

Digital Circuits
Prof. Santanu Chattopadhyay
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture – 59
8085 Microprocessor (Contd.)

Next, we will look into the 8085 in the maskable or vectored interrupts.

(Refer Slide Time: 00:22)

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:

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

– 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).

Handwritten notes on the slide: $5.5 \times 8 = 42$, $6.5 \times 8 = 52$, and $2CH$. A diagram shows the 8085 processor with three interrupt pins labeled RST 5.5, RST 6.5, and RST 7.5.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

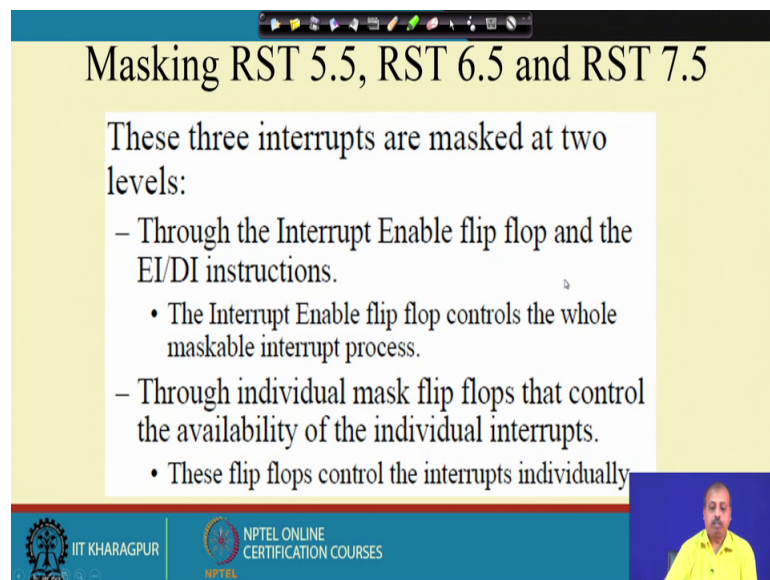
So, 8085 has got 4 masked or vectored interrupt inputs – RST 5.5, 6.5 and 7.5 and there is one non maskable interrupts so, which is which is actually the trap instruction the trap interrupt. So, we will come to that later. Out of them this RST 5.5, 6.5, 7.5 so, they are all maskable interrupts and they are automatically vectored according to this particular table.

So, this 5.5 this is a pin; so unlike that RST instruction that we were discussing that INTR so, that we are having generating the RST instruction. So, if you look into this 8085 processor then you will find that apart from this INTR pin for generating for giving the interrupt so, we have got three more pins which are RST 5.5. They are named as RST 5.5, RST 6.5 and RST 7.5. So, these are the names of the pins and this name also tells the vector address.

So, like any RST instruction execution what happens is that the number is multiplied by 8. So, 5.5 into 8, so, that gives me 42, ok. So, that is in hexadecimal so, that is equal to 2C hex. Similarly, 6.5 is 6.5 into 8 so, that is 50 sorry that is 52 and 52 is again 34 hex. So, this way we have got this RST pins that are for they are also their addresses are also computed by multiplying this value of this 5.5, 6.5 etcetera by 8 and whatever value you get so, that becomes the interrupt address, ok, the ISR address that is why this is called vectored interrupts. So, unlike INTR where the device was providing the ISR address so, here it is not there. So, here it is hardcoded. So, if the interrupt comes in any of these lines then automatically the processor will branch to the corresponding address 2C, 34 or 3C.

Next so, this vector so, they are automatically vectored according to the this according to this table and the vectors for these interrupt fall in between the vectors for the RST instruction and that is why they are named like RST 5.5, so 5 and a half. So, therefore this 5.5 falls between RST 5 and RST 6 and if you look into the address; so, you can understand that this 2C is between the addresses for 5 into 8 and 6 into 8, between those two so, this address comes, that is why the names are like that.

(Refer Slide Time: 03:17)



The slide features a title bar with navigation icons. The main content is on a yellow background with a white text box. At the bottom, there is a blue footer with logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, and a small video inset of a man in a yellow shirt.

Masking RST 5.5, RST 6.5 and RST 7.5

These three interrupts are masked at two levels:

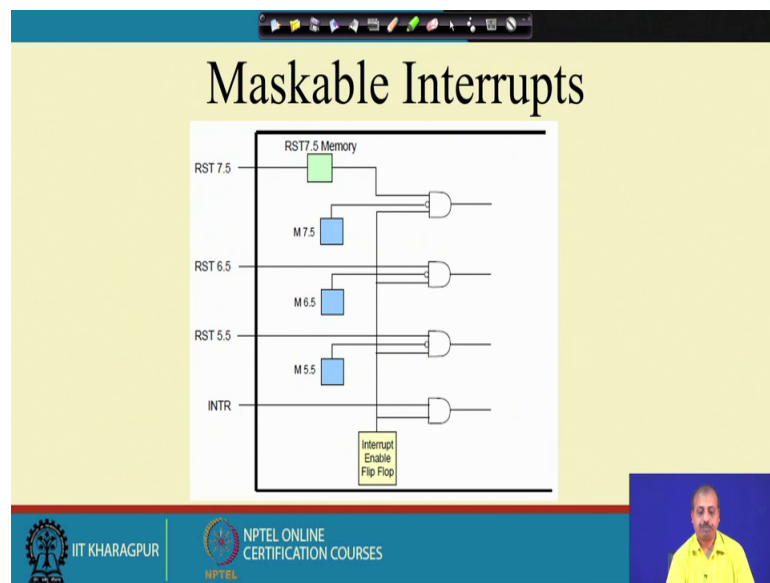
- Through the Interrupt Enable flip flop and the EI/DI instructions.
 - The Interrupt Enable flip flop controls the whole maskable interrupt process.
- Through individual mask flip flops that control the availability of the individual interrupts.
 - These flip flops control the interrupts individually

Now, these three interrupts are masked at two levels, so that this masking is a very important thing because sometimes we want that the processor should not be interrupted. So, we want to mask out this interrupts. So, we do not want the interrupts to reach the

processor, then this masking can be done at two different levels; the first level of masking is via the interrupt enable flip flop and the EI DI instructions. So, in the 8085 microprocessor that is one interrupt enable flip flop called IE and this IE flip flop can be set or reset by using this EI and DI instructions.

So, this call so if the IE instruction is executed then this interrupt enable flip flop is equal to one and now, all the none of the interrupts are masked and then there if it is 0 then it is allowed to give the interrupts. So, this is one type of masking, then other type of masking that we have is the individual mask. There are individual mask flip flops that can control availability of individual interrupts at individual mask interrupt level also you can control the um this interrupts, ok. So, if there are flip flops dedicated for individual interrupts.

(Refer Slide Time: 04:32)



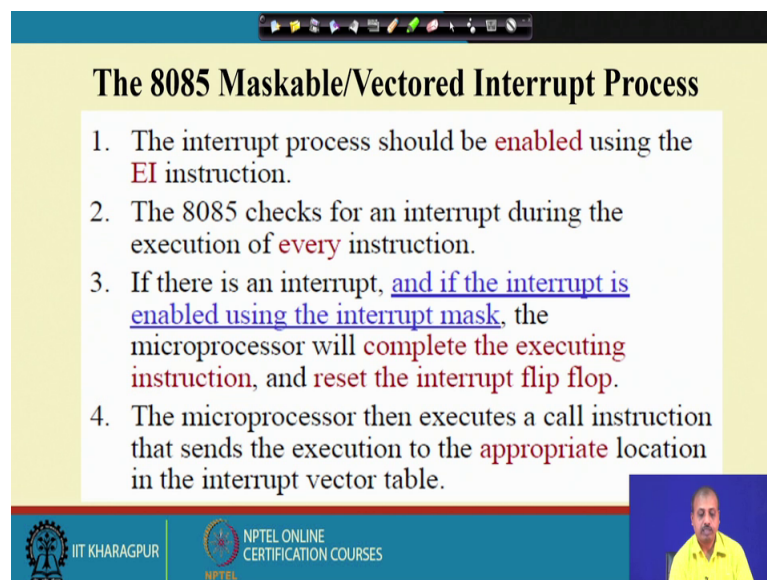
So, this is the conceptually we can think of it like this as if this is finally, the interrupt lines going to the processor. So, if you say that these are going to the actual processor then this interrupt enable flip flop, so, that bit is ended with ended here. So, if this interrupt enable flip flop is 0, you see that none of the interrupts will be made equal to 1. So, all of them will be equal to 0. So, as if it is masked off and then this individually you can control then RST 7.5. So, you can, this is so, will come to this point later.

Now, this if you want to mask out this flip flop this interrupts individually instead of through this interrupt enable flip flop so, you can set this mask bit M 7.5, M 6.5, or M

5.5. So, if I set this bit to 1 and the rest of the bits to 0 then this set to 1 so, this will make this AND gate output 0. Similarly, if this bit is 0, then this is so, this inverted one comes as once it does not have any control over the output of the AND gate. So, it will be determined by other inputs. So, you see that by putting a 1 here so, we can mask out the corresponding interrupt, ok. So, that is one type of masking.

And, this RST 7.5 it has got a special memory. So, it can remember whether one interrupt had occurred, maybe the interrupt is masked out. So, it does not reach the processor at present, but the mask with the memory bit of RST 7.5 will be set. So, later on the user program so, it can check that whether any interrupt had occurred when it was masked off. So, that for that purpose so, there is a memory bit for RST 7.5.

(Refer Slide Time: 06:27)



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.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, how it work? it is like this. The interrupt process should be enabled using the EI instruction so that is true for every interrupt to reach the processor and as I said that 8085 checks for an interrupt during the execution of every instruction. If there is an interrupt and if interrupt is enabled using the interrupt mask the microprocessor will complete the executing instruction and reset the interrupt enable flip flop. So, this is the extra thing that is added here. So, previously we did not note this particular part. So, if the interrupt enabled interrupt is enabled using the interrupt mask. So, it is either by setting those mask bits or the interrupt enable flip flop.

So, if the interrupt is allowed if the interrupt is not masked out then the after completing the current execute current instruction. The processor will reset the interrupt flip flop that interrupt and then it will branch to the appropriate location in the interrupt vector table. So, processor will then execute a call instruction that sends the execution to the appropriate location in the interrupt vector table. So, that is how this maskable vectored interrupt process continues.

(Refer Slide Time: 07:43)

The 8085 Maskable/Vectored Interrupt Process

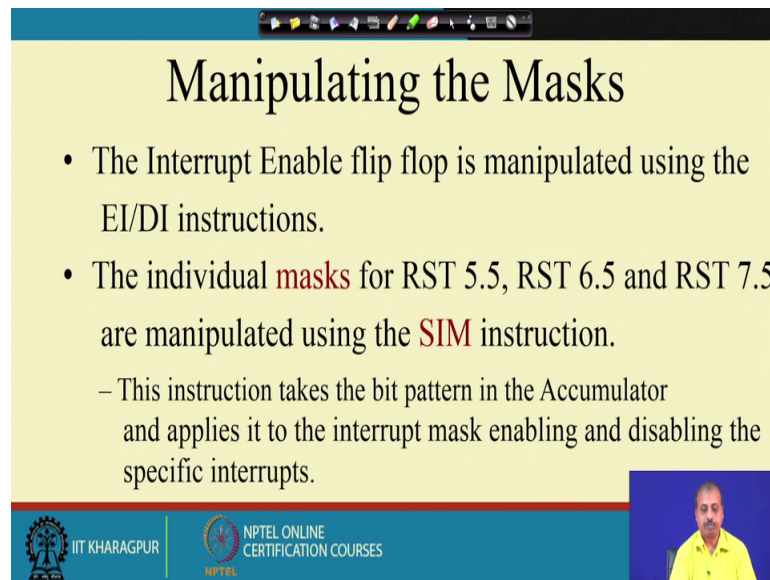
5. When the microprocessor executes the call instruction, it **saves the address of the next instruction** on the stack.
6. The microprocessor **jumps to the specific service routine**.
7. The service routine must include the instruction **EI** to re-enable the interrupt process.
8. At the end of the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

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

And, similarly, when the microprocessor executes the call instruction it saves the address of for the next instruction onto the stack, the return address. The processor will jump to the specific service routine given by the RST instruction that it has got and then or the RST pin on which the interrupt had come the service routine must include the instruction EI so that this interrupt is re enabled. So, other otherwise the interrupt will remain disabled so, it will not be sensed further.

Then, at the end of the service routine the return instruction the rate it will return the execution to where the program was interrupted. So, this is the whole sequence that occurs in case of this mask interrupts.

(Refer Slide Time: 08:29)



The slide is titled "Manipulating the Masks" and contains the following text:

- 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.

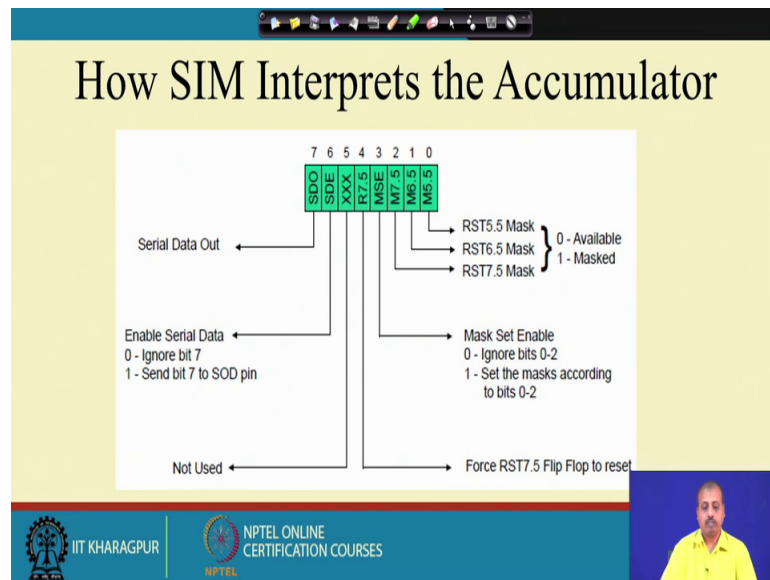
The slide footer includes the IIT KHARAGPUR logo, the NPTEL ONLINE CERTIFICATION COURSES logo, and a small video inset of a man in a yellow shirt.

Now, the question is how can we manipulate the masks. So, this mask bits so, they are user programmable. So, as a so you can decide like which interrupts are useful for you and which interrupts you do not want to disturb the program. So, that way you can put this mask at a different points. So, this interrupt enable flip flop is manipulated by this EI and DI instructions. So, if you set if you execute an EI instruction then the flip flop is set to 1, if you set or if you execute a DI instruction then that flip flop interrupt enable flip flop will be set to 0. So, that is one level.

Now, for this individual masking of RST 5.5, 6.5 and 7.5 so, there is a special instruction called SIM. So, it is known as Set Interrupt Mask or SIM instruction this instruction it will take a bit pattern in the accumulator and applies it to the interrupt mask enabling and disabling the specific interrupts. So, this SIM instruction actually it is a multipurpose instruction. So, it will be reading the accumulator and accordingly it will set some bits and the some of the bits are used for masking out this interrupt masks or the interrupt lines and some bits are used for serial data input output.

So, we will come to the serial data input output part later, but the part that is relevant for us now, is that some of the bits that they will be determining the masking pattern of the interrupts.

(Refer Slide Time: 10:07)



So, this is the situation. So, we have got this, this is how this SIM instruction will understand the meaning of the accumulator. So, accumulator is 8 bit, out of that this bit number 7 and 5 so 7 and 6 so, they are actually for serial transfer of data. So, we will not discuss about them now. Then this bit number 5 is not used, then rest of the bits so, this RST 5.5, 6.5 and 7.5 masks so, bit 0, 1 and 2 so, if you want to set the mask then you have to put a 1, if you want to make them available then you have to put a 0 there.

And, then this MSE is the set mask set enable. So, whether you really want to set the mask or not. Actually the problem is that there are two things that are done by the same instruction one is the serial data input output and another is the setting of this masks. Now, in for serial data input output so, we will we may not be interested to change the interrupt setting for the current program. So, in those cases what we can do? We can just make this masks set enable bit 0. So, that whatever be the values of this M 5.5, 6.5, and 7.5 they will simply be ignored and the interrupt masking pattern will not change.

However, when you are using this SIM instruction for setting the interrupt mask so, you should set this MSE bit to 1, so that this pattern in set in M 5.5, 6.5 and 7.5 so, they are used for actually setting the masks for the interrupts. And, then we have got this R 7.5 bit that is bit number 4 it forces the RST 7.5 flip flop to reset. So, basically you may do this thing like without servicing the interrupts you just want to reset the interrupt flip flop. So, that can be done by setting this RST 7.5 R 7.5 bit to 1, ok.

So, this is you have to set a particular bit pattern in the accumulator and then execute the SIM instruction, so that the interrupt masking will be done.

(Refer Slide Time: 12:27)

The slide is titled "SIM and the Interrupt Mask". It contains the following text:

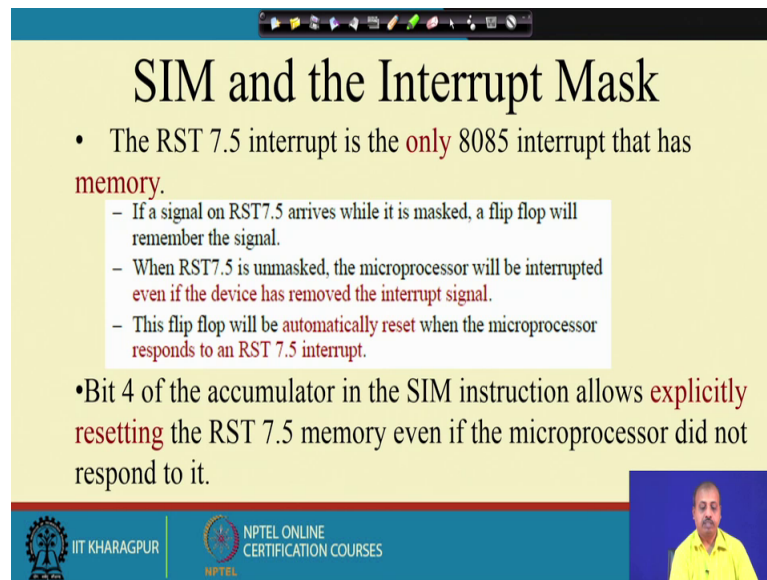
- Bit 0 is the **mask** for RST 5.5, bit 1 is the **mask** for RST 6.5 and bit 2 is the **mask** for RST 7.5.
 - If the mask bit is 0, the interrupt is **available**.
 - If the mask bit is 1, the interrupt is **masked**.
- Bit 3 (Mask Set Enable - MSE) is an **enable for setting the mask**.
 - If it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

The slide footer includes the IIT KHARAGPUR logo and the NPTEL ONLINE CERTIFICATION COURSES logo. A small video inset shows a man in a yellow shirt.

Now, so, as it is said that bit 0 is the mask for the RST 5.5, 1 is for the mask 6.5 and 2 is for the mask is the mask for RST 7.5. If the mask bit is 0 interrupt is available, the mask bit is 1 the interrupt is masked. Bit 3 mask set enable is an enable for setting the mask. If it is set to 0, the mask is ignored and the old settings remain that is so, if the particularly useful for this serial data transfer type of cases and if it is set to 1, the new setting will be applied.

The SIM instruction is used for multiple purposes and not only for setting interrupt mask, it is also used to control functionality for serial data transmission. That is why the bit 3 is necessary. So, as we have discussed previously that we want to do a serial transmission without affecting the interrupt flag. So, that is done by setting this bit 3.

(Refer Slide Time: 13:23)



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.
- 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.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

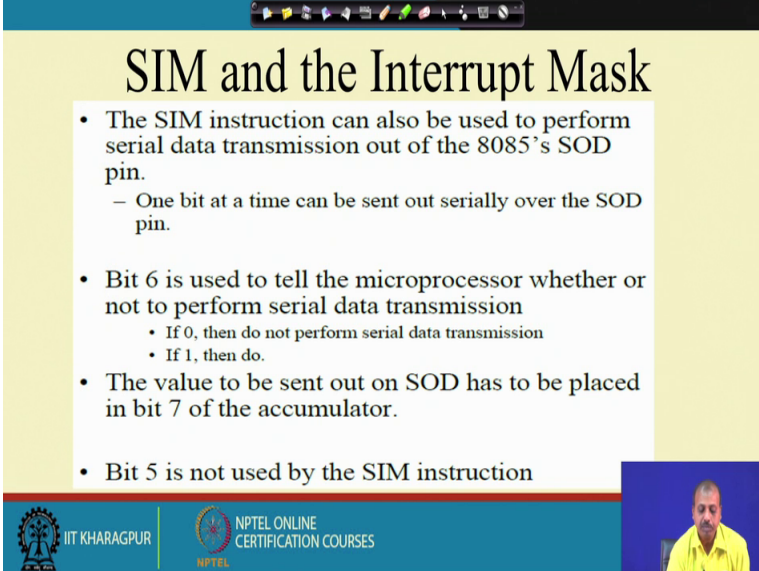
Now, this RST 7.5 interrupt is the only 8085 interrupt that has a memory. So, as we said that there is there was an R 7.5 flip flop that is available in inside the processor. So, even if the some interrupt had occurred and you did not respond to it so this bit, will maybe the 7.5 was masked off, so that that is why it was not sensed, but this particular flip flop remain set. So, you may want to reset the flip flop. So, if a signal on RST 7.5 arrives while it is masked, a flip flop will remember thus there is a signal that something had occurred.

When RST 7.5 is unmasked the processor will be interrupted even if the device has removed the interrupt signal. So, actually this is a memory so it remembers that something some interrupt had occurred and maybe that the device that interrupted so, um it has it has taken off the interrupt line, ok. So, it is not giving the interrupt anymore, but still since this flip flop is set. So, when the processor unmask the 7.5 interrupt then this flip flop will be sending an interrupt to the processor, telling that something telling that in the previously something had happened on that particular line. So, this flip flop will automatically reset when the microprocessor responds to a RST 7.5 interrupts. So, on getting or servicing RST 7.5 this interrupt is cleared.

However, in some cases we may not be interested to service this RST 7.5 anymore because the device that had requested for the interrupt has already removed the request, maybe it is no more interested in the interrupt service routine. So, what happens is that in

that case. So, we can use again the SIM instruction to reset this RST 7.5 memory. So, microprocessor did not respond to the interrupt service routine, but we just revoke this interrupt pin or this internet flip flop by resetting this bit explicitly.

(Refer Slide Time: 15:32)



SIM and the Interrupt Mask

- 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.
- 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

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

The SIM instruction can also be used to perform serial data transmission over the 8085 serial output data pin. So, one bit at a time can be sent serially over the SOD pin and bit 6 is used to tell the microprocessor whether or to or not to perform the serial data transmission that is that SDE serial data enable so, that tells whether it should do the transmission or not.

So, if 0 then do not perform serial data transmission and if 1, then do the transmission. The value to be sent on the sent out on the serial output data has to be placed in bit 7 of the accumulator. So, whatever bit you want to transmit. So, that should be placed on the bit 7, so that when you execute the SIM instruction it comes to the serial data output of that of the register and then from there it is outputted to the SOD pin. And, bit 5 is not used by the SIM instruction. So, it is not a meaningful for the SIM.

(Refer Slide Time: 16:33)

Using SIM Instruction to Modify 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	SDO	0
- Disable 6.5	bit 1 = 1	SDE	0
- Enable 7.5	bit 2 = 0	XXX	0
- Allow setting the masks	bit 3 = 1	R7.5	0
- Don't reset the flip flop	bit 4 = 0	MSE	1
- Bit 5 is not used	bit 5 = 0	M7.5	0
- Don't use serial data	bit 6 = 0	M6.5	1
- Serial data is ignored	bit 7 = 0	M5.5	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

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Let us look into some example to set the interrupt masks so that RST 5.5 is enabled, 6.5 is masked and RST 7.5 is also enabled. So, if we want to do that first we have to determine what should be the content of the accumulator. So, first enable 5.5; enable 5.5 means I have to put this bit 0 to 0. So, it is not masked, then disable 7.5 as disable 6.5 so, it is masked, so I have to set this mask to 1. So, bit 1 should be equal to 1 and then RST 7.5 be enabled. So, this bit 2 is equal to 0.

Now, bit 3, so, since we are looking for setting this particular mask pattern for the interrupt, so this MSE bit should be equal to 1. So, bit 3 is equal to 1. Bit 4 was for R7.5, so, that is not reset. So, we do not use this RST 7.5 reset, so do not reset the flip flop. Bit 5 is not used. So, the bit 6 is for serial data output. So, we are not interested in serial data output now. So, this SDE serial data enable is made 0 and SDO is it can be any value so, this is just put a 0 there.

So, the instruction sequence for setting this mask and interrupting format is like this first we execute the EI instruction to enable the interrupt in interrupt enable flip flop and then MVI A, 0A hex. So, if you look into this pattern so, this is 0. And this is for next four bits constitute the hexadecimal number A, so, this is 0A. So, MVI A, 0A hex so, that will put this particular bit pattern onto the A register and then the SIM instruction is executed. So, this bits are copied and accordingly the corresponding the masking pattern is set for the

interrupts and this serial data that is disabled. So, this way we can use this SIM instruction for setting the interrupt masks.

(Refer Slide Time: 18:48)

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.

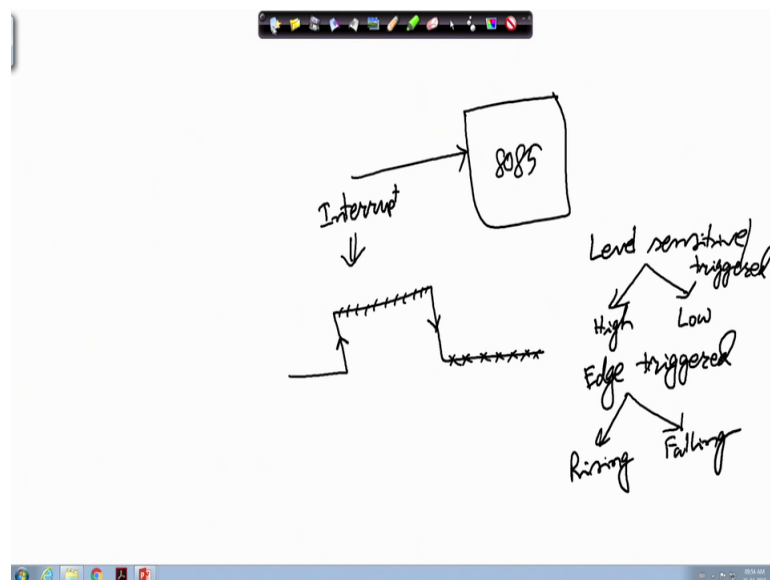
RST 6.5 and RST 5.5 are **level sensitive**.

- The interrupting signal **must remain present** until the microprocessor checks for interrupts.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Another important thing that we have is about that triggering. So, this RST 7.5 is a positive edge sensitive.

(Refer Slide Time: 19:08)



So, triggering actually if we so, it is like this that so, this is the microprocessor 8085, now if this is some interrupt line that is coming it can be INTR or any of those interrupt. So, if an interrupt is coming so, that I can have several types of situation; like this

interrupt signal that is maybe it is coming as a pulse, ok. Now this interrupt may be sensed interrupt may be accepted by the processor if it is taken high. So, it is called level sensitive, it is level sensitive. So, level sensitive can also be of two types level high and level low.

So, if we say that 8085 will sense this interrupt if the line if the interrupt line is at is at high level. So, that is high level sensitive interrupt. On the other hand so, if we say that this will sensed when the signal value is low, ok, so, that is the low level sensitive interrupt. So, basically I said that it will check for the interrupt on the last, but one clock cycle of every instruction execution; so, at that time if for a high level sensitive interrupt so, if it finds that the interrupt line is high then it will be sense it will be taking that an interrupt has occurred. On the other hand a low level sensitive interrupt means that if the if it finds that the signal value is low, then it will be getting that low level sensitive it will be getting the interrupt.

Now, apart from that another type of classification of interrupt can be the edge sensitive interrupts or edge triggered interrupt. So, this is called level sensitive or level triggered and we can also have this edge triggering, just like in flip flops we have got level triggered and edge triggered so, here also the similar concept is there. So, it will expect that on the interrupt line so, there is a Rising edge like this so, that will be taken as edge triggering that is that is Rising edge this is the Rising or Falling. So, this is Rising edge triggering and other possibility is it will get an interrupt on this falling edge. So, this is falling edge triggering.

So, using this technique using this or these are the possible alternatives that we can have. So, different processors will have different types of interrupting mechanisms. So, will see what happens in 8085 like; so, what are the interrupt mechanism that we have in 8085. So, out of the different um different interrupt mechanism things that we have so, this RST 7.5 is a positive edge sensitive. So, it will expect that when the positive edge occurs that is going that the line goes from low to high then a logic 1 is stored in the flip flop as a pending interrupt. So, out of this RST 7.5, 6.5 and 5.5, 7.5 has got a flip flop in it. So, that flip flop will be sensing the value. So, this is actually keeping the edge within it.

So, then this is 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. So, that is the beauty. So, if your device cannot keep the signal high for a long time. So, we should connect it through RST 7.5 pin because we know that there a positive edge will be sufficient for getting the interrupt administered or sensed by the processor, because it will be setting that R 7.5 interrupt line interrupt flip flop and then it will be the sensed later by the processor when you complete the current instruction.

Now, the line must go to 0 and back to one before a new interrupt is recognized. So, this is another point. So, since this is a all this is a positive edge sensitive. So, whenever the positive edge comes then only it is sensed. So, if it remains high for quite some time it will not be sensed as a new interrupt, because that that positive edge is required. On the other hand this 6.5 and 5.5 they are level sensitive interrupts and as their level sensitive the interrupting signal must remain present until the microprocessor checks for the interrupt. So, previously we have seen that the minimum duration for which the signal should be kept high is 17.5 clock cycles. So, this RST 6.5 and 5.5 these signals they should be kept high for that 17.5 clock cycles at least, for getting it sensed by the processor.

(Refer Slide Time: 24:02)

Determining the Current Mask Settings

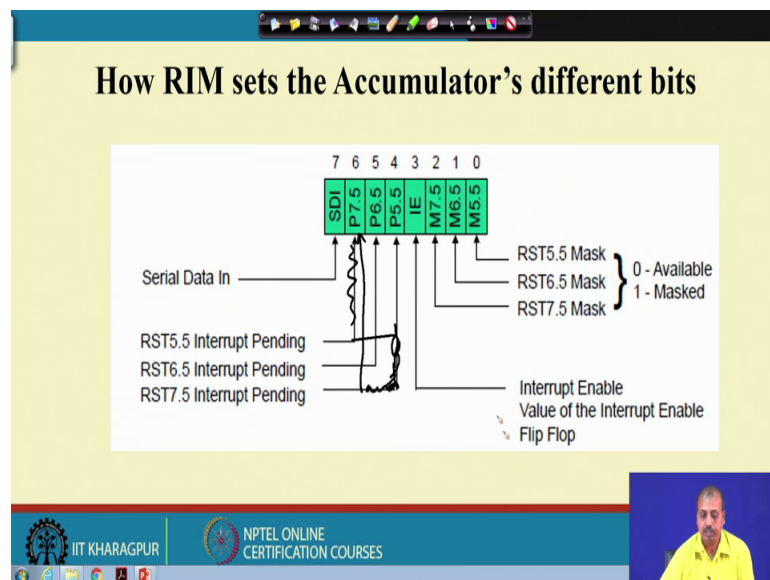
- RIM instruction: Read Interrupt Mask
 - Load the **accumulator** with an 8-bit pattern showing the status of each interrupt pin and mask.

So, you can also determine the current mask settings like a using the RIM or RIM instruction which is read instruction mask sorry read Interrupt Mask, ok. So, what

happens is this in this case is that the accumulator will be loaded with the 8-bit pattern for different positions. So, this RST 7 from the RST 7.5 memory so, it will come to bit 6 which we call present P7.5 that is the present value of 7.5. Then, this M7.5 mask bit so, this is copied into P6.5 bit number 5 of the accumulator, then this P M 6.5 will be copied into M5.5 sorry M6.5 is copied onto this M6.5 bit, then this M5.5 is copied into this M5.5 bit. And, then we have got this P6.5. So, this is the pin value, this is not this is not present this is the pin value. So, pin value is copied into this P bits and the mask settings are copied onto the M bits, ok. So, this pin values are copied onto P bits and mask values are copied onto M bits. Similarly, the interrupt enable flip flop is copied onto this IE bit, ok.

So, in this way if you execute a RIM instruction you get the status of this mask bits this R 7.5 memory flip flop also the pin bits also the pins, because there is no point getting the pin of RST 7.5 because it is controlled by this flip flop. So, that is why it is not kept separately, but for 6.5 and 5.5 the pin values are copied here and stored here. So, this RIM or RIM instruction so, this will be loading the 8-bit pattern showing the status of each interrupt pin and this mask.

(Refer Slide Time: 26:08)

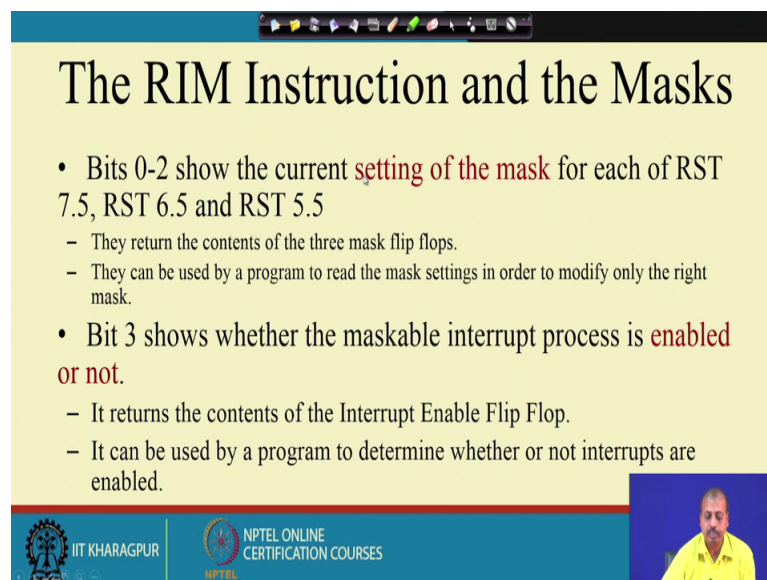


So, we can use this instruction for different purpose so, RIM instruction the other utility is to read the serial data. So, serial data inputs so, that is the other purpose and the other bits bit 6 is for RST 5.5 interrupt pending. So, if there is a pending interrupt from the pin

so, it will be coming. Then, RST 6.5 is the this P6.5 is the RST 6.5 interrupt pending bit and RST this 5.5 actually this diagram it is slightly wrong. So, this is this is not this one. So, this is this and this is actually this one ok.

So, we have got this pending bits stored in P7.5, 6.5, 5.5 and we have got the mask bits M5.5, 6.5, 7.5 in this in this side and this interrupt enable value interrupt enable flip flop value is kept in bit number 3, ok. So, this way this RIM instruction it will set the accumulator to different bit values. So, we can use this SIM and RIM instructions to set this interrupt patterns in a to a particular value.

(Refer Slide Time: 27:38)



The slide is titled "The RIM Instruction and the Masks". It contains the following text:

- 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 slide also features logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, and a small video inset of a man in a yellow shirt.

Now, so, this RIM instruction bits 0 to 0 to 2; 0, 1 and 2 they show the current setting of the mask for each RST 7.5, 6.5 and 5.5 pin. The return the contents of the three mask flip flops and they can be used by a program to read masks settings in order to modify only one only the right mask. So, if you want to modify it so, you can do that.

Bit 3 it shows whether the maskable interrupt process is enabled or not and it returns the contents of the interrupt enabled flip flop, ok. It can be used by program to determine whether or not the interrupts are enabled. So, to determine that so basically this RIM instruction is for getting the status of the interrupt occurrence in the processor, whether they are enabled, whether this masks are set and what are if something is pending and all any interrupt is pending. So, it is getting those information.