Microprocessors

Dr Mridul Gupta

Assistant Professor

Dept. of ECE

UNIT-2

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.
- Interrupts can be classified into two types:
  - Maskable(can be delayed)
  - Non-Maskable(can not be delayed)

Interrupts

- 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)

Interrupts

- An interrupt is considered to be an emergency signal.
  - The Microprocessor should respond to it as soon as possible.

- When the Microprocessor 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.

Responding to Interrupts

- 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 8085 Interrupts

- 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.

The 8085 Interrupts

- The 8085 has **5** interrupt inputs:
- The INTR input is the only non-vectored interrupt.
  - INTR is maskable using the EI/DI instruction pair.

- RST 5.5, RST 6.5, RST 7.5 are all automatically vectored.
  - RST 5.5, RST 6.5, and RST 7.5 are all maskable.

- TRAP is the only non-maskable interrupt in the 8085.
  - TRAP is also automatically vectored.

The 8085 Interrupts

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

Interrupt Vectors and 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 purpose of the IVT is to **hold the vectors that redirect the microprocessor** to the right place when an interrupt arrives.
  - The IVT is divided into several blocks. Each block is used by one of the interrupts to hold its “vector”

The 8085 Non-Vectored Interrupt Process

Step 1: The interrupt process should be enabled by writing the instruction EI in the main program. This is similar to keeping the phone receiver on the hook. The instruction EI sets the Interrupt Enable flip-flop. The instruction DI resets the flip-flop and disables the interrupt process.

Instruction EI (Enable Interrupt)

- ☐ This is a 1-byte instruction.
- ☐ The instruction sets the Interrupt Enable flip-flop and enables the interrupt process.
- ☐ System reset or an interrupt disables the interrupt process.

Instruction DI (Disable Interrupt)

- ☐ This is a 1-byte instruction.
- ☐ The instruction resets the Interrupt Enable flip-flop and disables the interrupt.
- ☐ It should be included in a program segment where an interrupt from an outside source cannot be tolerated.

The 8085 Non-Vectored Interrupt Process

Step 2: When the microprocessor is executing a program, it checks the INTR line during the execution of each instruction.

Step 3: If the line INTR is high and the interrupt is enabled, the microprocessor completes the current instruction, disables the Interrupt Enable flip-flop and sends a signal called $\overline{\text{INTA}}$—Interrupt Acknowledge (active low). The processor cannot accept any interrupt requests until the interrupt flip-flop is enabled again.

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

Step 5: When the microprocessor receives an RST instruction (or a Call instruction), it saves the memory address of the next instruction on the stack. This is similar to inserting a bookmark. The program is transferred to the CALL location.

The 8085 Non-Vectored Interrupt Process

Step 6: Assuming that the task to be performed is written as a subroutine at the specified location, the processor performs the task. This subroutine is known as a service routine.

Step 7: The service routine should include the instruction EI to enable the interrupt again. This is similar to putting the receiver back on the hook.

Step 8: At the end of the subroutine, the RET instruction retrieves the memory address where the program was interrupted and continues the execution. This is similar to finding the page where you were interrupted by the phone call and continuing to read.

The 8085 Non-Vectored Interrupt Process

- The 8085 recognizes 8 RESTART instructions: RST0 -RST7.
  – each of these would send the execution to a predetermined hard-wired memory location:

Restart Instruction	Equivalent to
RST0	CALL 0000H
RST1	CALL 0008H
RST2	CALL 0010H
RST3	CALL 0018H
RST4	CALL 0020H
RST5	CALL 0028H
RST6	CALL 0030H
RST7	CALL 0038H

RST (Restart) Instruction

- The 8085 instruction set includes 8 RST instructions.
- These are one byte CALL instructions that transfers the program execution to a specific location listed in Table A.
- The address in the program counter (i.e. the address the next instruction to an RST instruction) is stored on the stack before the program execution is transferred to the RST call location.

- In case of a hardware interrupt, we will use an RST instruction to restart the program execution.

RST (Restart) Instruction

- Table A.

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

Restart Sequence

- The restart sequence is made up of three machine cycles
- In the 1st machine cycle:
  - The microprocessor sends the INTA signal.
  - While INTA is active the microprocessor reads the data lines expecting to receive, from the interrupting device, **the opcode for the specific RST instruction**.
- In the 2nd and 3rd machine cycles:
  - **the 16-bit address of the next instruction is saved on the stack**.
  - Then the microprocessor jumps to the address associated with the specified RST instruction.

Restart Sequence

- The location in the IVT associated with the RST instruction can not hold the complete service routine.

  – The routine is written somewhere else in memory.

  – Only a JUMP instruction to the ISR’s location is kept in the IVT block.

Hardware Generation of RST Opcode

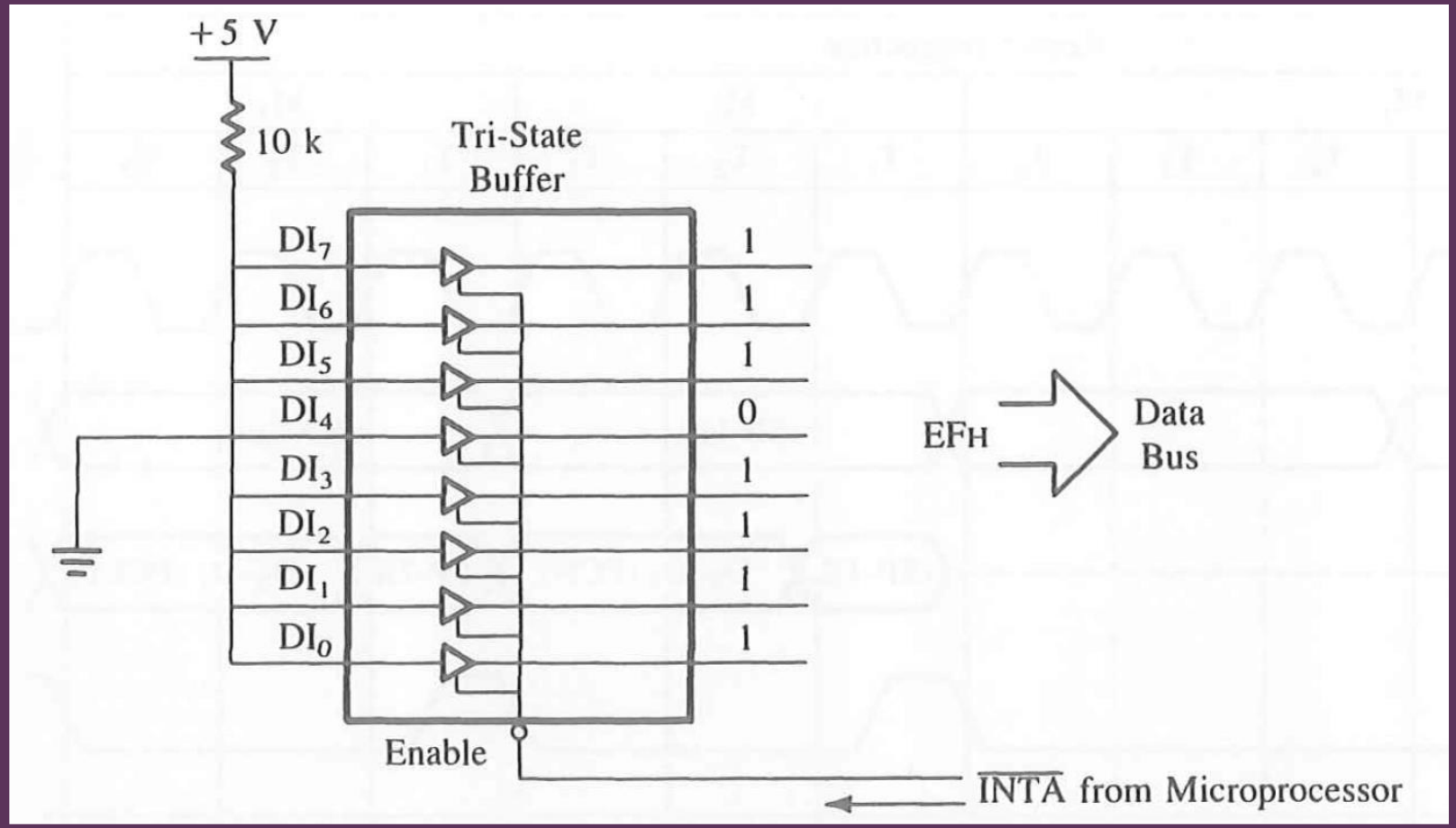
- How does the external device produce the opcode for the appropriate RST instruction?

  – The opcode is simply a collection of bits.

  – So, the device needs to set the bits of the data bus to the appropriate value in response to an INTA signal.

Hardware Generation of RST Opcode

- The following is an example of generating RST 5:
  - RST 5's opcode is EF=11101111



Hardware Generation of RST Opcode

- During the interrupt acknowledge machine cycle, (the 1st machine cycle of the RST operation):
  - The Microprocessor activates the INTA signal.
  - This signal will enable the Tri-state buffers, which will place the value EFH on the data bus (Bits DI3, 4 and 5 will be altered).
  - Therefore, sending the Microprocessor to the RST 5 instruction.

The RST 5 instruction is exactly equivalent to CALL 0028H

Issues in Implementing INTR Interrupts

- How long must INTR remain high?
  - – The microprocessor checks the INTR line one clock cycle before the last T-state of each instruction.
  - – The INTR must remain active long enough to allow for the longest instruction.
  - – The longest instruction for the 8085 is the conditional CALL instruction which requires 18 T-states.

Therefore, the INTR must remain active for 17.5 T-states.

Issues in Implementing INTR Interrupts

- How long can the INTR remain high?
  - – The INTR line must be deactivated before the EI is executed. Otherwise, the microprocessor will be interrupted again.
  - – The worst case situation is when EI is the first instruction in the ISR.
  - – Once the microprocessor starts to respond to an INTR interrupt, INTA becomes active (=0).

Therefore, INTR should be turned off as soon as the INTA signal is received.

Issues in Implementing INTR Interrupts

- Can the microprocessor be interrupted again before the completion of the ISR?
  - – As soon as the 1st interrupt arrives, all maskable interrupts are disabled.
  - – They will only be enabled after the execution of the EI instruction.

Therefore, the answer is: “only if you allow it to”. If the EI instruction is placed early in the ISR, other interrupt may occur before the ISR is done.

Multiple Interrupts & Priorities

- How do we allow multiple devices to interrupt using the INTR line?
  - – The microprocessor **can only respond to one signal** on INTR at a time.
  - – Therefore, we must allow the signal from only one of the devices to reach the microprocessor.
  - – We must **assign some priority** to the different devices and allow their signals to reach the microprocessor according to the priority.

The 8085 Maskable/Vectored Interrupts

- The 8085 has 4 Masked/Vectored interrupt inputs.
  - RST 5.5, RST 6.5, RST 7.5
    - They are all maskable.
    - They are automatically vectored according to the following table:

The vectors for these interrupt fall **in between** the vectors for the RST instructions.
That's why they have names like RST 5.5 (RST 5 and a half).

Interrupt	Vector
RST 5.5	002CH
RST 6.5	0034H
RST 7.5	003CH

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.

The 8085 Maskable/Vectored Interrupt Process

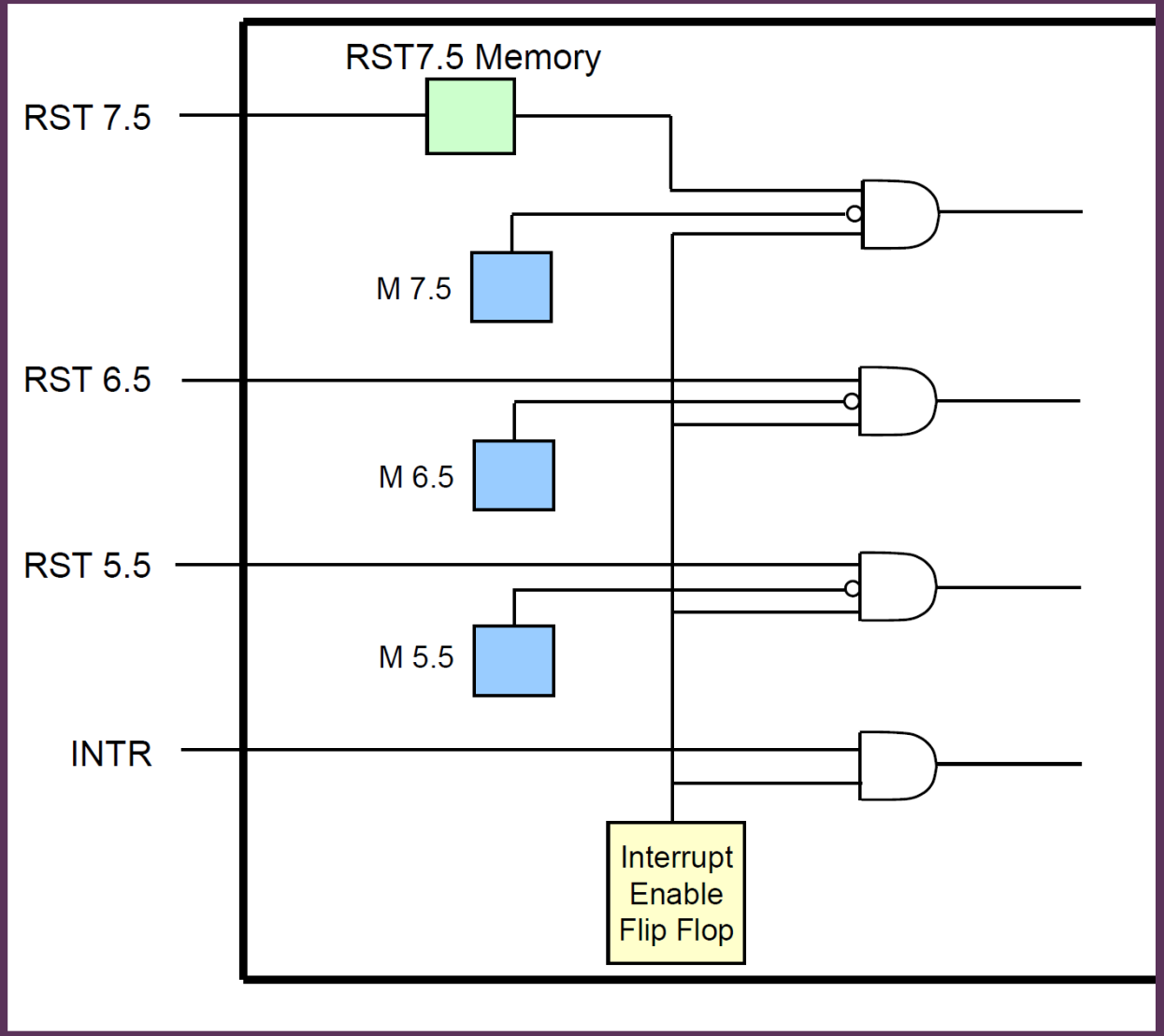
5. When the microprocessor executes the call instruction, it saves the address of the next instruction on the stack.
6. The microprocessor jumps to the specific service routine.
7. The service routine must include the instruction EI to re-enable the interrupt process.
8. At the end of the service routine, the RET instruction returns the execution to where the program was interrupted.

Masking RST 5.5, RST 6.5 and RST 7.5

- These three interrupts are masked at two levels:
  - Through the Interrupt Enable flip flop and the EI/DI instructions.
    - The Interrupt Enable flip flop controls the whole maskable interrupt process.

- Through individual mask flip flops that control the availability of the individual interrupts.
  - These flip flops control the interrupts individually.

Maskable Interrupts

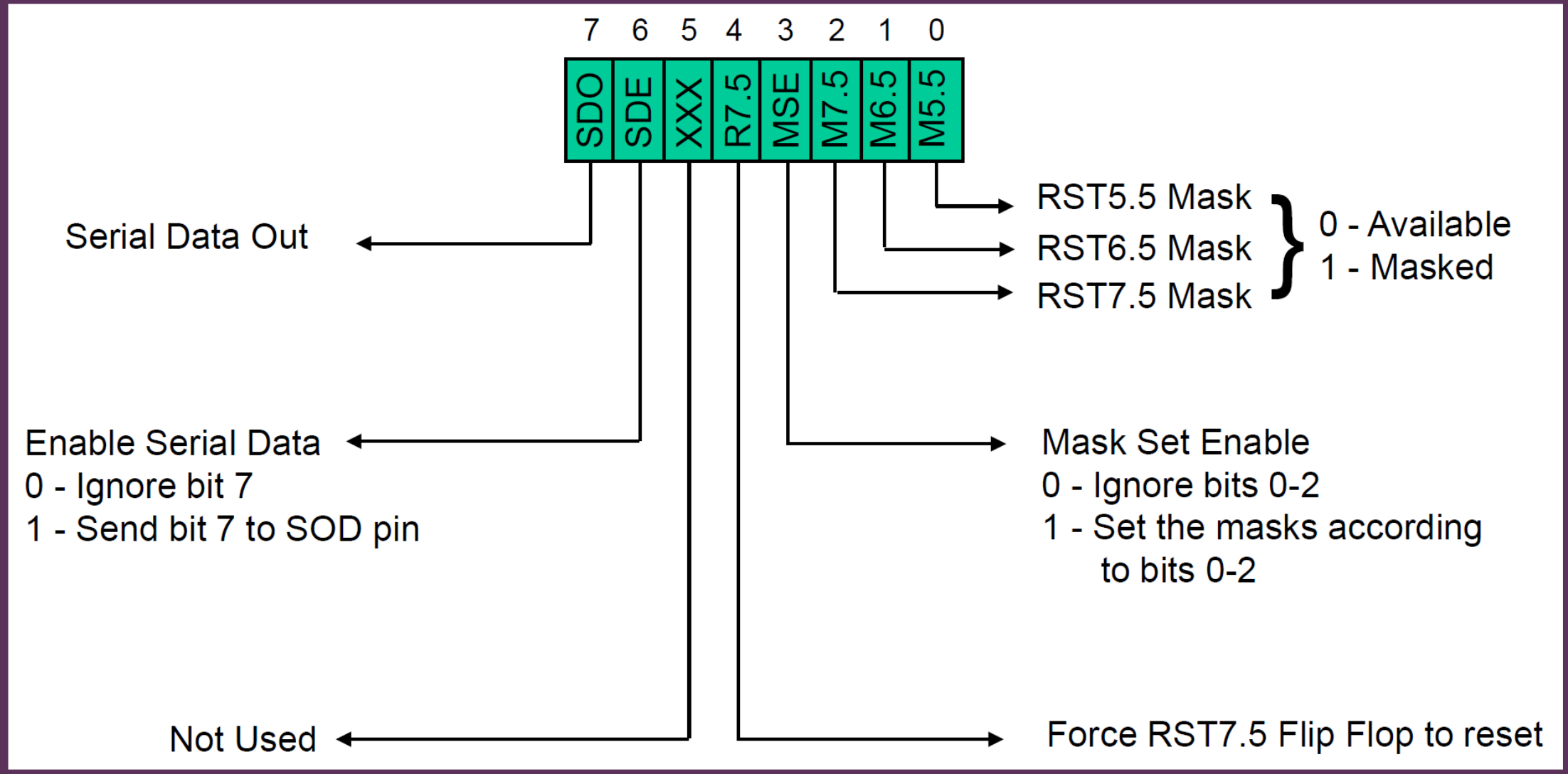


Manipulating the Masks

- The **Interrupt Enable flip flop is manipulated using the EI/DI instructions**.

- The individual masks for RST 5.5, RST 6.5 and RST 7.5 are manipulated using the SIM instruction.
  - –This instruction takes the bit pattern in the Accumulator and applies it to the interrupt mask enabling and disabling the specific interrupts.

How SIM Interprets the Accumulator



SIM and the Interrupt Mask

- 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.

SIM and the Interrupt Mask

- 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.
  - The SIM instruction is used for multiple purposes and not only for setting interrupt masks.
    - It is also used to control functionality such as **Serial Data Transmission**.
    - Therefore, bit 3 is necessary to tell the microprocessor whether or not the interrupt masks should be modified

SIM and the Interrupt Mask

- The RST 7.5 interrupt is the **only** 8085 interrupt that has **memory**.
  - If a signal on RST7.5 arrives while it is masked, a flip flop will remember the signal.
  - When RST7.5 is unmasked, the microprocessor will be interrupted even if the device has removed the interrupt signal.
  - This flip flop will be automatically reset when the microprocessor responds to an RST 7.5 interrupt.

SIM and the Interrupt Mask

- 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.

- The SIM instruction can also be used to perform serial data transmission out of the 8085's SOD pin.
  - One bit at a time can be sent out serially over the SOD pin.

SIM and the Interrupt Mask

- Bit 6 is used to tell the microprocessor whether or not to perform serial data transmission
  - If 0, then do not perform serial data transmission
  - If 1, then do.

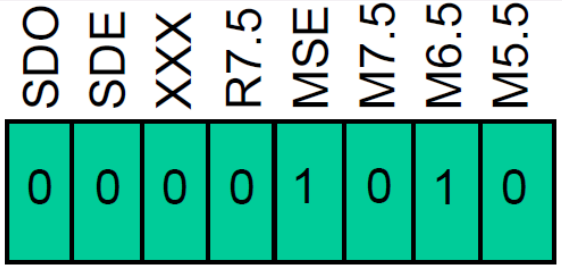
- The value to be sent out on SOD has to be placed in bit 7 of the accumulator.

- Bit 5 is not used by the SIM instruction

Using the SIM Instruction to Modify the Interrupt Masks

- 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



Contents of accumulator are: 0AH

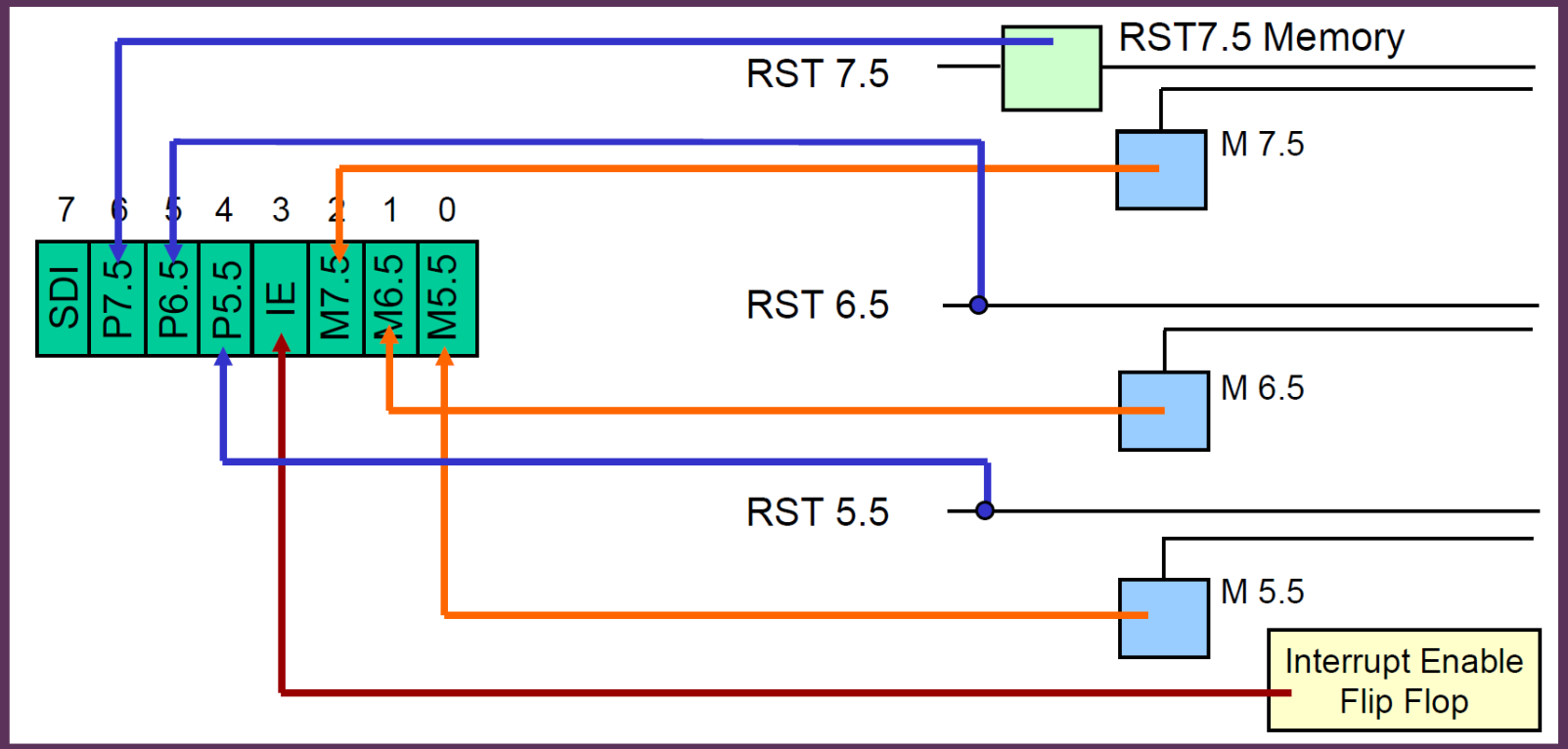
```
EI          ; Enable interrupts including INTR
MVI A, 0A   ; Prepare the mask to enable RST 7.5, and 5.5, disable 6.5
SIM         ; Apply the settings RST masks
```

Triggering Levels

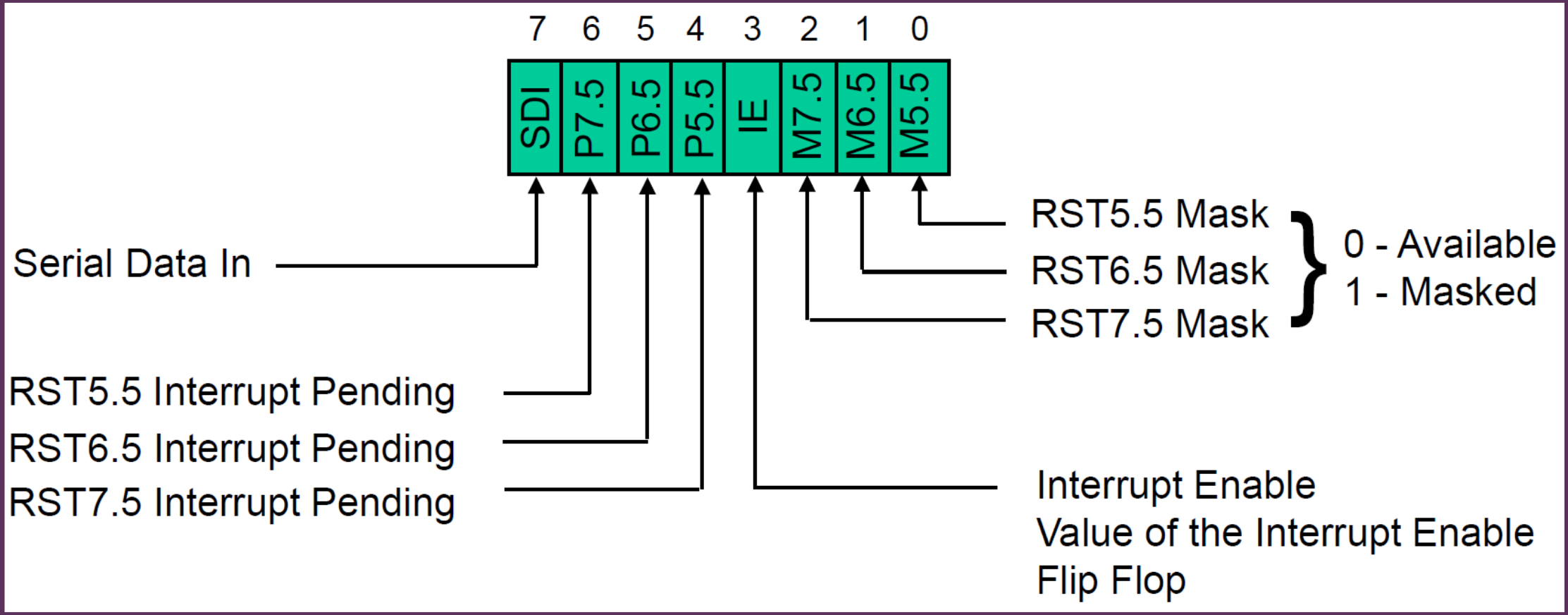
- RST 7.5 is positive edge sensitive.
  - When a positive edge appears on the RST7.5 line, a logic 1 is stored in the flip-flop as a “pending” interrupt.
  - Since the value has been stored in the flip flop, the line does not have to be high when the microprocessor checks for the interrupt to be recognized.
  - The line must go to zero and back to one, before a new interrupt is recognized.
- RST 6.5 and RST 5.5 are level sensitive.
  - The interrupting signal must remain present until the microprocessor checks for interrupts.

Determining the Current Mask Settings

- RIM instruction: Read Interrupt Mask
  - – Load the accumulator with an 8-bit pattern showing the status of each interrupt pin and mask.



How RIM sets the Accumulator's different bits



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.

Using RIM and SIM to set Individual Masks

- Example: Set the mask to enable RST6.5 without modifying the masks for RST5.5 and RST7.5.
  - In order to do this correctly, we need to use the RIM instruction to find the current settings of the RST5.5 and RST7.5 masks.
  - Then we can use the SIM instruction to set the masks using this information.
  - Given that both RIM and SIM use the Accumulator, we can use some logical operations to masks the un-needed values returned by RIM and turn them into the values needed by SIM.

Using RIM and SIM to set Individual Masks

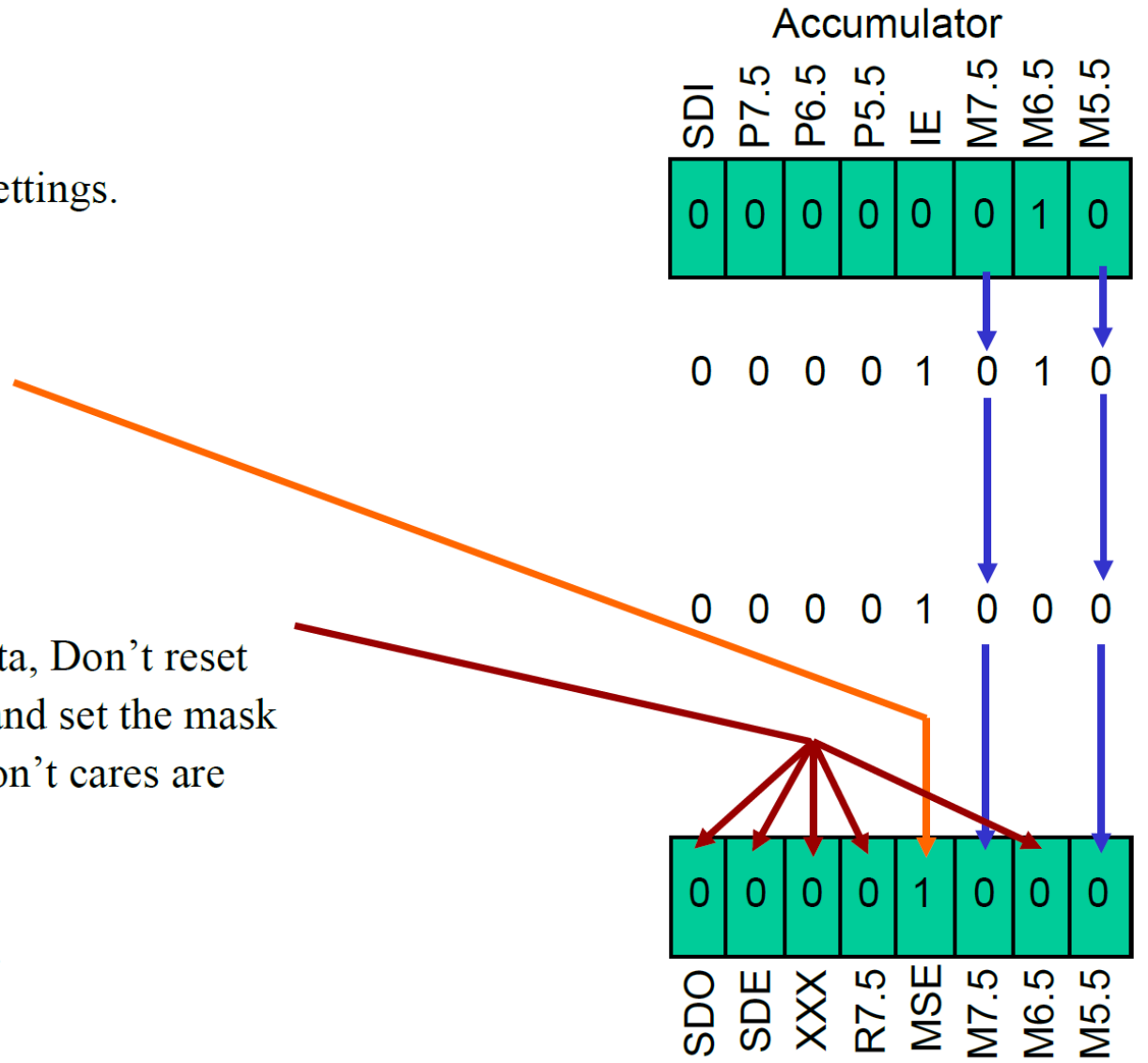
– Assume the RST5.5 and RST7.5 are enabled and the interrupt process is disabled.

```
RIM          ; Read the current settings.

ORI 08H      ; 0 0 0 0 1 0 0 0
             ; Set bit 4 for MSE.

ANI 0DH      ; 0 0 0 0 1 1 0 1
             ; Turn off Serial Data, Don't reset
             ; RST7.5 flip flop, and set the mask
             ; for RST6.5 off. Don't cares are
             ; assumed to be 0.

SIM          ; Apply the settings.
```



TRAP

- TRAP is the only non-maskable interrupt.
  - It does not need to be enabled because it cannot be disabled.
- It has the highest priority amongst interrupts.
- It is edge and level sensitive.
  - It needs to be high and stay high to be recognized.
  - Once it is recognized, it won't be recognized again until it goes low, then high again.

- TRAP is usually used for power failure and emergency shutoff.

Internal Interrupt Priority

- Internally, the 8085 implements an interrupt priority scheme.
  - The interrupts are ordered as follows:
    - TRAP
    - RST 7.5
    - RST 6.5
    - RST 5.5
    - INTR
- However, TRAP has lower priority than the HLD signal used for DMA.

The 8085 Interrupts

Interrupt Name	Maskable	Masking Method	Vectored	Memory	Triggering Method
INTR	Yes	DI / EI	No	No	Level Sensitive
RST 5.5 / RST 6.5	Yes	DI / EI SIM	Yes	No	Level Sensitive
RST 7.5	Yes	DI / EI SIM	Yes	Yes	Edge Sensitive
TRAP	No	None	Yes	No	Level & Edge Sensitive