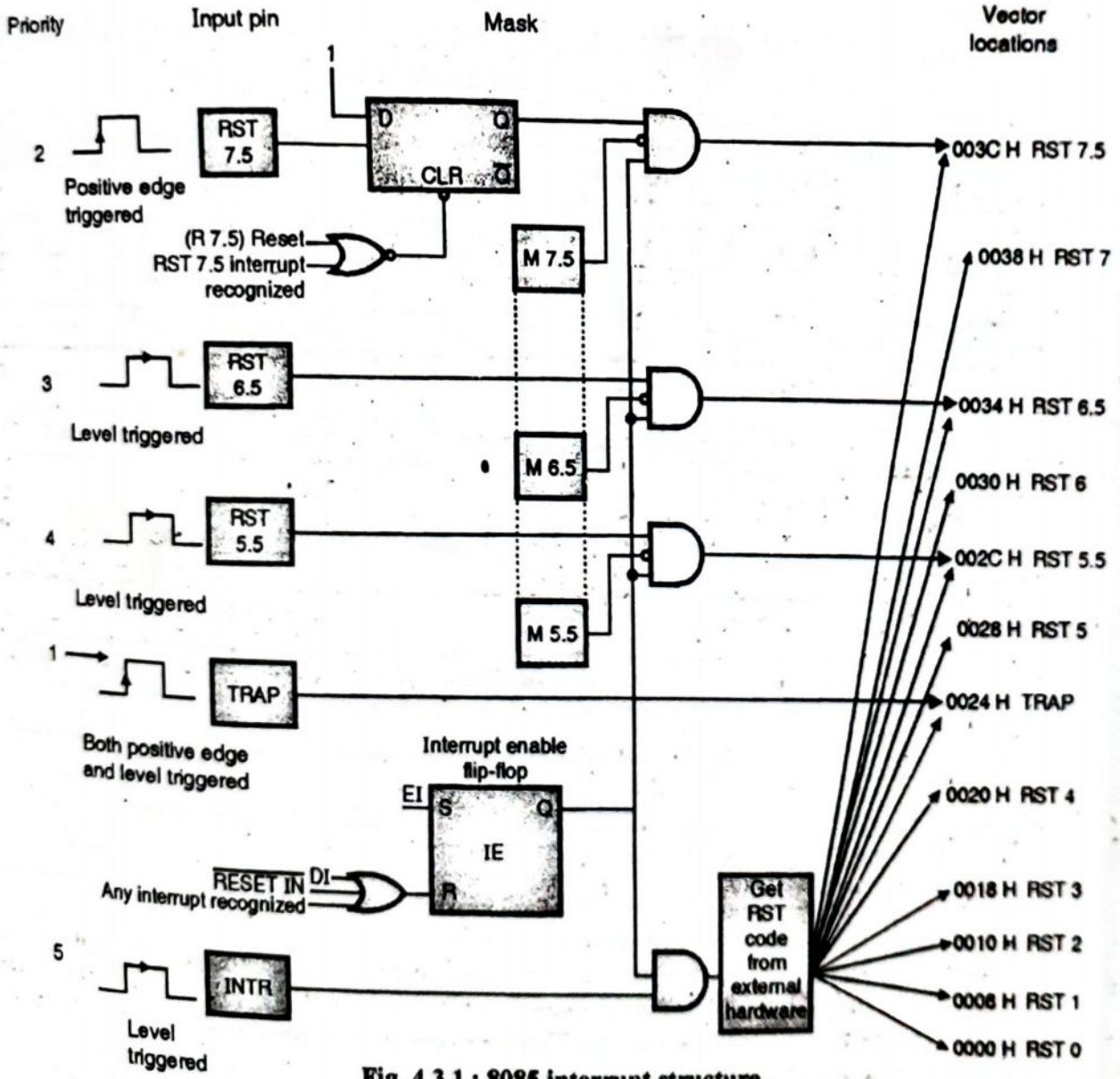

Fig. 4.3.1 : 8085 interrupt structure

4.3.1(A) TRAP

- It is a non maskable edge and level triggered interrupt.
- It is unaffected by any mask or interrupt enable.
- The TRAP signal must make a LOW to HIGH transition and remain HIGH until acknowledged. This avoids false triggering due to noise or glitches.
- It has the highest priority among all interrupts.
- This interrupt transfers the microprocessor's control to location 0024H.

Application

It is used for emergency purpose like power failure, parity error checker, smoke detector etc.

4.3.1(B) RST 7.5

- It is maskable, edge triggered interrupt request input line. This interrupt is triggered at the rising edge of the signal.
- It has highest priority among all maskable interrupts and second priority among all interrupts.
- The interrupt vector location for this interrupt is 003C H.
- Action taken by microprocessor, when it receives RST 7.5.
  - **Step I :** The peripheral activates RST 7.5 signal to set R7.5 flip-flop.
  - **Step II :** When R7.5 flip-flop is set and if INTE flip-flop is set and M7.5 mask latch is reset, the logic activates an internal RST 7.5 request signal.
  - **Step III :** In response to this signal, the microprocessor completes current instruction cycle and calculate starting address of ISR. It activates RST 7.5 acknowledge and any interrupt acknowledge signals to reset R 7.5 and INTE flip-flops respectively.
  - **Step IV :** The microprocessor executes two memory write cycles to save contents of program counter on the stack and then executes corresponding ISR.
- Using SIM instruction the M6.5 and M5.5 flip flop can be set or reset and hence enable or disable RST 6.5.and 5.5.
- DI clears the interrupt enable flip flop, while EI sets the interrupt enable flip flop. (EI and DI are instructions).
- M7.5 and interrupt enable flip flop together decide whether RST 7.5 is enabled or disabled.

4.3.1(C) RST 6.5 and RST 5.5

- These are level triggered, maskable interrupt request input lines.
- RST 6.5 transfers microprocessor's control to location 0034 H while RST 5.5 transfers microprocessor's control to location 002C H.
- Action taken by microprocessor, when it receives RST 6.5 interrupt.
  - **Step I :** The peripheral activates RST 6.5 signal.
  - **Step II :** When RST 6.5 pin is at logic 1, INTE flip-flop is set and M 6.5 mask latch is reset, the logic activates an internal RST 6.5 request signal.
  - **Step III :** The microprocessor completes current instruction cycle and calculate the starting address of ISR. It activates any interrupt acknowledge cycle to reset INTE flip-flops.

Step IV : The microprocessor executes two memory write cycles to save contents of program counter on the stack and then executes corresponding ISR.

- Using SIM instruction the M6.5 and M5.5 flip flop can be set or reset and hence enable or disable RST 6.5.and 5.5.
- DI clears the interrupt enable flip flop, while EI sets the interrupt enable flip flop. (EI and DI are instructions).
- M7.5 and interrupt enable flip flop together decide whether RST 7.5 is enabled or disabled.

Note : The above steps are also true for RST 5.5.

4.3.1(D) INTR

- It is level triggered, maskable interrupt request input line.
- This interrupt works in conjunction with RST N or CALL instruction.
- The INTR logic consists of INTE flip-flop, OR gate and inverter. The INTR pin is logically ANDed with the output of INTE flip-flop. To activate an internal INTR request signal, the INTR pin should be held at logic 1 and INTE flip-flop should be set. This flip-flop is set by executing EI instruction. It is reset by executing DI instruction. It is also reset by $\overline{\text{RESET IN}}$ or any interrupt acknowledge signal. In response to any interrupt the microprocessor resets an INTE flip-flop that means it disables all maskable interrupts.
- Using SIM instruction the M6.5 and M5.5 flip flop can be set or reset and hence enable or disable RST 6.5 and RST 5.5.
- DI clears the interrupt enable flip flop, while EI sets the interrupt enable flip flop. (EI and DI are instructions).
- M6.5 and M5.5 and interrupt enable flip flop together decide whether RST 6.5 and RST 5.5 are enabled or disabled.

Action taken by the microprocessor when it receives an INTR.

Step I : The 8085 microprocessor checks for the status of the INTR signal while it executes each instruction.

Step II : If the INTR signal is high, then the microprocessor completes the execution of the current of the current instruction and in response to the INTR signal sends an active low $\overline{\text{INTA}}$ interrupt acknowledge signal, if the interrupt is enabled.

Step III : In response to the $\overline{\text{INTA}}$ signal, the external logic places on opcode on the data bus. In case of multibyte instruction, additional interrupt acknowledge cycles are generated by the microprocessor, in order to transfer additional bytes to the microprocessor.

Step IV : The 8085 then saves the address of the next instruction onto the stack and executes the received instruction.

4.3.2 Software Interrupts

Q. *Write a short note on the utility of software interrupts.*

- In case of software interrupts the cause of the interrupt is the execution of the instruction.

4.4 Software Interrupts

Q. What are restart instructions ?

- The 8085 microprocessor has eight instructions (RST 0 to RST 7)
- These instructions allow transfer of program control from main program to predefined service routine addresses.
- Predefined service routine is also referred to as ISR.
- After completing the ISR program control is transferred back to main program.
- 8085 microprocessor provides eight software interrupts RST 0 to RST 7. These instructions are used to call interrupt service routines.
- Format of RST N instruction OPCODE is as follows :

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	1	N_2	N_1	N_0	1	1	1

- The vector locations, for RST N instruction are as follows :

Instruction	Address of ISR
RST 0	0000H (8 × 0) = 0000H
RST 1	0008H (8 × 1) = 0008H
RST 2	0010H (8 × 2) = 0010H
RST 3	0018H (8 × 3) = 0018H
RST 4	0020H (8 × 4) = 0020H
RST 5	0028H (8 × 5) = 0028H
RST 6	0030H (8 × 6) = 0030H
RST 7	0038H (8 × 7) = 0038H

16/40
2/8

- The difference between two successive locations is only 8 bytes. Hence jump instruction must be stored into corresponding location to transfer microprocessor's control to user defined ISR address.
- The execution of RST instruction is as follows : Consider an example of RST 2 instruction. When RST 2 instruction is fetched and executed by 8085, it branches to address 0010 H. Before the program control is transferred, the current PC contents (i.e. return address) are stored onto stack. After the ISR of RST 2 is completed the program control is transferred back to main program.
- Software interrupts are not used to handle asynchronous events. They are used to call software routines like single step, break point etc.

16/48
48/3

4.4.1 Comparing Hardware Interrupt and Software Interrupt

Q. What is the difference between hardware and software interrupts ?

Sr. No.	Software interrupt	Hardware interrupt
1.	It is synchronous event.	It is an asynchronous event.
2.	This interrupt is requested by executing instruction.	This interrupt is requested by exte device on a pin.
3.	PC is incremented.	PC is not incremented.
4.	The microprocessor does not execute any interrupt acknowledge cycle or idle machine cycle to acknowledge this interrupt. The microprocessor execute normal instruction cycle.	The microprocessor executes ei interrupt acknowledge cycle bus or machine cycle to acknowledge interrupt.
5.	It cannot be ignored or masked.	It can be masked except for TRAP.
6.	It has highest priority among all interrupts.	The priority is lower than that of softw interrupt.
7.	It does not affect on interrupt control logic.	It affects on interrupt control logic.
8.	It is not used to interface peripherals that means it does not improve throughput of system. It is used in program debugging.	It is used to interface peripherals interrupt driven I/O. It impro throughput of the system.

4.5 Masking / Unmasking of Interrupts

Q. Explain RIM and SIM instruction. How these instructions can be used enable/disable interrupts and for serial I/O ? Explain.

For masking / unmasking of interrupts there are four instructions.

- EI • DI • RIM • SIM

1. EI : Enable Interrupt

This instruction is used to enable all maskable interrupts.. i.e. interrupts RST RST 6.5, RST 5.5 and INTR can be enabled / activated using the EI instruction

- Whenever an interrupt is acknowledged, the interrupt enable flip-flop reset and all the interrupts are disabled.
- If the interrupts are to be enabled, then the EI instruction is to be exec within the ISR.

2. **DI : Disable Interrupt**
   - This instruction resets the interrupt enable flip-flop i.e. it can be used to disable RST 7.5, RST 6.5, RST 5.5 and INTR interrupts.

3. **SIM : Set Interrupt Mask**

Q. Explain RIM and SIM instruction. How there instructions can be used to enable/disable interrupts and for serial I/O? Explain.

- This instruction is used to enable / disable the RST 7.5, RST 6.5 and RST 5.5 interrupts.
- This instruction does not affect on the TRAP and INTR inputs.
- It can also be used for serial data transmission.
- It transfers the control word from accumulator to the interrupt control logic, and the serial control logic. So it is essential to load the (SIM format) control word into the accumulator before the execution of SIM instruction.

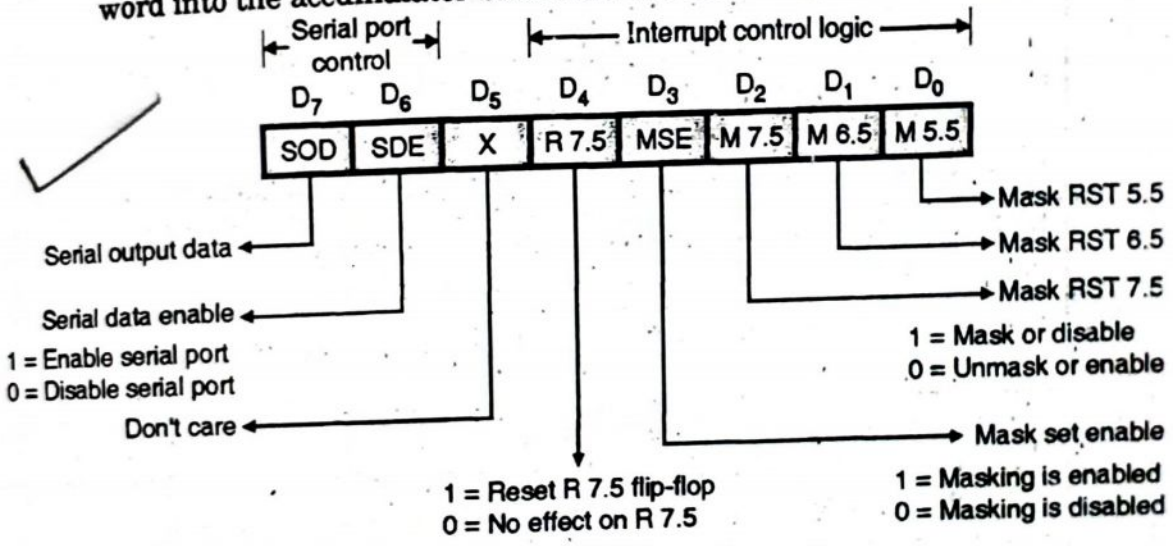

Fig. 4.5.1 : SIM format

- Bits D_7 and D_6 are serial port control. The SDE is enable bit used to enable/disable serial output data. If D_6 bit is enabled, D_7 bit is transferred to SOD pin.
- Bit D_4 is R 7.5 part of interrupt control logic. It is used to reset R 7.5 flip flop regardless of RST 7.5 masking.
- Bits D_3 to D_0 are part of interrupt control logic. These bits are used to mask RST 5.5, RST 6.5 and RST 7.5 interrupts. The MSE bit is master control over M 7.5, M 6.5 and M 5.5 bits. If MSE = 0, the M bits have no effect, but if MSE = 1, the M bits decides mask or unmask of respective interrupts.

Example

To enable RST 6.5 and disable RST 7.5 and RST 5.5 the following SIM format is required.

SOD	SDE	X	R 7.5	MSE	M 7.5	M 6.5	M 5.5	= 0D H
0	0	0	0	1	1	0	1	

```
MVI A, 0D H   : Load SIM format in accumulator
SIM           : Set interrupt mask
```

29

4. RIM : Read Interrupt Mask

- This instruction is used to check the status of all pending and maskable interrupts.
- It can also transfer serial data bit from the serial input data line (SID) to the D_7 bit of the accumulator.
- This instruction transfers the contents of the interrupt control logic and serial control logic to the accumulator. Hence, accumulator is loaded with the status format after execution of the RIM instruction.
- At a time there can be more than one interrupt requests may occur. In such cases if the priority of interrupts is higher than they are serviced, if the interrupts are of lower priority then the 8085, stores the information about these interrupt requests. Such interrupts are called **pending interrupt**. The programmer can monitor the status of these pending interrupts using the RIM instruction.

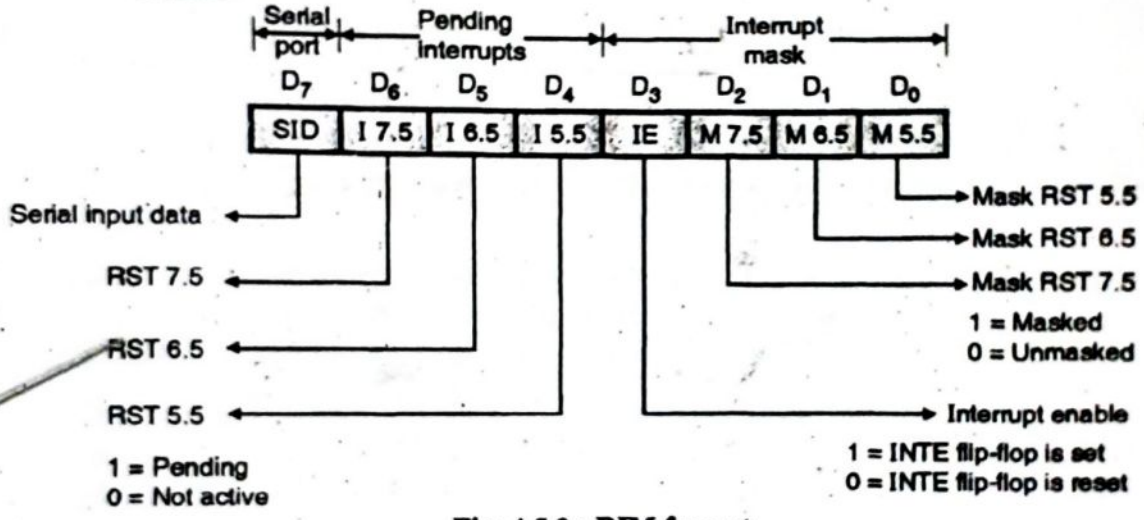

Fig. 4.5.2 : RIM format

- Bit D_7 is status of SID pin on serial port. When RIM instruction is executed the logic level of SID pin is copied at bit D_7.
- Bits D_6, D_5 and D_4 are status of pending interrupts. If RST 7.5 and RST 5.5 inputs occurs the 8085 will branch to ISR of RST 7.5. As the RST 5.5 is having less priority then RST 7.5 so it will be pending interrupt. When RIM instruction is executed the status of RST 7.5, RST 6.5 and RST 5.5 pending is copied at bits D_6, D_5 and D_4 respectively.
- Bits D_3 to D_0 are status of interrupt enable flip flop, mask 7.5, mask 6.5 and mask 5.5. When RIM instruction is executed the status of masking is loaded at bits D_3 to D_0.

Example

(a) RIM instruction is executed and accumulator bit pattern is 0010 0000. The bit pattern indicates that RST 6.5 is pending interrupt.

(b) RIM instruction is executed and accumulator bit pattern is 0000 1010. The bit pattern indicates interrupt enable flip flop is set, RST 6.5 is masked while RST 7.5 and RST 5.5 are unmasked.