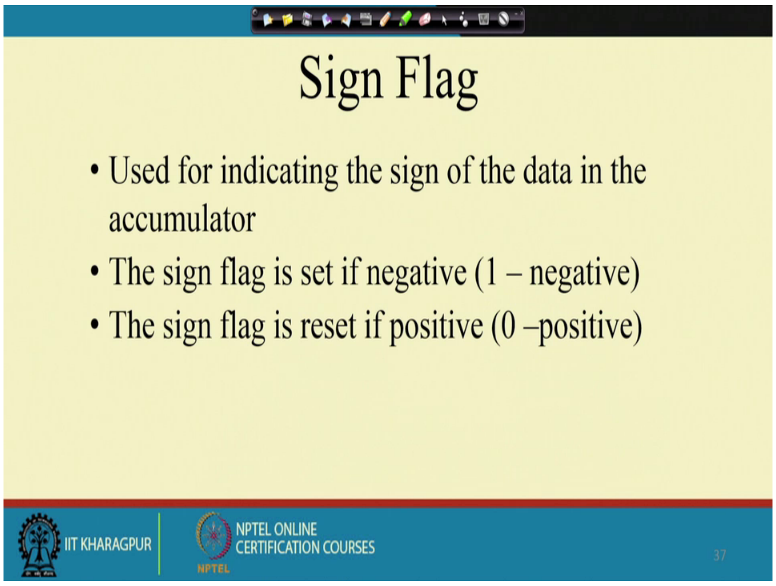
Digital Circuits

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

Lecture - 52
8085 Microprocessor (Contd.)

(Refer Slide Time: 00:18)

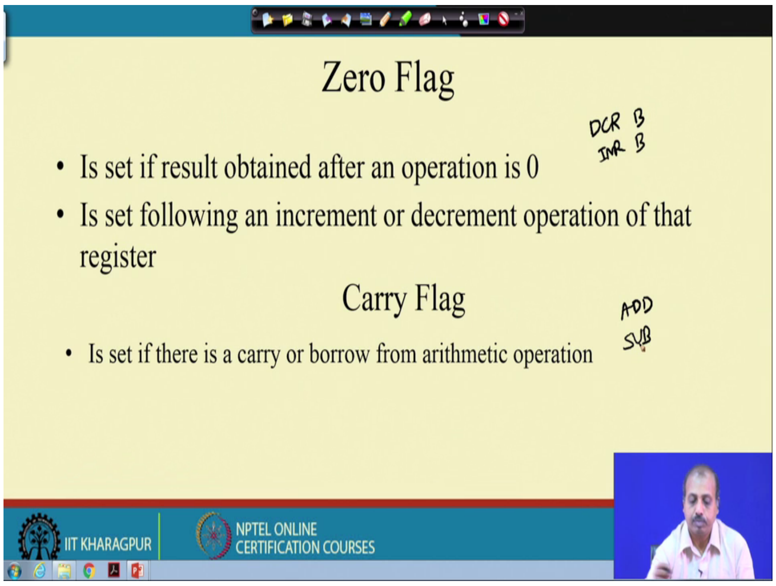


Sign Flag

- Used for indicating the sign of the data in the accumulator
- The sign flag is set if negative (1 – negative)
- The sign flag is reset if positive (0 –positive)

So, as far as the sign flag is concerned, so as I said that it is used for indicating the sign of the data in the accumulator. So, after you have done some operation, if the accumulator content becomes negative, in that case the sign flag will be set to 1; and if the accumulator content is positive then it will be 0.

(Refer Slide Time: 00:36)



Zero Flag

DCR B
INR B

- Is set if result obtained after an operation is 0
- Is set following an increment or decrement operation of that register

Carry Flag

ADD
SUB

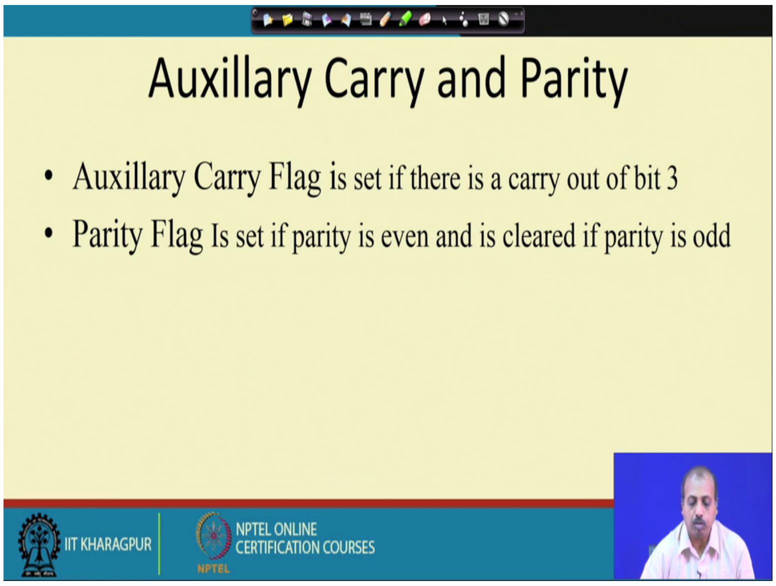
- Is set if there is a carry or borrow from arithmetic operation

So, then the 0 flag, so if the result is obtained is after the after doing the operation is 0. Then this is carry 0 flag is set and following an increment or decrement operation of the of the register. So, if you are doing some increment or decrement or some registered then also this 0 flag is set.

So, for example, if I have got an instruction like say we have got some instruction like say decrement B ok. So, this B register value will be decremented and this decrement operation if it is either if the B register content becomes 0 due to that then also the 0 flag will be set or maybe increment operation. So, INRB, so these are not on the A register ok. So, with the other registers also this is possible.

Now, the carry flag, so carry flag is set if there is a carry or borrow from the arithmetic operation. So, if the carry is generated if you are doing an add operation ok. So, after doing the addition if a carry is generated from bit number 7 or if you are doing a subtract operation then a borrow may be generated from bit position 7. So, that way this carry or borrow if it is generated due to arithmetic operation then the carry flag will be set.

(Refer Slide Time: 02:02)



Auxillary Carry and Parity

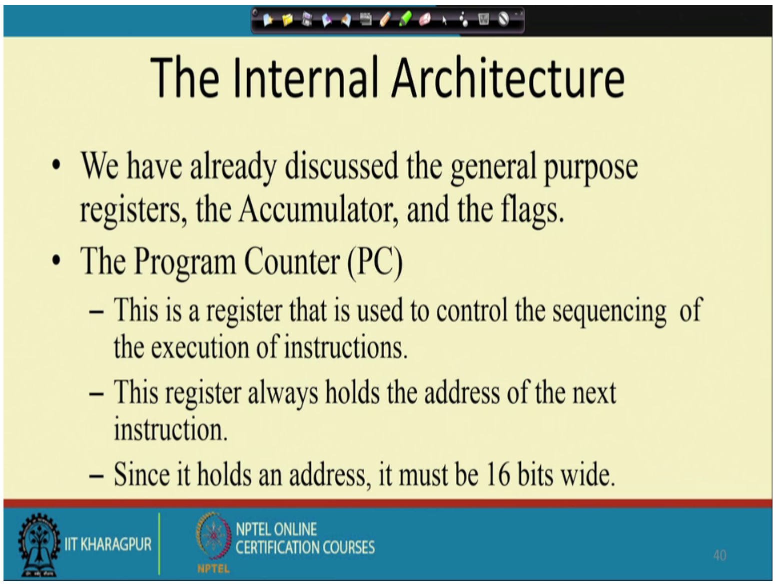
- Auxillary Carry Flag is set if there is a carry out of bit 3
- Parity Flag Is set if parity is even and is cleared if parity is odd

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

Now, auxiliary carry, so auxiliary carry flag is set if there is a carry out of bit 3. So, this I have already explain, so bit position three onwards if there is a carry due to either carry or borrow ok. So, either addition or subtraction instruction. So, carry or borrow is generated from bit 3 then this auxiliary carry will be set, parity flag is set if the parity is even and is cleared if the parity is odd.

So, 8085 it follows an odd parity structures. So, if the number of bits in the accumulator is even then this parity bit will be set. So, that the total number of bits total number of bits become odd. And if the number of bits in accumulator is odd, then this parity bit will be set to 0 telling that this is a the number of ones there is already odd. So, this way this parity flag is going to be used and some programs. So, it can use this parity flag to know whether the number of ones in the accumulator is even or odd. So, we do not need to count we need to check separately. So, we can just do a check on the parity flag and take a decision.

(Refer Slide Time: 03:14)



The Internal Architecture

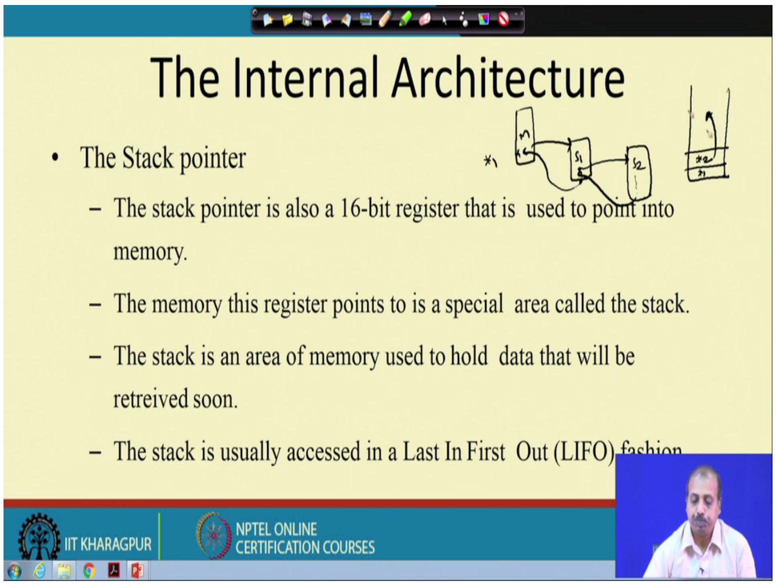
- We have already discussed the general purpose registers, the Accumulator, and the flags.
- The Program Counter (PC)
  - This is a register that is used to control the sequencing of the execution of instructions.
  - This register always holds the address of the next instruction.
  - Since it holds an address, it must be 16 bits wide.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 40

So, if you look into the internal architecture. So, we have already seen that general purpose registers the accumulator and the flags. Now next we look into the program counter, so there the program counter register PC. So, this is a 16 bit register and this register is used to control the sequencing of execution of instruction. So, as I said that at any point of time this program counter value it tells me, what is the address of the next instruction to be executed.

So, it always holds the address of the next instruction and it is auto incremented. So, as soon as we are accessing the next instruction this program counter value will be incremented, and it will point to the next address from where the fetch has to be done.

(Refer Slide Time: 04:09)



The Internal Architecture

- The Stack pointer
  - The stack pointer is also a 16-bit register that is used to point into memory.
  - The memory this register points to is a special area called the stack.
  - The stack is an area of memory used to hold data that will be retreived soon.
  - The stack is usually accessed in a Last In First Out (LIFO) fashion.

So, since it holds and address, so this is a 16 bit register and there is a stack pointer register. So, stack pointer this is also a 16 bit register, and it point it points to some memory location and this the memory this register points to is a special area called stack. So, this stack is used for holding the returner like in high level language program.

So, we are writing procedures and we know that from some main routine. So, we can call a procedure and from they are the from the procedure you come back to the main routine when the procedure is over.

Similarly, when you are writing program in assembly language in the assembly language then also you can write something called a subroutine. So, you can jump from the main program to a subroutine and after finishing that subroutine you can come back to the point at which you have called at subroutine, now this return address has to be saved somewhere. So, that at the end of the subroutine the processor will know where to go back ok. So, this return address is saved in the stack and the stack pointer is used for pointing to the memory location where the last address has been stored.

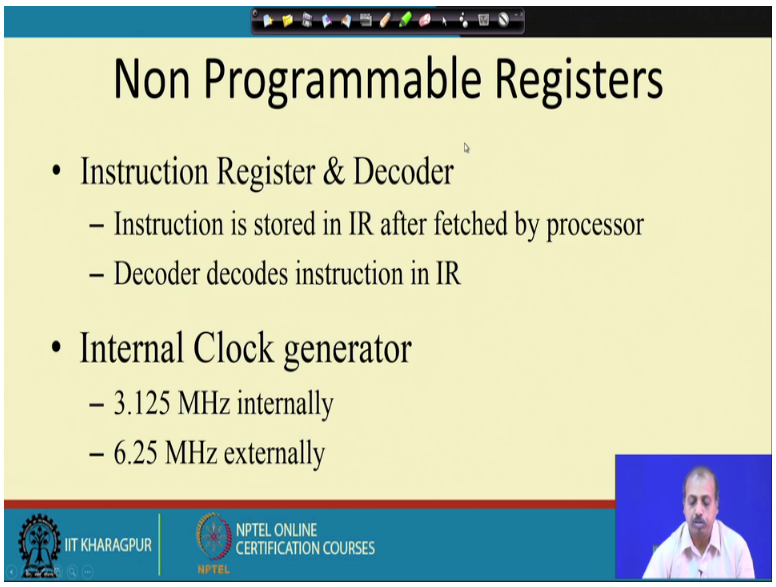
So, this stack is an area of memory the that is used to hold the data that will be retrieved soon. So, this may be that return address or it may be some parameters that we have passed from this main program to the subroutine. So, like that so they are they can be retrieved from the stack. And it is usually accessed in a last in first out fashion that is quite obvious because, whenever you are this is your main program and from here.

So, you have this is the main program and from here, so you have called a subroutine. So, this is the subroutine S 1 and within the subroutine somewhere here you have called another subroutine say S 2. Now when S 2 is over, so you have to go back to this point continue the execution and then from this end so you have to go back to this point.

So, this is in the so what so whenever you have jumping from M to S 1. So, if this is the stack in the stack you are saving the return address. So, you are saving this particular return address, so let us call it star 1. So, star 1 is saved here then when you are going from S 1 to S 2 that return address S 1 return address is saved here star 2 ok.

Then from when S 2 and so we take out this value and come back to S star 2. So, that is a so the number star 2 enter last into the stack, but it is taken out of the stack at the earliest. So, that is how it is called last in first out. So, whatever enter last will be taken out first. So, this way we can this stack will say it can be used for doing some operation.

(Refer Slide Time: 06:55)



Non Programmable Registers

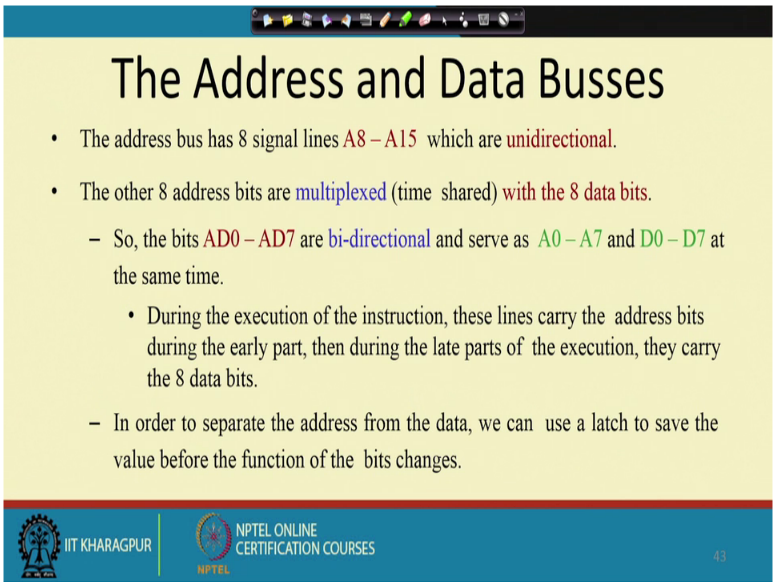
- Instruction Register & Decoder
  - Instruction is stored in IR after fetched by processor
  - Decoder decodes instruction in IR
- Internal Clock generator
  - 3.125 MHz internally
  - 6.25 MHz externally

There are some non-programmable registers like say instruction register and decoder, this instruction register this is a special purpose register is also known as IR. So, whenever and instruction is fetched from the memory. So, we have got this we have got this the fetched pattern stored in the instruction register. And from the instruction register it goes to the decoder the decoder will try to identify the meaning of the instruction.

So, this instruction is stored in the instruction register after fetched by processor and the decoder will decode the instruction into IR. So, this instruction register, so this is a being read and written, but from the users point of view. So, this instruction register is not accessible because, user will not be able to write something on to the instruction register or read the content of the instruction register through some program.

Now, there is a internal clock generator circuitry. So, the externally, so you have to connects a clock some crystal. So, if you connect clock with crystal 3.125 megahertz then outs the externally. So, there is a externally we connect this a crystal of 6.25 megahertz, and internally it is divided by 2. So, it becomes 3.125 megahertz ok, so that is the clock frequency at which this 8085 will work.

(Refer Slide Time: 08:24)



The Address and Data Busses

- The address bus has 8 signal lines A8 – A15 which are unidirectional.
- The other 8 address bits are multiplexed (time shared) with the 8 data bits.
  - So, the bits AD0 – AD7 are bi-directional and serve as A0 – A7 and D0 – D7 at the same time.
    - During the execution of the instruction, these lines carry the address bits during the early part, then during the late parts of the execution, they carry the 8 data bits.
  - In order to separate the address from the data, we can use a latch to save the value before the function of the bits changes.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

43

Then there are address and data buses, so this address bus has got 8 signal lines A8 to a 15, which are unidirectional the other at 8 address lines are multiplexed with the 8 bit data bus. So, this AD 0 through AD 7 they are bidirectional and such they are both the purposes of the lower order address bus A 0 to A 7 and the data bus D 0 to D 7, simultaneously it is not at the same time precisely at the same time.

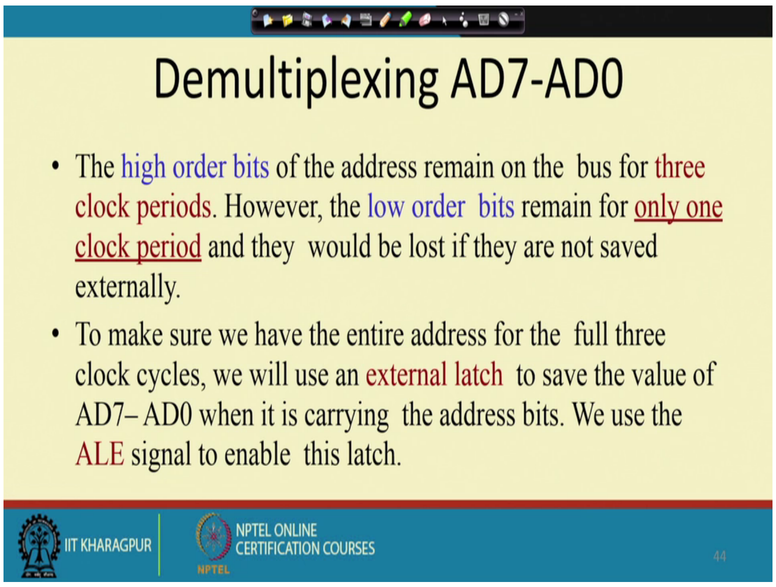
So, so when this address data bus this AD 0 to AD 7 is used as address bus. So, it is not used as date bus and similarly when it is used as a data bus, it is not used as address bus. So, during the execution of the instructions, so this the lines carry the address bits during the early part and then during the later part of execution they will carry the data bits.

So, basically if you are trying to access memory, so first you have to give the address and while giving the address. So, the address higher order part is an A8 to A 15 and the lower order part is in A0 to A7. After sometime when the address has been noted by the memory then you can, we can we can withdraw the address from the address bus.

And now whatever value this memory put on to the data bus it comes as AD0 to D7 to the processor. Similarly, if you are trying to write something on to the processor also to the memory also the same thing the first you put the address. So, memory understands the address then you withdraw the address and put the data bus content. So, to be written ok, so that is so now, the part acts as the data bus.

So, this in order to separate this address from data, so we can use a latch to save the value before the function of this bits changes. So, as I said that for initially for some time this A0 to A7 holds the address. So, in most of the memory design, so will find that this address bus content has to be held continually for that purpose we can have some external latch by which this address value is latch to there.

(Refer Slide Time: 10:30)



Demultiplexing AD7-AD0

- The high order bits of the address remain on the bus for three clock periods. However, the low order bits remain for only one clock period and they would be lost if they are not saved externally.
- To make sure we have the entire address for the full three clock cycles, we will use an external latch to save the value of AD7– AD0 when it is carrying the address bits. We use the ALE signal to enable this latch.

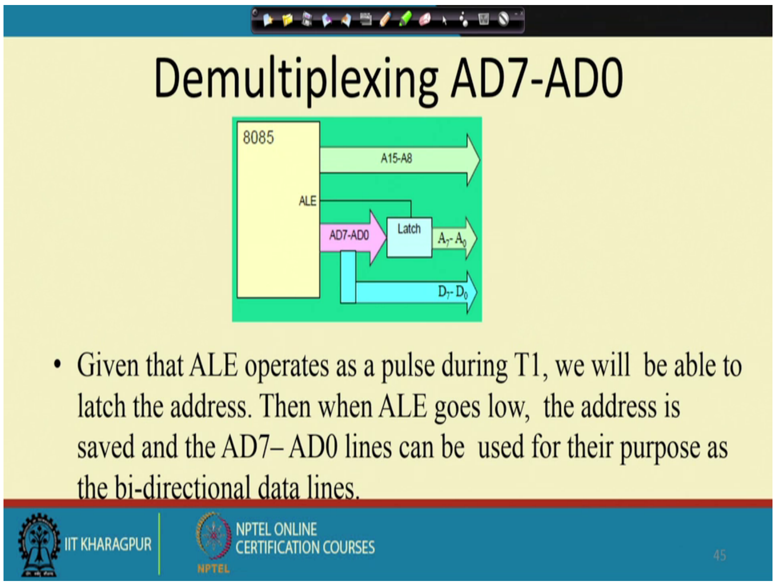
IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES NPTEL 44

So, this higher order address bus, so this has got this higher order address bits they will remain on the bus for three clock periods for 8085 this memory, memory access takes three clock cycles.

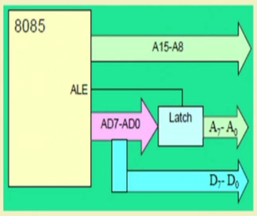
So, for three clock periods the higher order address bus will hold the value. However, as far as the lower order address bus is concerned. So, it is only for one clock period this lower order address bus holds the address bits valid address bits. So, what we need to do is after this one clock cycle. So, this value is withdrawn, so somehow we need to save this address somewhere.

So, that the memory sees continually that it has got all those address lines, but this for that purpose we have to have some external circuitry. So, which will use an external latch to save the value of AD 0 AD 7 to 0, when it is carrying the address bits and we will use the ALE signal to enable this latch.

(Refer Slide Time: 11:30)



Demultiplexing AD7-AD0



- Given that ALE operates as a pulse during T1, we will be able to latch the address. Then when ALE goes low, the address is saved and the AD7– AD0 lines can be used for their purpose as the bi-directional data lines.

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES NPTEL 45

So, this is the situation, so you see that this A8 to A15. So, actually this side we have got the memory. So, this side we have got the memory it is this is the memory. So, these are the address lines, so this these are the address lines for the memory and these are the this is the data line fine. So, this address lines, so this our this the higher order address bus it has got A8 to A 15 that is held continually for the entire read operation memory read or write memory axis operation. However, this AD7 to 0, so it has got this address values A0 to A7 only for the first clock cycle, and on for the first clock cycle is ALE signal is also activated.

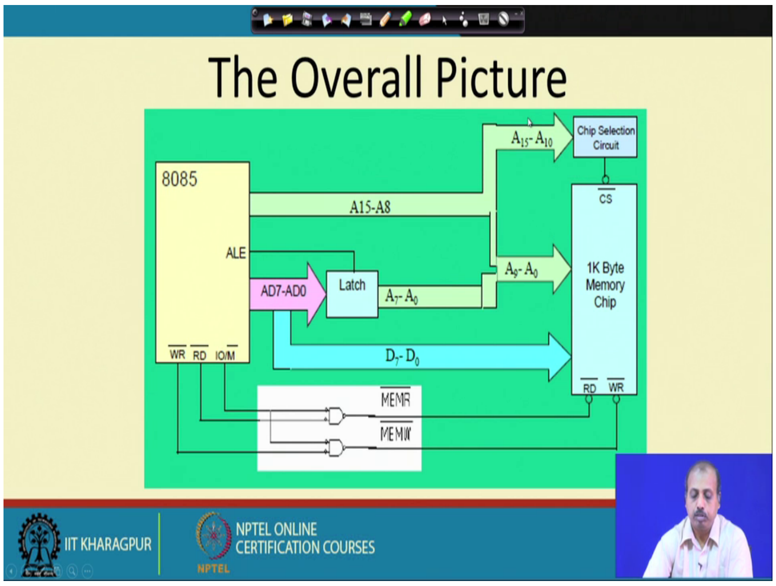
So, externally what you do you use a latch. So, that this when this ALE is high this value of A7 to A0 gets the stored into this latch. So, after that after of the first cycle this ALE signal is deactivated as a result this latch becomes inactive and it does not change its value even if this lines input to the latch changes since this ALE signal is low. So, latch will not put that value on to it, so it will continually hold this A0 to A7 at the output.

Now, after that this D0 to D7, so these are the data bus. So, there either the memory will put the content on to this data bus, or the processor will if it wants to write to the memory, so it can put the data onto this data bus essentially. So, this process as far as the memory is concerned it sees this address 16 bit address line continually for four cycles and it also sees for the for four cycles and it also sees the data for the last part of the operations. So, as a result, so it can do this read write operation easily. So, this ALE operates as a pulse during the first clock cycle.

So, when this 8085 is reset, so it every instruction access for the first cycle. So, it will give a pulse on the ALE line, so this will able this will this will be able to latch the address, because A0 to A7 is also put on to this AD7 to 0 line. So, that the address will get latched and then ALE will go low.

So, address will be saved and the line AD7 to AD0 can be used for other purpose that is there for holding the data. So, this is known as the demultiplexing of the address data bus. So, in many processers wherever we try to reduce the number of pins, so we can use this demultiplexing technique.

(Refer Slide Time: 14:22)



So, that pin requirement is low ok, so next is overall picture is like this. So, if I have got this ALE here 8085 here and we are trying to connect a 1 kilobyte memory chip ok, then this 1 kilobyte means, so this is the address bus is n bits. So, this A0 to A9, so they are to be connected and lines A10 to A15, so they total, total address generated by 8085 is 16 bit. So, out of that A0 to A 15 they will goes through an chip selection circuitry. So, in the in some classes earlier we have seen how to use this decoders to generate the chip select signals for various chips.

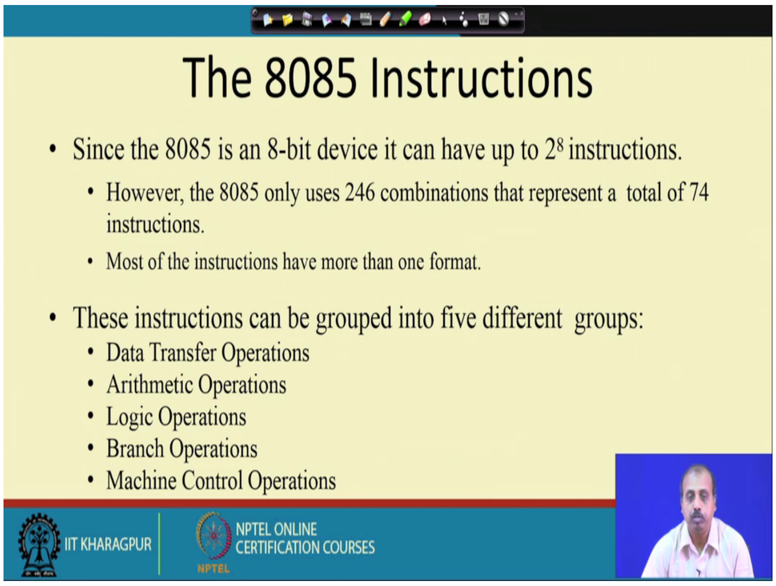
So, some decoding logic will be put here, so that it will select this particular chip for some address range. Now this A8 to A15, so they are coming directly from the 8085 has higher order address bus out of that A10 to A 15 is fetch the chip selection logic and A8 and A9 they will come to the lower side, now for the lower order address bus lower order address. So, AD7 to AD0, so they are passing through this latch and this ALE signal is selecting the enabling the latch. So, as a result A7 to A0 will be getting latched onto here.

So, ultimately, so this A0 to A7 and from this side you are getting A8 to A9. So, total A0 to A9. So, they are forming the address bus for the memory chip and this D0 to D7, so that is the data bus. So, it is there now I need to generate the read write signals for the memory. So, this IOM bar lines, so as I said that if M bar is low then this IO then this is a memory operation. So, this m bar and read bar, so if these two lines are inverted and then passed through a NAND gate then that means, so this is the memory read bar signal.

So, when the processor is doing a memory operation and it is doing a read operation then this memory read bar line will be low. So, as a result it is connected to this read bar pin of this chip. So, that it is doing the read operation similarly this write bar line and this IOM bar line. So, they are connected into this gate and it generates the memory write bar signal. And this memory write bar signal connects to the write bar signal of the chip.

So, this way this is the overall pictures, so if I have got more number of this is only one kilobyte memory chip interfacing that has been shown. So, if I have got more such chips then the chips selection circuit will generate appropriate chip selection logic. So, rest of the thing will remain unaltered.

(Refer Slide Time: 17:03)



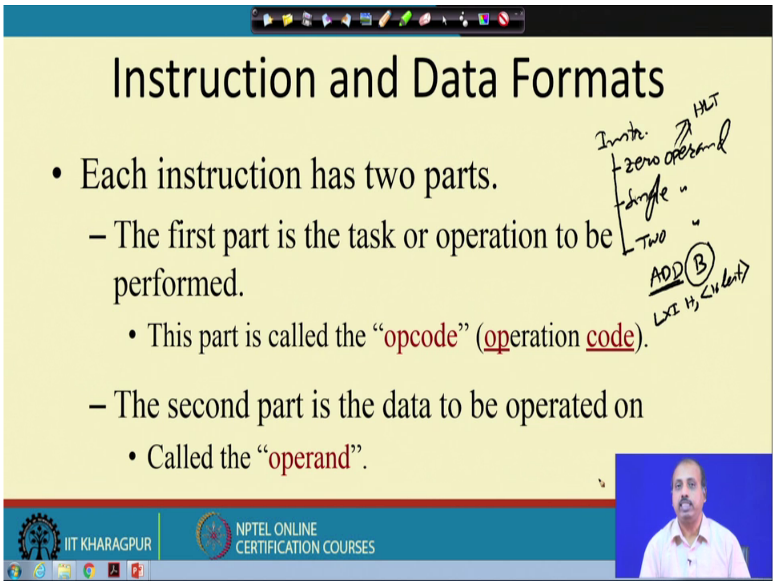
The 8085 Instructions

- Since the 8085 is an 8-bit device it can have up to 2^8 instructions.
  - However, the 8085 only uses 246 combinations that represent a total of 74 instructions.
  - Most of the instructions have more than one format.
- These instructions can be grouped into five different groups:
  - Data Transfer Operations
  - Arithmetic Operations
  - Logic Operations
  - Branch Operations
  - Machine Control Operations

Next we will be looking into the most vital part of 8085 like as a user of the system like how can I what are the operations that I can do with this 8085 processor. So, that defines the instruction set of this 8085, so 8085 is an 8 bit device. So, it can have up to 2 power 8 that is 22 246 instructions out of that only 246 combinations are used and that represents a total of 74 instructions. And naturally most of the instructions have more than one format.

So, we will see that this instructions that we have, so they can be grouped into five different groups data transfer, arithmetic, logic, branch and machine control. So, these are the different classes of instructions that we have in 8085, so you look into this individual classes one by one.

(Refer Slide Time: 17:58)



Instruction and Data Formats

- Each instruction has two parts.
  - The first part is the task or operation to be performed.
    - This part is called the "opcode" (operation code).
  - The second part is the data to be operated on
    - Called the "operand".

Instr.
zero operand → HLT
single "
Two "
ADD B
LXI H, <16 bit>

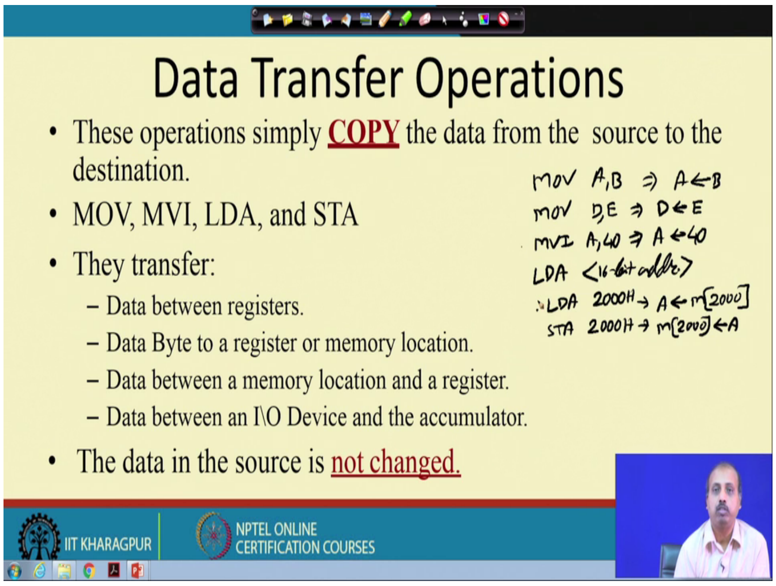
So, each instruction has got two parts the first part is the task or operation to be performed. So, this part is called opcode because it is operator it is it is telling the operation to be done. And the second part is that data to be to be operated on and it is called operant.

So, we have got opcode and operands, so any instruction can be divided into this opcode and operand part. Now depending upon the instruction that we have, so number of operands maybe 0 it may be 1 or it may be 2. So, we will see some instructions like that. So, say, so based on the number of operands the instructions maybe classified instructions may be classified as 0 operand, 0 operand single operand single operand or two operand. So, in case of 8085 we do not have more than two operands per instruction.

So, it is at most two operand, so 0 operand instruction like say the halt instruction. So, this is A0 operand instruction, so no operand is necessary single operand instruction I have already given a number of examples like say add B. So, this B is the operand other operands are other operand that is a is implicit for the instruction. So, you do not need to specified it separately. So, and this add is the memory, so that way this is a single operand instruction. So, two operand instruction like there are instruction like, which says that LXI H comma some 16 bit value, some 16 bit value. So, this tells that this 16 bit value has to be loaded into the HL register pair.

So, we will see this instruction later, but here you see this is an example of two operand instruction where H is the first operand and the 16 bit value that we have that is the second operand. So, this way I can have different number of operands in the instructions, next we will see how this instructions will look like ok.

(Refer Slide Time: 20:13)



Data Transfer Operations

- These operations simply **COPY** the data from the source to the destination.
- MOV, MVI, LDA, and STA
- They transfer:
  - Data between registers.
  - Data Byte to a register or memory location.
  - Data between a memory location and a register.
  - Data between an I\O Device and the accumulator.
- The data in the source is not changed.

MOV A,B $\Rightarrow$ $A \leftarrow B$
MOV D,E $\Rightarrow$ $D \leftarrow E$
MVI A,40 $\Rightarrow$ $A \leftarrow 40$
LDA <16-bit addr.>
LDA 2000H $\rightarrow$ $A \leftarrow m[2000]$
STA 2000H $\rightarrow$ $m[2000] \leftarrow A$

So, first category of instruction that we look into, so they are known as data transfer operation so, these operations they are many they are useful for copying the data from source to the destination address. So, the instructions that I have common is MOV, MVI, LDA, STA etcetera.

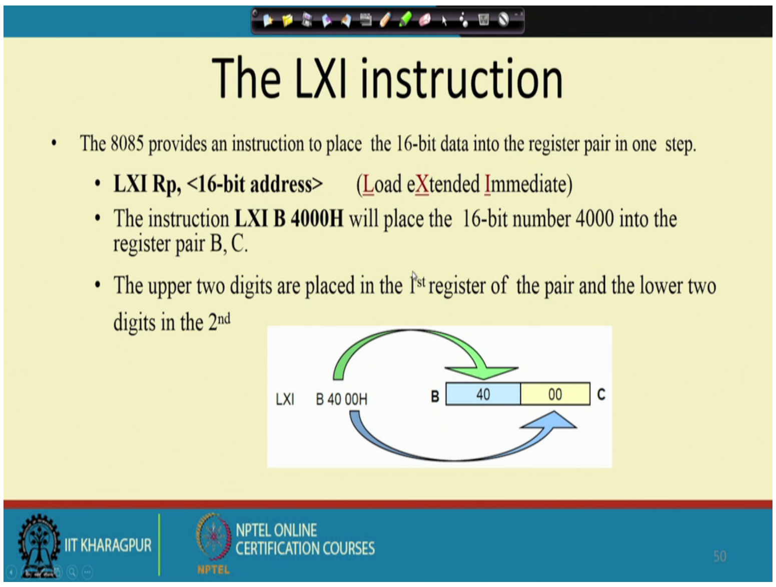
So, it is for example, we can have instructions like say MOV A comma B. So, this means the A register gets the content of B register or say MOV D comma E. That means, the D register gets the content of E register then there is MVI instruction MOV immediate. So, it is like this MVI A comma 40, so that means the A registered will get the value 40 ok.

Then LDA the format is LDA and a 16 bit address, like 16 bit address like LDA say 2000 H. So, this means the accumulator register will get the content of memory location 2000, fine then STA is for the it is just the reverse of LDA. So, STA 2000 hex that means, memory location 2000 will get the content of the accumulator. So, this way we have got this LDA, STA, MOV, MVI this type of instructions which is basically the copying of content of one, one register or memory location to another register or memory location.

So, data between they can transfer data between registers like MOV A comma B, data byte to A register or memory location like say this one MVI A comma 40 ok. Data between a memory location and A register like say this one LDA we have seen, or data between IO device and the and the accumulator. So, there, so that can also be done, so which is IO access will see separately later.

So, that is the IO device access can be done, but that data in the source is not change. So, these are all copying, so the source content will always remain unaltered. So, source content is not temporary with, so this category of instructions of they are known as data transfer instructions.

(Refer Slide Time: 23:02)



The LXI instruction

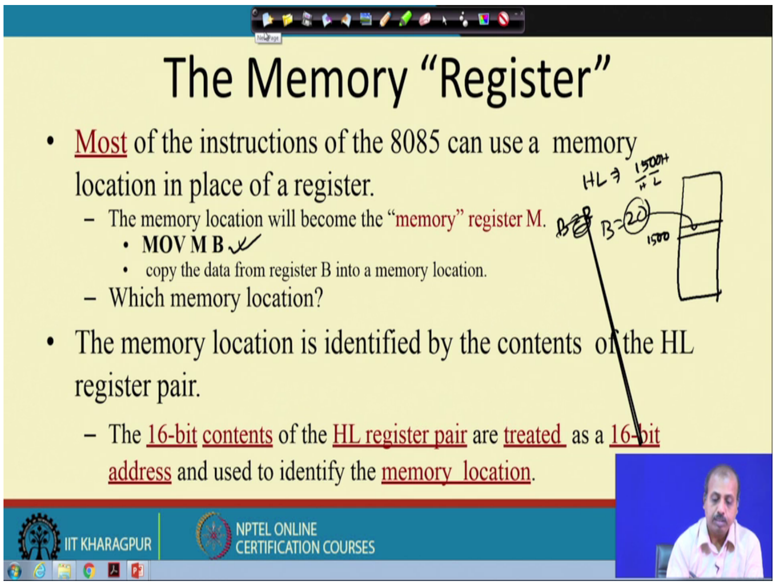
- The 8085 provides an instruction to place the 16-bit data into the register pair in one step.
  - **LXI Rp, <16-bit address>** (Load eXtended Immediate)
  - The instruction **LXI B 4000H** will place the 16-bit number 4000 into the register pair B, C.
  - The upper two digits are placed in the 1st register of the pair and the lower two digits in the 2nd

LXI
B 40 00H
B
40
00
C
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL
50

So, this LXI instruction, so this 808 so this LXI instruction it may this provides an instruction to place 16 bit data into the register pair in one step. So, this LXI the format is LXI register pair comma 16 bit address. So, it is load extended immediate, so it tells that this LXI B 4000. So, this will tell that the 16 bit value 4000 will be put into the pair B C ok. So, this B will get 4 0 and C will get 0 0 the upper two digits are placed in the first register. So, this is a hexadecimal number, so if I talk in terms of 8 bit value. So, 4 0 is the first byte 0 0 is the next byte 4 0 is the most significant byte 0 0 is the least significant byte.

So, the most significant byte goes to the B register and the least significant byte goes to the C register, so this is the LXI instruction.

(Refer Slide Time: 24:02)



The Memory "Register"

- Most of the instructions of the 8085 can use a memory location in place of a register.
  - The memory location will become the "memory" register M.
    - MOV M B
      - copy the data from register B into a memory location.
  - Which memory location?
- The memory location is identified by the contents of the HL register pair.
  - The 16-bit contents of the HL register pair are treated as a 16-bit address and used to identify the memory location.

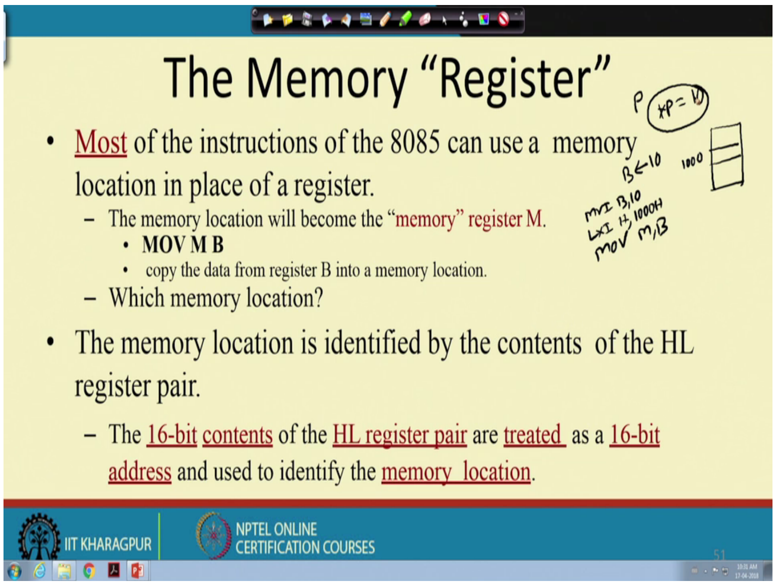
So, we can also talk about think about memory as a registered. So, most of the instruction is 8085 can use a memory location in place of A register like say this instruction. So, this MOV M comma B, so this memory location will become the memory register M, so this M comma B. So this tells that the content of a registered B will be copied on to memory location M, but what is the address. So, I so this, so when I say BCDE, you are talking about a particular register, but when I say M so, that is it is a memory location, so this memory location address has to be told and this address is implied it is implicitly identified by the content of the HL register pair.

So, it is like this if I have got this HL pair has got the value say 1500 hex that is H is having 15, and this is in H and this 0 0 is in L. Say that is the situation then if this is my memory this is the my memory and this is the location 1500 and if the content. So, the now in this instruction what will happen is that it will first go to the HL register and it will find out what is the content. It finds that the content is 1500 and then it will be copying the content of B register, content of B register suppose B register was equal to say 20.

So, suppose B register was equal to 20, so this 20 will be copied onto this location 1500. So, that is the meaning of this MOV M comma B instruction that we have. So, copy that the data from registered B into memory location and which memory location.

So, this is given by the 16 bit contents of the HL register pair and HL register pair treated as a 16 bit address and that is used for identifying memory locations. So, this is the way by which we can implement pointers ok. So, this HL pair, so it can be loaded with the address of the pointer. So, if I have as I was telling that if I have got a pointer P now.

(Refer Slide Time: 26:30)



The Memory "Register"

- Most of the instructions of the 8085 can use a memory location in place of a register.
  - The memory location will become the "memory" register M.
    - MOV M B
    - copy the data from register B into a memory location.
  - Which memory location?
- The memory location is identified by the contents of the HL register pair.
  - The 16-bit contents of the HL register pair are treated as a 16-bit address and used to identify the memory location.

So, you are writing like star P equal to say 10, so that can be done in this way the first this B registered can be loaded with the value 10 by instruction MVI B comma 10. And then if this P happens to be the memory location the address of P is say 1000 then this LXI H comma 1000 hex. So, that will be loading the HL pair with the value with this value, and then we will be doing this MOV M comma B.

If we do MOV M comma B then this content memory location thousand will get the value 10. So, this is equivalent to this statement star P equal to 10. So, this is the way by which pointers can be implemented. So, if we are using this indirect addressing in the HL register using the HL register pair.