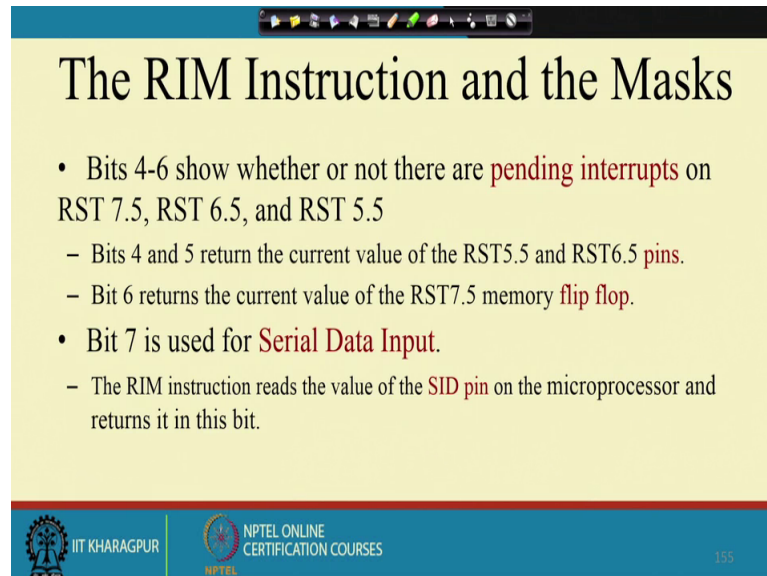




**Lecture - 60**  
**8085 Microprocessor (Contd.)**

(Refer Slide Time: 00:19)



**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.

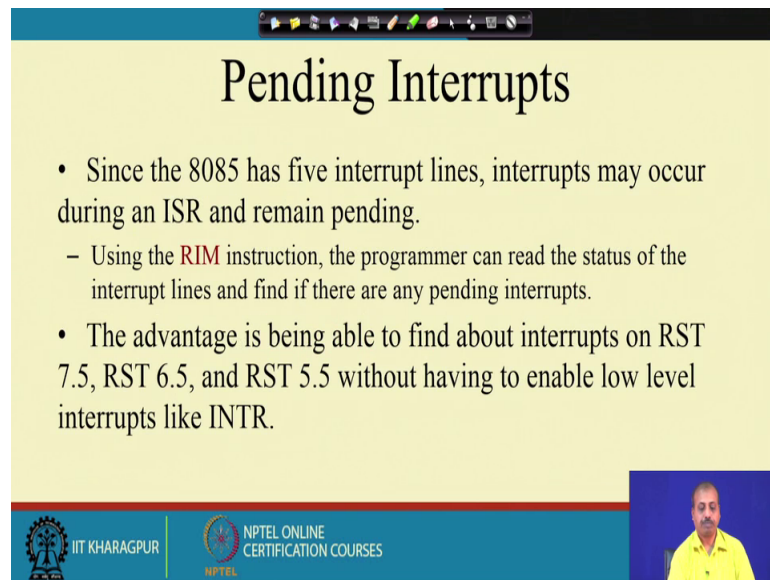
 IIT KHARAGPUR |  NPTEL ONLINE CERTIFICATION COURSES

155

This for the RIM instruction; so, other bits like bits 4 to 6 show whether or not there are pending interrupts on RST 7.5, 6.5 and 5.5 line. So, bits 4 and 5, they return the current value of RST 5.5 and 6.5 pins bit, 6 returns the current value of RST 7.5 memory flip flop. So, you note here that for bits RST 7.5, we do not read the pin value because RST 7.5 is an S triggered flip flop. So, if it may be that that edge is already done; so, if you read the pin so it does not have any meaning and the pin that the edge is already registered in the R 7.5 memory bit RST 7.5 memory bit. So, that is already there. So, it is better to copy it from there.

So, this bits 4 to 6. So, they are holding the pending interrupt status and bit seven is for the serial data input. So, the RIM instruction it reads the value of the SID pin on the microprocessor and returns it in this particular bit. So, after executing the RIM instruction the SID pin whatever be the value. So, that will come to the most significant bit of the accumulator and then accumulator. So, based on that; so, we can we can read, we can read the value through the SI of the SID pin into the accumulator bit 7.

(Refer Slide Time: 01:40)



The slide is titled "Pending Interrupts" and contains the following text:

- Since the 8085 has five interrupt lines, interrupts may occur during an ISR and remain pending.
  - Using the **RIM** instruction, the programmer can read the status of the interrupt lines and find if there are any pending interrupts.
- The advantage is being able to find about interrupts on RST 7.5, RST 6.5, and RST 5.5 without having to enable low level interrupts like INTR.

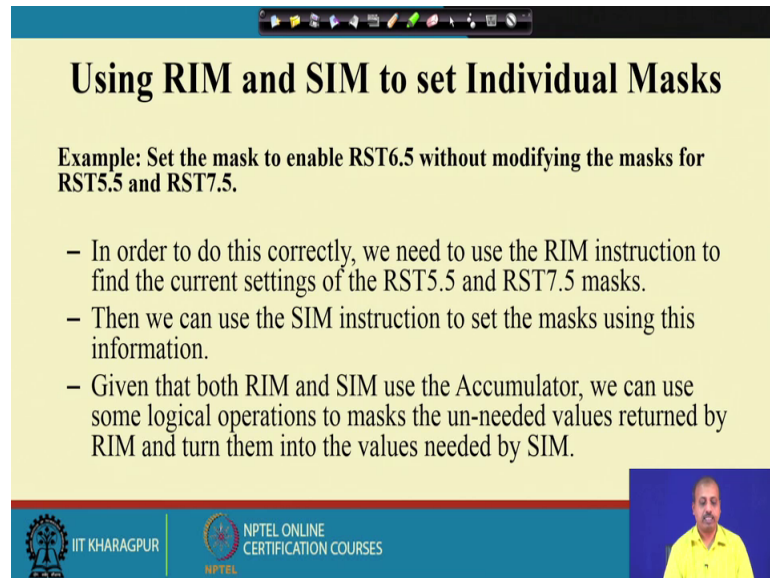
The slide footer includes the IIT Kharagpur logo, the NPTEL Online Certification Courses logo, and a small video inset of a man in a yellow shirt.

Now the pending interrupts that we are talking about say 8085 has got 5 interrupt lines and interrupts may occur during an ISR and remain pending. So, this can happen. So, using the RIM instruction the programmer can read the status of the interrupt lines and find if there are any pending interrupts. So, this is the utility of this of this RIM instruction the advantage is being able to find about interrupts on RST 7.5, 6.5 and 5.5 without having to enable low level interrupt like INTR.

So, many a time, what happens is that maybe in my system RST 7.5 and 6.5, they are very important devices and 5.5 INTR, the devices connected to those lines are not that much interrupt not that much important. So, maybe I was servicing RST 7.5 and in between some 6.5 had occurred, but 6.5 having lower priority than 7.5. So, it was not the accepted by the processor.

So, this pending bit so at the end of the 7.5 ISR also if I check the pending bit, I will find that RST the 6.5 interrupt had come. So, I can directly branch to the 6.5 interrupt service routine from there and otherwise, what I have to do is I have to enable all the interrupts again and that may create some chaos because now many other interrupts may pending on 5.5 and INTR lines and they will also come into picture ok. So, that has to be avoided.

(Refer Slide Time: 03:15)




**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 mask the un-needed values returned by RIM and turn them into the values needed by SIM.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES



So, for that purpose, this RIM instruction is useful. So, here is an example set the mask to enable RST 6.5 without modifying the masks for RST 5.5 and 7.5. So, what you want is the current setting for 5.5 and 7.5, they should continue only the 6.5 setting should be 6.5 interrupts should be enabled.

So, first of all we have to do a RIM instruction to find the current setting of 5.5 and 7.5 masks, then we can use the SIM instruction to set the masks using this information. So, this RIM and SIM instructions, since they use the accumulator, we can use some logical operations to mask the un needed values ok, the un needed values returned by the RIM and SIM instruction and by this RIM instruction and then turn the turn into the values needed by the SIM instructions.

(Refer Slide Time: 04:10)

### Using RIM and SIM to set Individual Masks

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

Instruction	Comment	Accumulator
RIM	; Read the current settings.	0 0 0 0 1 0 0
ORI 08H	; 0 0 0 0 1 0 0 ; Set bit 4 for MSE.	0 0 0 0 1 0 1
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.	0 0 0 0 1 0 0
SIM	; Apply the settings.	0 0 0 0 1 0 0

Accumulator

SDO SDE R7.5 R6.5 IE M7.5 M6.5 M5.5

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

So, here is the example. So, you see we assume that RST 5.5 and 7.5 are enabled and the interrupt process is disabled. So, as a result since that this IE is 0 and 7.5 and 5.5, they are not masked and this is 6.5 is masked. So, what we do is the first we execute a RIM instruction that gives me the current setting which is like this and then we want to we want to set the interrupt mask in some fashion. So, for that we have to set this we have to prepare for this SIM instruction in the SIM instruction bit number 4 is the MSE Mask Set Enable. So, for whatever value I have got so with that; so, if I do an ORing with this particular pattern ok. So, then this bit becomes equal to 1.

So, the with this. So, if you can do this ORing. So, you get the bit number 4 set whatever be the value that you read here. So, if you do or with this 0 8 hex. So, bit number 4 is set.

Now, what we want is that the serial data should be turned off. So, serial data turn off. So, so this SDE value should be equal to 0 SDO, it is can be 0 or 1, it does not matter; so, it is made 0, then RST 7.5 flip flop. So, that is do not we do not want to reset the RST 7.5 flip flop. So, this bit is set to 0 and this RST 6.5 is turned off. So, the mask and set the mask for RST 6.5 turn off. So, 0.5 turn off is this 1 M 6.5. So, that is made 0 and don't cares for the and this and the mask for do not reset the mask for reset for RST 7.5 and set the mask for 6.5 off. So, mask for 6.5 has been set off and don't cares are assumed to be 0.

So, rest of the things they will taken as 0. So, this way we can get we can apply this particular setting. So, when I do an and immediate with this. So, I will get this particular bit pattern and this bit pattern when it is put into a SIM instruction. So, it will do the desired thing that we want. So, this way, we can use this RIM and SIM instruction along with a few logical operations to set this interrupt line interrupt enable and for selecting some interrupt pins and getting them masked out or enabled so like that.

(Refer Slide Time: 06:50)

**TRAP** ⇒ RST 4.5     $4.5 \times 8 = 36$

- 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.

IIT KHARAGPUR    NPTEL ONLINE CERTIFICATION COURSES

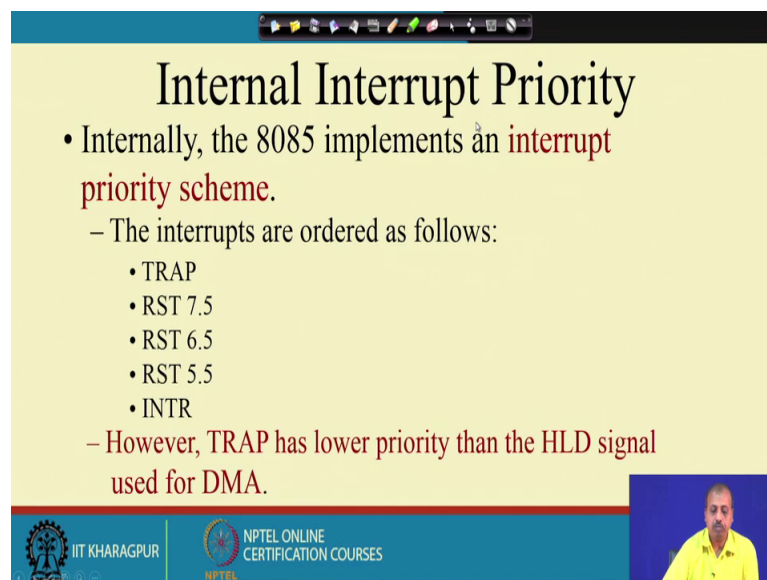
Another very important interrupt that we have not discussed so far is the trap. So, this is the only non maskable interrupt that 8085 has; so, all other interrupts they are maskable, but trap is a non maskable interrupt. So, it does not have any effect the EI and DI instructions, they will not have any effect on the trap line. So, it does not need to be enabled because it cannot be disabled it has the highest priority amongs all the interrupts and in fact, so, this trap is also known as RST 4.5. So, this trap is also the RST 4.5. So, when this trap occurs then the processor will jump to 4.5 into 8 that is location 34. So, sorry location 36 in decimal; so, this will be jumping to location 36 so that is between this RST 4 and 5 so that is why it is called RST 4.5 and the corresponding address is 36 decimal.

So, it has the highest priority among all the interrupts, it is edge and level sensitive. So, it is a combination of this edge sensitive and level sensitive. So, it is edge and level sensitive it needs to be high and stay high to be recognized. So, since it is a very

important interrupt. So, this is often used for power failure or some emergency condition now if there is a glitch that is coming on the power supply line and it is sensed as an interrupt. So, that is that may be undesirable.

So, what is required in case of trap to be sensed is that it should be there should be an edge as well as it should there should be it should be high for quite some time for it to be sensed. So, it is edge and level sensitive, it needs to be high and stay high to be recognized and once it is recognized, it will not be recognized again until it goes low because it because of the nature that it is it is edge and level sensitive. So, that to bring that edge sensitivity into picture. So, once it is sensed it must go low and then only it can come high again.

(Refer Slide Time: 09:09)



**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 slide includes logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, and a small video inset of a presenter in a yellow shirt.

So, sorry; so, next we will be looking into the internal priority, internal priorities that we have o internally 8085 implements an interrupt priority scheme and this interrupts are ordered like this the trap has the highest priority followed by 7.5, 6.5, 5.5 and INTR; however, trap has lower priority than the hold signal that is used for the DMA. So, the DMA will come later. So, this is for Direct Memory Access so for doing some operation data transfer directly between memory and the secondary storage without involving the microprocessor.

So, for that type of operation we need this direct memory access or DMA and this is this hold and hold acknowledge. So, these are the two pins associated with that operation. So,

they are. So, when the hold signal comes then what the processor does is that it releases all the buses; so, it just as if it will not use any external bus. So, it will stop all the operations and the bus is given to some other master. So, that these data transfer to memory can take place directly about this trap has got lower priority than the hold signal.

(Refer Slide Time: 10:21)

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

So, to summarize; so, if you look into different interrupts the maskable; so, this interrupt INTR RST 5.5, 6.5, 7.5, they are all maskable. So, where trap is non maskable masking method INTR is DI EI RST 7.5, 5.5 and 6.5, 7.5. So, DI EI and SIM, trap cannot be masked so there is nothing. So, this INTR is not vectored, but remaining interrupts, they are vectored interrupts because for them, we know what is the address at which the execution the I the interrupt service routine should be located.





And so, whether any of this interrupts have got memory yes RST 5 7.5 have got a memory others do not have. So, others they require that it should be high for that 17.5 T states for getting it sensed in the worst case. Triggering method so we have got INTR level sensitive 5.5, 6.5, they are also level sensitive 7.5 is edge sensitive and trap is level and edge sensitive. So, the edge should be there followed by it should be high for quite some time. So, this way this 8085 interrupts can summarize their different features.

(Refer Slide Time: 11:42)

## Direct Memory Access

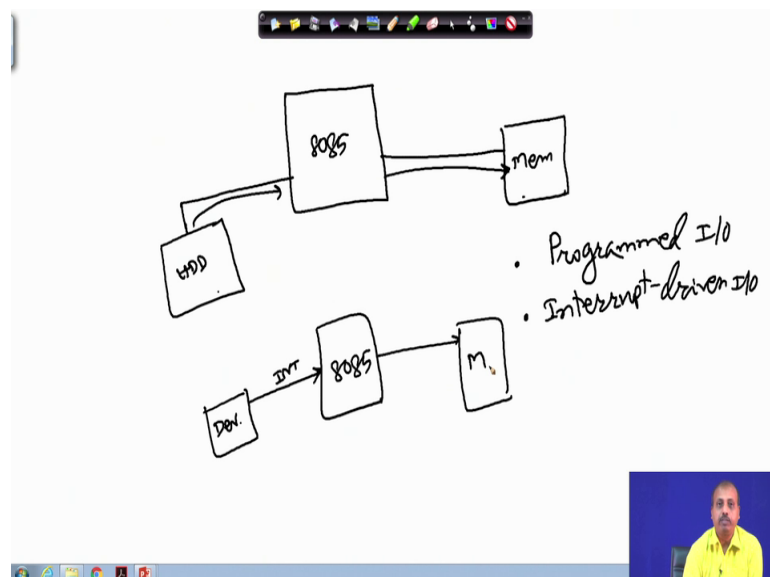
This is a process where data is transferred between two peripherals directly without the involvement of the microprocessor.

- This process employs the HOLD pin on the microprocessor
  - The external DMA controller sends a signal on the HOLD pin to the microprocessor.
  - The microprocessor completes the current operation and sends a signal on HLDA and stops using the buses.
  - Once the DMA controller is done, it turns off the HOLD signal and the microprocessor takes back control of the buses.



Next, we will be looking into another important concept which is known as direct memory access. So, this direct memory access is a process where data is transferred between 2 peripherals directly without involvement of the microprocessor.

(Refer Slide Time: 12:03)



So, it is like this that if we look into this data transfer process between any processors. So, suppose this is our 8085 processor and this is the memory ok. So, you have got your address data bus line connection and this 8085 is connected to some external disc external device which may be a hard disc for example.



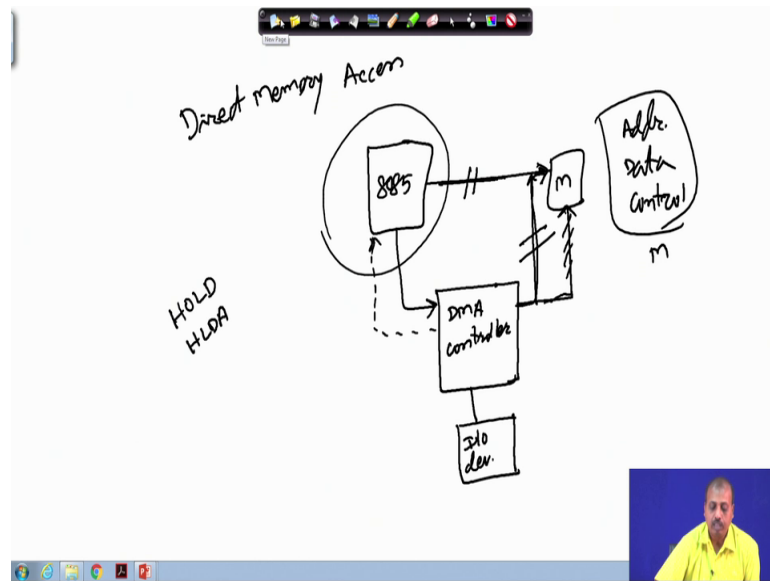
Now, if it is required that some data has to be transferred from this hard disc to memory then what is required is that 8085, if I do it through this 8085 processor, then what it will do it will read values from this hard disc and then write something on to the memory write the value on to the memory, then again, it will read the next byte from the hard disc and it will write the byte on to the memory so it will go like. So, this is this is known as the programmed I O, programmed input output operation.

Difficulty with this programmed input output operation is that this the device the I O device that we are talking about. So, they are generally much slower compared to this processor like 8085. So, that a good amount of time 8085 just waits for the for the device to be ready to transfer the, to give the next byte next byte of data and so that 8085 can read it and transfer. So, that is the problem with the programmed I O.

There is another type of in data transfer which is known as interrupt driven I O. So, interrupt driven I O. So, it is like this that this processor. So, this processor does not tell the device whether you are ready with the data the device rather it gives an interrupt to the processor ok. So, it gives an interrupt to the processor telling that I am ready with the data. So, the 8085 goes into the ISR. So, in the ISR 1 byte of data or some amount of data whatever data this device is ready with are read and they are put into the memory. So, the point is that now this processor otherwise it is not waiting for the device to be ready. So, whenever the device is ready so it will tell the processor by an interrupt and that interrupt will be in the interrupt service routine will be actually the data transfer job that is done.

So, again; so, this is an improvement over the programmed I O, but it is a still not that much fast.

(Refer Slide Time: 14:46)



So, what is done and the third possibility is known as the Direct Memory Access or DMA. So, direct memory access. So, we have got the 8085 processor. So, we have got the memory. So, this memory is connected to the 8085 processor plus there is another device which is known as the DMA controller or direct memory access controller.

So, what this direct memory access controller does is when 8085 tells this direct memory access controller and that I want to get some data transferred. So, this I O device; so, this is connected to the DMA controller. So, this I O device is connected to the DMA controller. So, this 8085 tells the DMA controller that I want some data to be transferred from the I O device to the memory or vice versa from memory to the I O device. So, this DMA controller will do that job. So, it will do transfer the content to the memory ok. So, after; so, this processor is free. So, processor can do anything else it wants to do. So, it is if it definitely it cannot use the memory, but it can do other computations that it within the processor and all.

So, those type of cases operations it can do and the DMA controller will do the transfer and when it is done with the transfer, then it will again tell the processor that I am done ok, I am done with the operation and then this 8; 8085 will restart will continue with the previous operation that.

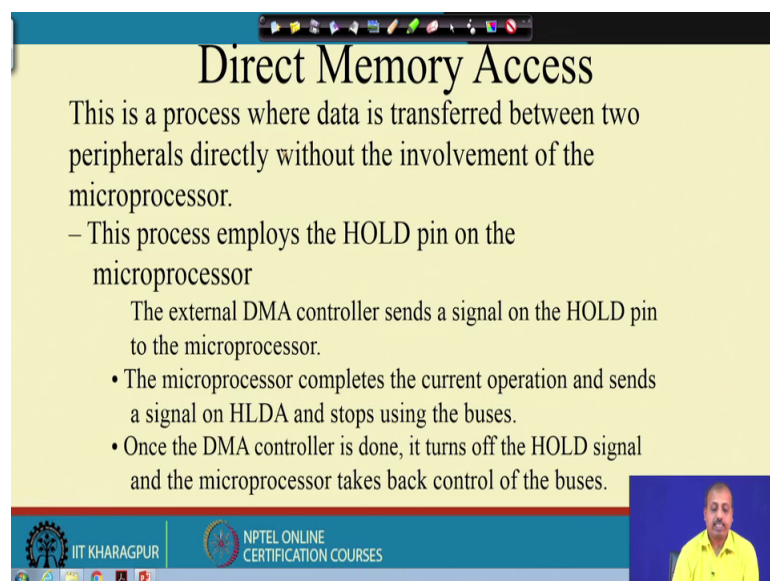
Now, the point that is to be noted is that for reading or writing something to and from memory. So, what we need is that the memories address bus and data bus and control bus. So, they are to be given proper values to the memory.

Now, in this situation what is happening is this is memory has got two drivers, one of this address data control buses are coming from this 8085 and in an another situation it is coming from the DMA controller. So, actually though I have shown it like this, but in reality the connection is like this fine.

So, we have got two drivers for the same bus; so, to solve the problem. So, what is done is that when 8085 tells the DMA controller that it will lead to want some data transfer through DMA mode. So, it will release the control here ok, it will release the control of this bus and now this DMA controller will control this bus and after sometime when the transfer is over when it is informed by the 8085 to the 8085 that it is over then this is this control is taken off and this control is re established.

So, that way this whole DMA operation works and you remember that there were two pins hold and hold acknowledge, 2 pins of 8085. So, they were actually responsible for this DMA type of operation. So, we will see how this hold acknowledge pins, they are going to be useful for this transfer.

(Refer Slide Time: 18:02)



**Direct Memory Access**

This is a process where data is transferred between two peripherals directly without the involvement of the microprocessor.

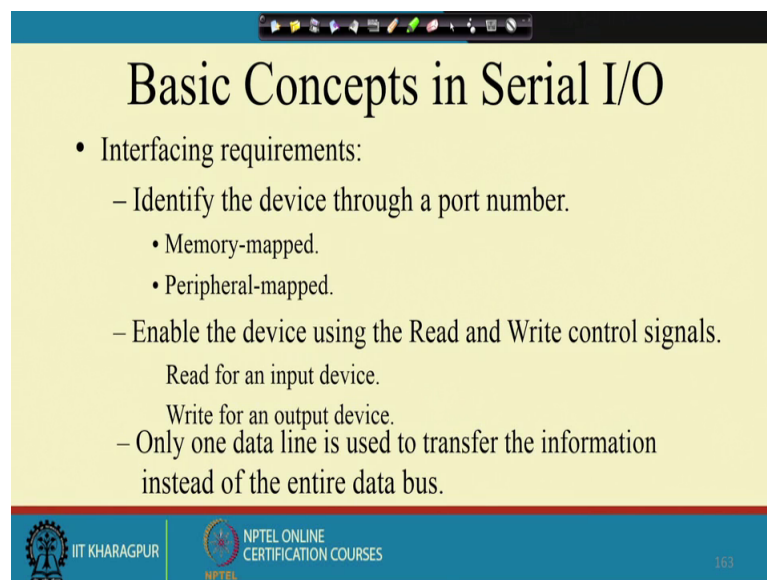
- This process employs the HOLD pin on the microprocessor
  - The external DMA controller sends a signal on the HOLD pin to the microprocessor.
  - The microprocessor completes the current operation and sends a signal on HLDA and stops using the buses.
  - Once the DMA controller is done, it turns off the HOLD signal and the microprocessor takes back control of the buses.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, this direct memory access is a process where data is transferred between 2 peripherals directly without the involvement of the microprocessor. So, processor is not involved in the in this transfer the process employs the whole pin of the microprocessor the external DMA controller sends a signal to on the hold pin. So, the to the microprocessor that you please release the buses.

The microprocessor completes the current operation and sends a signal on the hold acknowledge and stops using the buses. So, it releases the buses that is what that is what I was saying and once the DMA controller is done so it turns off the hold signal and the microprocessor takes back the control of the buses. So, this is how this whole operation take place by means of this hold signal.

(Refer Slide Time: 18:55)



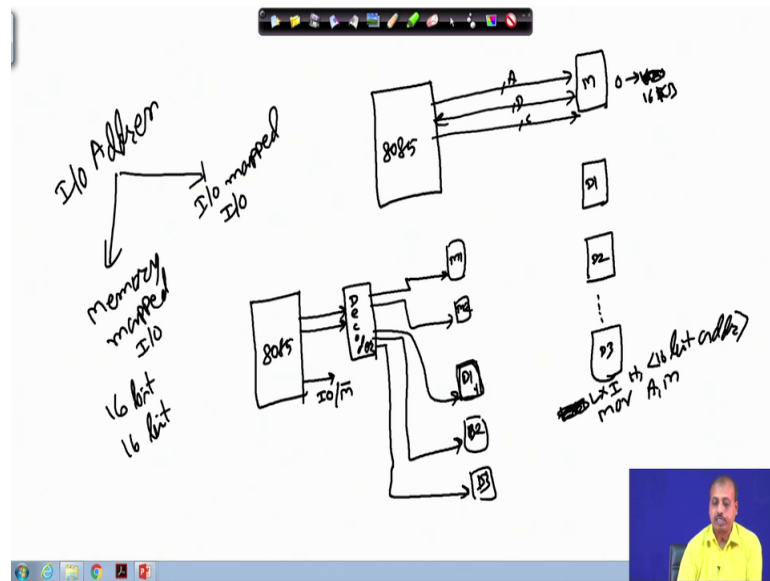
The slide is titled "Basic Concepts in Serial I/O" and lists the following requirements:

- Interfacing requirements:
  - Identify the device through a port number.
    - Memory-mapped.
    - Peripheral-mapped.
  - Enable the device using the Read and Write control signals.
    - Read for an input device.
    - Write for an output device.
  - Only one data line is used to transfer the information instead of the entire data bus.

The slide footer includes the IIT Kharagpur logo, the NPTEL Online Certification Courses logo, and the number 163.

So, next we will be looking into the serial input output operation. Now this serial input output operation means that I want to transfer data serially, but before going to that. So, we want to discuss about the parallel the instead of parallel the serial the parallel I O transfer a bit. So, what happens is that ok. So, this as I said that the programmed I O or interrupt driven I O whatever you call it. So, ultimately the device have the processor has to read from the device or it has to write on to the device. Now, if I have forgot say if this is the 8085 processor.

(Refer Slide Time: 19:37)



And in this 8085 processor so I have got some memory and this memory it ranges over the address is say 0 to say 6 K or say 16 kilobyte 0 to 16 kilobyte. So, this range is this range by the memory by the memory. So, this address data buses are connected there by means of decoder and all. So, this is the address data and control bus connections I am not showing the latches and latches and all demultiplexing the address data bus and all. So, I am not showing it separately assuming that all those are there.

Now, if there are a few devices connected to this processor. So, this is device 1, device 2, device 3 like that now these devices are to be also accessed now this for accessing a particular device. So, the processor needs to tell what is the corresponding address now while telling this address 8085 can do it in 2 different ways the I O address that it is talking about it can be classified into two categories one is called memory mapped address or memory mapped I O another is called I O mapped I O in case of memory mapped I O. So, what happens is that if this is the 8085 processor and these are the memory chips. So, suppose I have got 2 memory chips in my system; M 1 and M 2 and we have got a few devices D 1, D 2, D 3, etcetera.

Now, this address decoder that I have the decoder the address decoder; so, it has got some of the bits of the address lines and it is generating the enable signals for the memory chips. So, some of these lines may be connected to the enable signal of the devices as well, fine. So, if I do this thing then what happens is that there is no difference

between a memory location and a device location; so, as far as the processor is concerned. So, it just needs to know what is the address of this device and that we what is; so, for a memory location the address is 16 bit. So, this device is also nothing, but a 16 bit address.

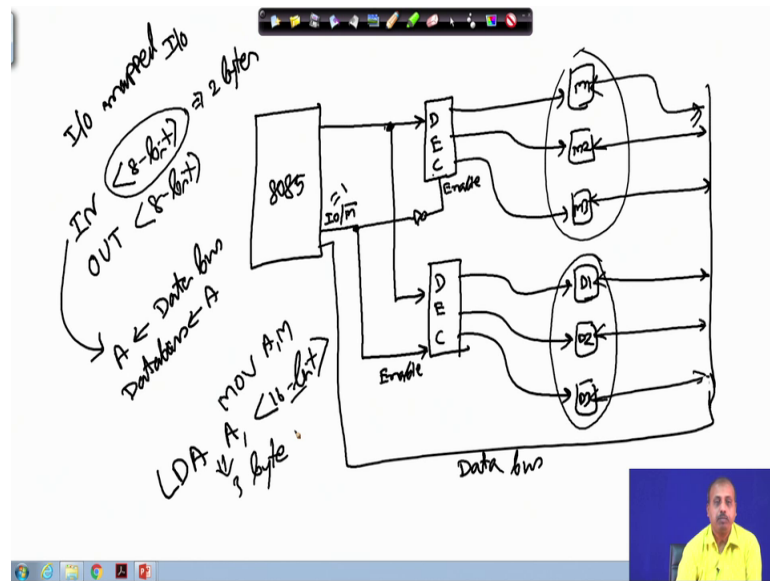
So, these are some special addresses that this memory should that that the processor should know, like if I have an instruction like LXI say if I have an instruction like MOV A comma M and before that if I if I do 1 LXI H comma some 16 bit value 16 bit address. Now, if this 16 bit address happens to be one location within this memory chips M 1 and M 2, then this MOV A comma M instruction. So, it will read from the memory location and get it into the 8085.

On the other hand, if this 16 bit address is such that the it is it is it is selecting this device then in the next MOV A comma M instruction. So, it will read from the device and get it into the memory. Similarly, if this 16 bit address is actually such that this decoder enables this line D 2, then this D 2 will be selected as the that content will be selected and put into the 8085 accumulator.

So, this type of operations; so, they are known as memory mapped I O operation because I O devices, they are treated as nothing, but some extension of the memory locations ok. So, this type of I O operation so they are known as I O sorry memory mapped I O operation. So, the access is exactly like the memory addresses.

On the other hand, if you look into this 8085 processor; so, you know that there is another special pin which is known as I O M bar. So, whenever the processor is doing this I O or this memory operation. So, it is making this I O M bar line equal to 0 ok. So, that is so far we have ignored this line, but we can use it for a special purpose like we can make it like this.

(Refer Slide Time: 24:28)



So, say this is the 8085 processor and here I have got the decoder corresponding to the for the memory chips. So, this decoder is dedicated for this decoding of the memory like this enables this enables this. So, this enables this like that.

So, we can have another decoder which we can dedicate for the I O devices D 1, D 2, D 3 like that. So, this D 1 this devices so, they are enabled by this now the address line. So, they go here as well as the address line they come here, but the I O M bar line that we have here. So, this I O M bar line is connected to the enable line of this decoder and the inverted version of that is connected to the enable line of this decoder fine now the 8085 it so for I O operation; so, it has got another type of instruction which are known as in and out instruction fine. So, this in instruction is in and one 8 bit address and this out is out and 8 bit address.

So, in instruction what happens is that it reads from the data bus and the content comes to the accumulator register. So, whatever is available on the data bus comes to the accumulator register similarly the out instruction whatever be the content of the data accumulator. So, that is put on to the data bus line now this for this in instruction and out instruction this execution. So, this I O M bar line where bit is equal to 1 where as for the instructions like MOV A comma M where it is doing memory operation this I O M bar line is made equal to 0.

So, this in out instruction; so, when they are executed this line is equal to 1 as a result these decoder is enabled and these decoder is disabled. So, it will not enable any of this memory chips, but it will be depending upon the address that we have here. So, it will enable one of these devices and that device value. So, whatever be will be the value put on to the data bus. So, all of them are connecting to the data bus all of them are connecting to the data bus. So, like this and this is connected to the data bus of the processor. So, this is the data bus of the processor.

Now, if I do this if I do this then this is the data bus. So, for this in out type of operations; so, it will get the data from this devices onto this data bus and that will come to the 8085 chip where as for this memory operation so it will get it from there. So, the point is that this in out instructions. So, they are operated in the I O mode and they come under the heading called I O mapped I O because the I O devices. So, they are having a separate decoder.

And they are having the I O addresses and maps separately, but only difference is that in case of I O mapped I O this in out instruction the address bus is 8 bit. So, it is not 16 bit. So, you can connect up to 256 devices only.

Now, but the advantage that we have is that this type of instructions. So, they have got only the length is only 2 bytes where as if you are having some LDA instruction LDA A comma some 16 bit address which load some 16 bit value which load some memory content to the address is this 16 bit value into the a register. So, this instruction is a three byte instruction ok.

So, that way memory mapped I O the instruction sizes are larger for I O mapped I O the sizes are small, sizes are 1 byte less and as for the operation is concerned. So, it takes less number of clock cycles because it has to read only 8 bit. So, so it is taking only 2 machine cycles that is 4 plus 3 7 cycles where as this memory operations. So, it is going to take some. So, this LDA instruction; so, this takes 3 plus 1; 4 machine cycles that is 12 plus the 9 plus 4 that is 13; 13 T states will be required for the LDA instruction. So, that will be there.

So, that is how this memory mapped I O and I O mapped, I O is there and in case of that it is the users choice of course, you need some extra hardware in case of I O mapped I O,



but we can do that we can use it for doing the I O operations at a much lower cost as far as the execution time is concerned.