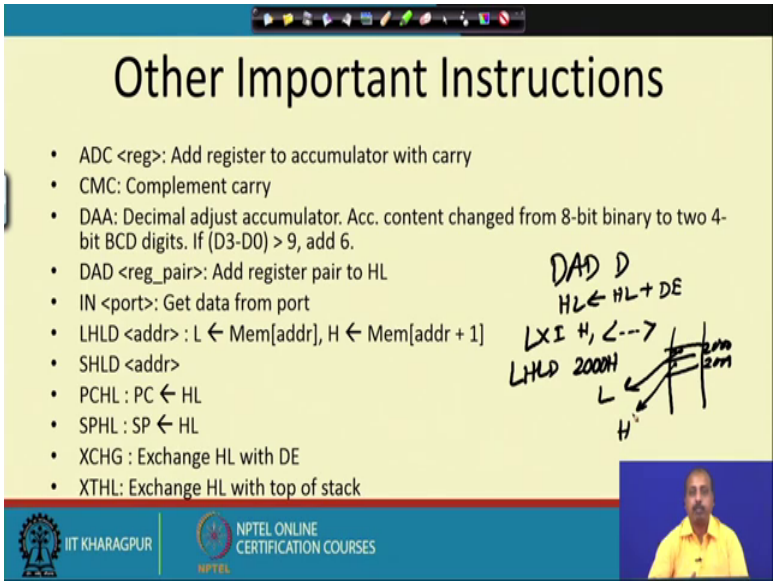
**Microprocessors and Microcontrollers**
**Prof. Santanu Chattopadhyay**
**Department of E & EC Engineering**
**Indian Institute of Technology, Kharagpur**

**Lecture – 22**
**8085 Microprocessors (Contd.)**

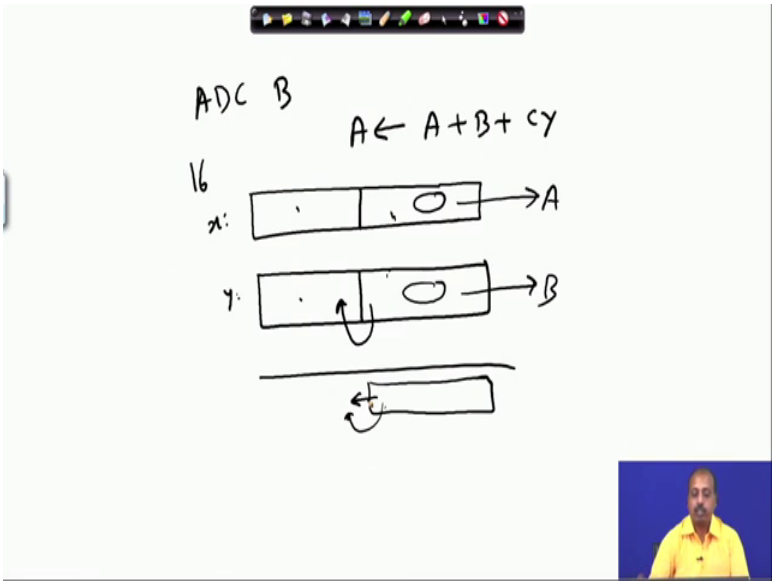
So, other important instructions that we have in 8085, so, there are a number of many interesting instructions that are there. So, that we did not discuss so far. So, we will just summarize them.

(Refer Slide Time: 00:28)




The first instruction that we have is known as the as the ADC; so, add with carry. So, add register way to accumulator with carry.

(Refer Slide Time: 00:44)




So, the instruction is like this. So, ADC B, what it will do? The register A it will get the content of A plus B plus the carry flag. So, many times it is necessary like if you are doing a multi byte addition like say if you are, you know that 8085 bit allows you to do only single byte addition, but if I have got numbers that can be represented in 16 bits. So, what we do this 16 bit number is consisting of 2 8 bits. So, this is the first 8 bit this is the second 8 bit.

So, after, this is say the number x and this is say the number y that we have. Now, while doing this addition, first part, first if this somehow this part is moved into the A register and this part is available in say B register somehow by doing some programming you can do that; then, when I add these 2, some carry may be generated that need to be propagated to the next stage in the sum. So, in the sum when I am doing it, so, I have got this part and it has got a carry to be propagated and while I am doing this addition this carry also needs to be added. So, if you execute simple add then that will not be possible, that carry will get discarded of this add with carry instruction can be useful when we are having this multi byte addition.

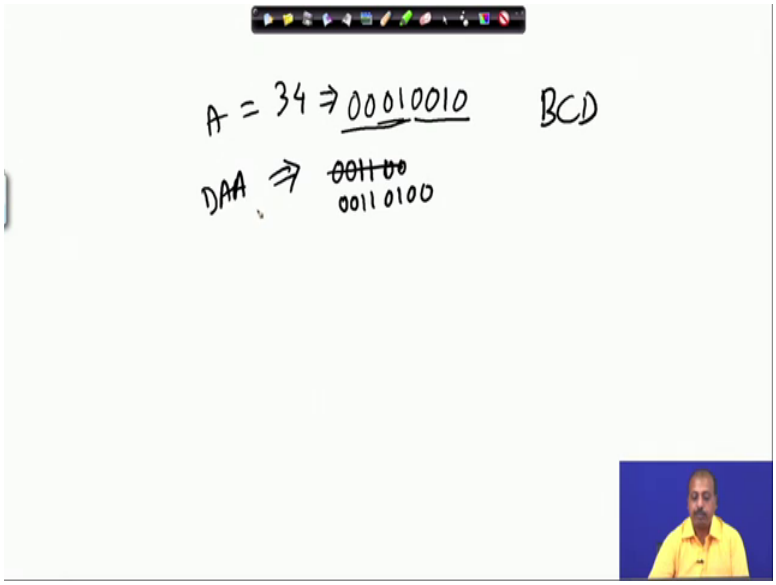
So, the next instruction that we will see is the complement carry, CMC. So, whatever be the previous value of carry, that will be complemented and you will get the new carry value which is the complement of the other one. So, this is also necessary at various times like if you are say transmitting some bit pattern which is say alternate 0es and ones onto the serial output line. So, you can use this CMC instruction after transmitting the one bit. So, you can

just do a CMC and then we do a rotate right or and then the carry will be coming to the accumulator again and then you again transmit it through the through the SOD pin.

So, that way by using that that (Refer Time: 03:10) instructions and using those complement carry instruction, you can transmit an alternating ones and 0s on to the serial output lines. So, this is just one example. So, there are many other cases where you need to do complementation of carry.

Next, important instruction that we have is the DAA or Decimal Adjust Accumulator. So, accumulator content changed from 8 bit binary to 2 4 bit BCD digits and how is it done if D3 to D0, this bit pattern is greater than 9 we add 6 to that pattern. So, how is it done? So, BCD number we have just introduced.

(Refer Slide Time: 03:55)




So, for example, if the number is say 34, in the binary number system, this will be stored as 10010 and these bits are 0. So, this is the 8 bit pattern; this is the first 4 bit and this is the next 4 bit for the decimal number 34.

Now, in BCD what we what will happen is that, if you execute, suppose this is the content of the accumulator register. Now, if you execute the instruction DAA, what will happen, this content will be changed to something like this. So, this will be 3 and 4. So, this will be 0011 and this 4 it will become 0100 sorry 0011 and 0100. So, it will become something like this. So, this way we can have this decimal adjust accumulator instruction. So, this will convert

this number into BCD number system and so, if this, the other instruction that we have is that that DAD. So, DAD register pair, add register pair to HL. So, this is another instruction. So, you have instruction like say DAD D. So, what it will do? The HL pair will get the value of HL plus DE. So, many times they are useful when you are using say array indexing and all that in your program. So, this register pairs need to be added. So, you can do that.

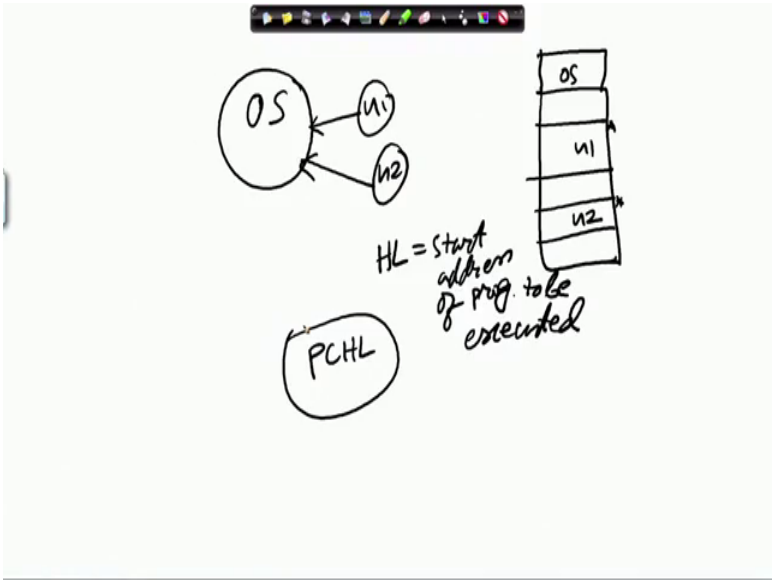
Next, we will look into we will come to this IN instruction later, in and out instructions are there. So, we will come to that later. Then, there is another instruction LHLD. LHLD it will be storing the L register, the memory register of address, LHLD address in that L register will get the lower register will get the value of the memory address and the H register will get the content of memory address plus 1.

So, this is the HL pair, it will be loaded from the memory. So, previously we have seen that LXI instruction LXI H comma some value, some immediate value. In that case we have seen that the HL pair is loaded with that immediate value. Now, this LHLD instruction it says that LHLD, memory locations 2000 x. So, what will happen if this is the memory location 2000 and the next one is 2001 then the content of this location 2000 will be loaded into the L register and content of location 2001 will be loaded into the H register.

So, this way this HL pair will get the content of memory locations 2000 and 2001. Similarly, we have got the SHLD instruction. So, it will be storing the content of HL register into the corresponding memory location; at the location address. Here, the same thing that the L register content will go to the memory location address and H registers content will go to the memory location address plus 1, due to that Intels policy that the higher order byte will be stored at higher order address. So, since H is higher order byte compared to L so that higher order byte will go to the next address and the lower order byte will go to the first address.

Then there is another instruction which is PCHL. So, this PCHL instruction, this is useful when you are trying to swap between this HL and program counter. So, this is the only instruction by which you can modify the program counter. Now, why do we may need to modify the program counter? So, apparently as a general user you do not need to modify the program counter, because if you reset the processor the program counter will be loaded with program counter will also be reset to 00000 and from that point onwards, the processor will go on executing the programs one after the other.

(Refer Slide Time: 08:26)




But, considering this situation where we have got where we have got some operating system and this operating system in any computer system, operating system is there and we have got a number of users, so, u 1, u 2 like that and they are submitting some jobs to the operating system.

Now, how they are executed. So, in the memory, we have got the locations the part the portions of memory which are dedicated to different users, may be this part is for u 1, this part is for u 2 like that. Now, when the operating system will try to execute the program corresponding to u 1, we can have the operating system code here. So, operating system is here sitting here and the when the system reboots the operating systems starts executing, after some time it may decide that now I want to execute u 1. So, how will it be done?

So, the way it can be done is by a single instruction like it can, somehow when this program will be loaded then this HL pair will be having the start address of the program. So, if this has the start address of the program to be executed. So, when the loader decides that it will be next executing the program u 1, it will load the HL pair with the start address of u 1. When the loader decides that it will be executing the program u 2, it will load the HL pair with the start address of u 2. So, start address means that these address or for u 2 this address. So, that will be done and after that the loader will execute a PCHL instruction.

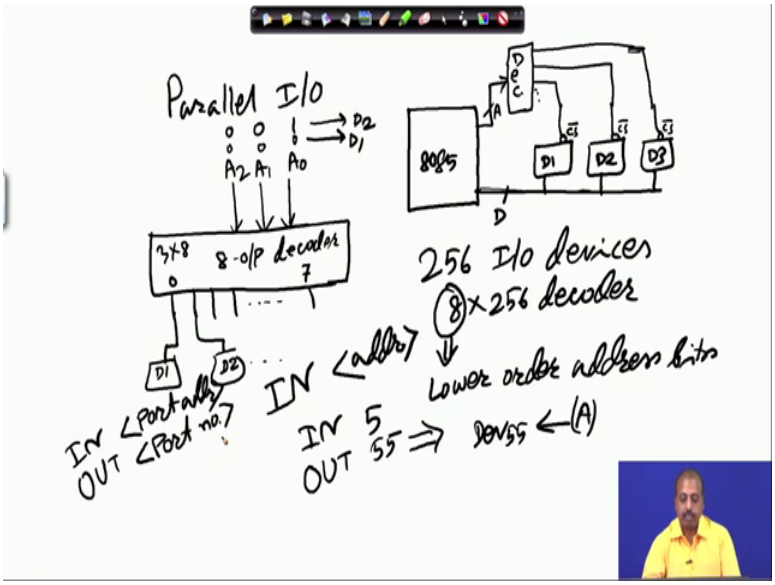
So, this PCHL instruction what it will do after say HL is holding the start address of the corresponding program. So, PCHL will load the pc with that program. So, this will be getting us the value of the pc value will be loaded with that HL pair. So, the HL pair, pc value will be pointing to the start of the user code. So, the user code will be executed. So, this way we can use this PCHL instruction for doing this program load and execution.

Another, similar instruction is the SPHL just like that program counter can be loaded with the HL pair. So, you can have the stack pointer loaded with the HL pair. So, if you want that different program they will load this stack point at differently so they can utilize the content of this HL pair to be loaded into the stack pointer. Then there is another instruction exchange. So, that will exchange the content of HL with the DE pair. So, HL and DE these 2 registered pair contents are exchanged.

So, this may be useful in some cases where we are again with respect to use of program execution, so, they may be required and just like this exchange you can have XTHL. So, it will exchange the HL pair with the top of the stack. So, top of the stack whatever it may contain, that will come to HL and HL content will go to the top of the stacks. So, that can happen. This may be useful again for program control. When you are designing operating system, this type of instruction the xc, so, this PCHL SPHL exchange and XTHL, these instructions are more useful from the system designers view point for general users. So, they may not have that much appeal, but for system designers, they may be useful.

Next, we will discuss about the IN instruction. In fact, there are 2 instructions in and out by which you can control the IO operation from the processor.

(Refer Slide Time: 12:29)




So, this is, basically comes under the broad heading of this parallel IO operation parallel input output operation. So, as we said that in my processor if this is the 8085 processor, then I can have a number of devices connected can have a number of devices connected. So, they are not memories, they are different IO devices D1, D2, D3, like that.

Now, how this 8085 can read the values from these devices or output some values to these devices. So, definitely for getting the values form these devices or to output some values to the devices, I should have the data bus of the 8085 connected to them. So, they must be connected to the data bus of the 8085.

Now, since there are a number of devices, we need to select one of these devices for reading the content and for that purpose, we assume that this devices they have got the chip select line and this when you are trying to select if you are trying to read a particular device or to write on to a particular device, the corresponding chip select line should be activated. And, now you can understand that here must be a decoder by which this 8085 should be able to select these devices for input output operation and that is done by means of another decoder.

So, externally, just like for memory interface we connect one decoder, here also I should have a decoder and this address bus should be connected to this. This is the address bus that should be connected to this. So, decoder will have a number of outputs. So, you can connect the outputs like this may be to the first one, next may be to the next one, third one to the third

one. So, that way you can have a number of devices connected and this decoder can be the address whatever is put.

So, depending upon the decoder that we are using, suppose, I have got say this decoder that we are talking about say this decoder has got 8 output. So, this is 0 and this is 7. Now, since, this is a 8 input decoders, there must be 3 inputs. 8 output decoder, there must be 3 input. So, let us say, we connect the lines A0, A1 and A2 of the processor of the 8085 to the inputs of this decoder. So, this is the 8 output decoder or you can say it is a 3 to 8 decoder. Now, to this 0, I have connected device-1, to this one I have connected devic-2. So, like that I have connected the device. So, for selecting this D1, the processor should put 0 in these 3 bits for selecting D1.

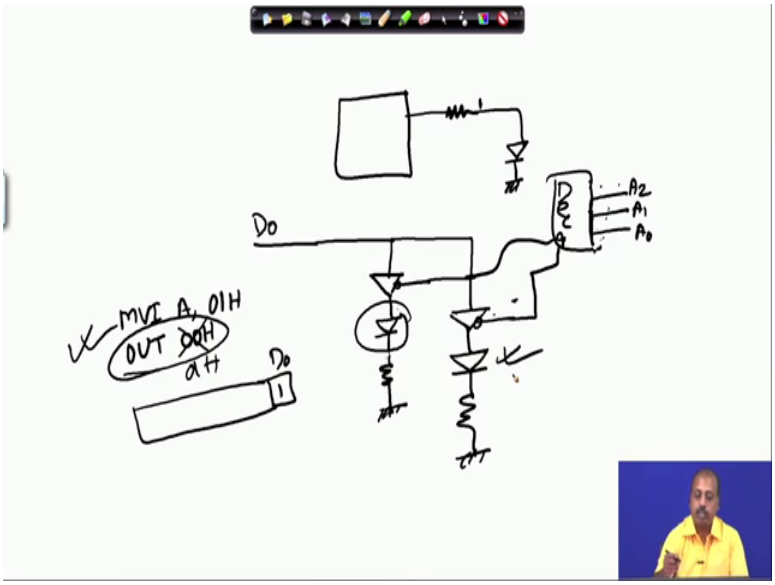
Similarly, for selecting D2, this is for D1 for selecting D2, it should put 0 1 that is for D2. So, this way for each of these 8 devices it should put the pattern 0 to 8. In case of 8085, we can allow 256 such IO devices connected in this fashion. So, the decoder that we can put is a 8 to 256 decoder. This 8 bits that I am talking about. So, this is nothing, but the lower order address bits.

So, this lower address bits, again you can say it is multiplexed and all that. So, for this lower order address bits will be connected to the decoder and from there this device will be selected. The instructions that we have for reading the content from a device is the IN. The IN instruction it accepts IN some address like you can say IN 5. So, what will happen? So, it will put the value 5 on to this address bus and as a result this device this decoder will select one the fifth device and the fifth device value will be available on to the data bus. It is expected that this fifth device will put the data onto the data bus and that value will be available on to the processor.

Similarly, if you want to output some a pattern to a device, you have to say like out say 55. So, if the device address is 55, that particular device to that particular device will write the value of the accumulator. So, in this case the device 55 will get the content of the accumulator. So, as far as the processor is concerned, each of these devices they are looked upon as a code. So, we call it as IN port address. Similarly, we have got this OUT and port address or port number can say port number.

So, this port number is an 8 bit value and that port number can naturally range from 0 to 255. So, this is this is how this IN and OUT instructions can be useful.

(Refer Slide Time: 18:54)




So, if we are connecting just one LED to the 8085, if we have got a simple LED somewhere here and we are trying to connect it then, if it is the data bus content then somehow we have to put a pattern 1 here and this part being grounded, you can have a limiting current, limiting resistance here and this LED will be turned on whenever we put a one on the data bus. But, of course, this particular feature it does not have the facility for this chip select and all that. So, for that purpose you need to put some tri state buffer and all that.

So, may be, if you want to augment this structures, you can do it like this. So, put a tri state buffer and this is the content of the data bus coming. So, put this is the tri state logic and then below this we have got that LED. We can put some resistance and then you can put some resistance and then you can put it to ground now. This is basically some sort of chip select, this may be connected to the line D0 of the processor of the 8085 processor and then this point may be coming from that decoder that I was talking about that address decoder.

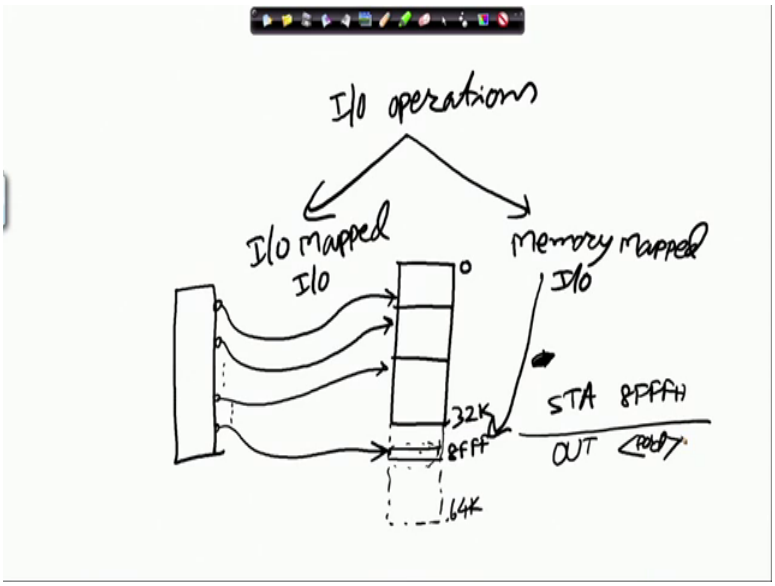
Now, the decoder will be when this bit is selected, may be the if this address is a port address is say 0. If this port address is a 0 H, then what this decoder will do. So, this decoder will have say 3 inputs there if it is a 3 to 8 decoder. So, the line A 0, A 1 and A 2 are connected here and this may be the line 0, so, it is connected to this.

Now, in this configuration what will happen when the processor will put this A 0, A 1, A 2 as 0 and this tri state buffer will be activated and then if this D 0 bit from the processor is 1, then

this LED will glow and if this D 0 bit from the process is 0, it will be 0. So, we can have a simple program like this that MVI A comma 0 1 H and then out 0 H. So, you have a simple program like this. So, when this will put the accumulator the least significant bit as 1 the D 0 bit as 1 and then when I am executing this out 0 H instruction this decoder will select this tri state buffer this tri state buffer will become enabled and it will be connected to this.

So, if you have got another such configuration, another such a LED buffer like this say from the D 0 line itself, if I just draw another tri state buffer and then I put another LED here and this is connected to say the line 1 from the decoder then when you execute this program only this LED will glow. Now, in another program instead of 0 if you make it is a 0 1 H then this LED will glow. So, that will be controlled by the address selection part. So, this way we can have this port addressing in 8085.

(Refer Slide Time: 22:55)




Now, another important thing that we have in 8085 is the addresses that we have talked about, IO operation, the input output operation, they can be classified into 2 categories; one is called IO mapped IO operation and another operation another category is called memory mapped IO operation.

Essentially, what it tells is that as far as an IO input output device is concerned, when the operation is taking place it is getting 8 bit values or the processor is outputting an 8 bit value to the port. Now, the port is in if that sense it is same as the memory location. So, memory

location is also 8 bit wide. So, at some point of time either you are reading from the memory or you are writing to the memory. So, for the IO operation also the OUT instruction is like writing onto the port and IN operation is reading from the port.

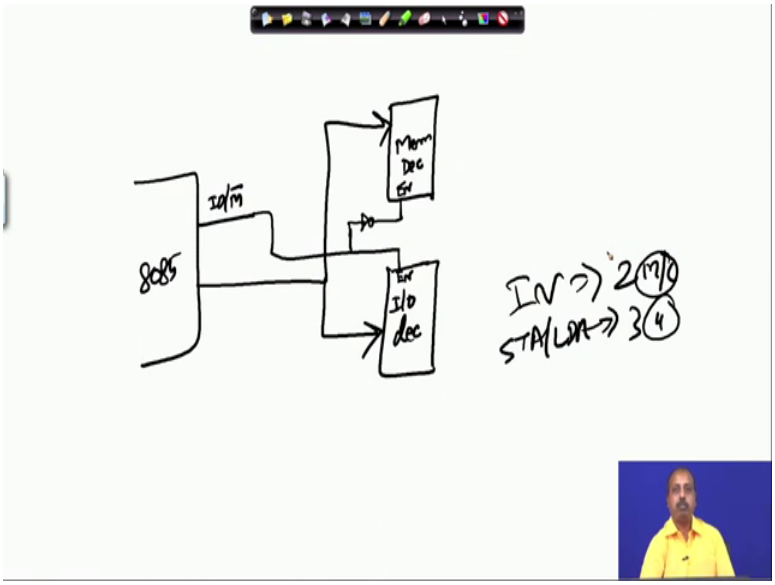
Now, in the memory map, what happens is that, if this is the total if these are the memory chips that we have in the system then the memory chips may be located in the from the memory address 0 to say 32K. So, these are the locations where we have got memory chips and remaining 32K I do not have any memory chip fine. In this total address space of the processor is 64K from 32K to 64K I do not have any memory chip. What I can do, I can use these some of these locations to connect the IO devices there. So, what we, in that case the advantage that we have is in instead of having 2 different decoders we can have a single decoder for both the memory and the IO operations.

So, we have got if this is the decoder some of the memory some of the chips enable line, they are enabling this chips, some of the decoder output they are enabling this chips and some of the decoder outputs, may be some time later. So, this output may be selecting this particular IO device. If that is the situation then these out operation and memory read operation they are similar.

So, if this address is say 8FFF then what will happen is that, instead of doing a out operation we can simply say a stored accumulator type of instruction we can say STA some 8FFFH. So, as a result the accumulator content it will try to store at location 8FFF and so the value that is there in the accumulator will be going to the device. This is though it is shown as a part of the memory map, in reality this is not the memory map. This is not a memory location, this is the IO device. So, this way I can visualize this IO operation as memory operations only and in that case. So, this they will be called memory mapped IO.

On the other hand, we have got these IO mapped IO where this IO operations are handled separately as the through the IO instruction. So, for IO mapped IO this STA instruction has to be modified to OUT instruction and the appropriate port number has to be put there and the decoders will be separate. So, if a system has got IO mapped IO then there will be 2 separate decoders one for the one for the memory chips and one for the IO devices. That way will have 2 different decoders and the decoding will be totally separate 2.

(Refer Slide Time: 26:56)




So, if I have got 2 decoders then one for memory and one for IO suppose this is the decoder for the memory part and this is the decoder for the IO part, then to distinguish their operation, we can take help of the 8085s pin which is known as the IO M bar pin. So, you remember this particular status line, which is set to 0 when the memory operations are taking place and this bit is set to 1 when IO operations are taking place. The same address bus lines will be connected to this memory decoder as well as this IO decoder and, but this IO M bar lines, this decoders they will have some enable pin. If we assume that there are some enable pin and this enable pin, if the decoder will be operating only if this enable value is 1.

So, for the IO part, I can connect it directly here and for this one, I can have an inverted connection. I can put an inverter and then this inverter can be connected to this to enable .
So, when the memory of the processor is doing memory operation then these IO M bar line is 0, and then this memory decoder will be enabled. As a result this memory operation that chip select will go to the memory chips and if it is IO operation then this either IN - OUT type of instructions then the IO M bar line will be equal to 1, as a result these IO decoder will be enabled and the IO operations will takes place.

So, that IN - OUT instruction they will select the IO devices and the operations will takes place. So, that way we have got memory mapped IO and IO mapped IO. So, memory mapped IO advantage is that we do not need this additional hardware, but at the cost that every IO operation is nothing, but a memory operation and this IN instruction it takes 2 machine cycles

whereas, this LDA STA type of instructions, they take 3 machine cycles. As a result one machine cycle extra will be needed when we are doing this STA LDA type of instructions. So, memory mapped IO execution time will be slower compare to IO mapped IO, but as far as the hardware is concerned IO mapped IO will be better.