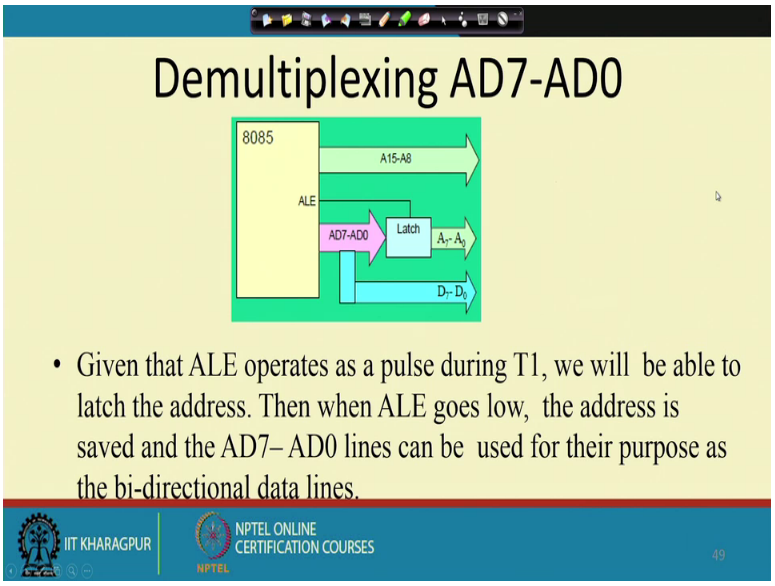
Microprocessors and Microcontrollers

Prof. Santanu Chattopadhyay
Department of E & EC Engineering
Indian Institute of Technology, Kharagpur

Lecture - 11
8085 Microprocessors (Contd.)

So, the demultiplexing of this AD 7 to AD 0 line is done as shown in this diagram.

(Refer Slide Time: 00:19)



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 49

So, this 8085 this A 8 to A 15. So, these lines they continually hold the higher order address bus for 3 clock cycles, then in the first clock cycle of the memory axis. So, these lines the A 7 to A 0. So, those bits are put onto these pins AD 7 to AD 0, and simultaneously this ale signal is generated.

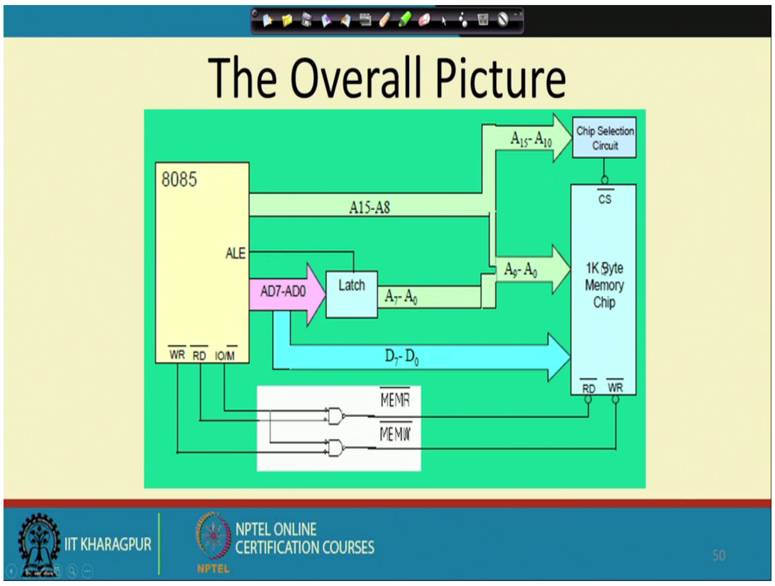
So, externally we use an 8-bit latch to store the values of A 7 to A 0 into this latch. And from this latch this lines A 7 to A 0. So, they we get a clear value here because after the first clock cycle this ale signal will be taken off. As a result, whatever value that was latched here in the first clock cycle it will they will remain here for the remaining 2 clock cycle.

So, you get a clear A 0 to A 7 value here. So, this A 8 to A 15 is there for 3 clock cycles, and A 0 to A 7 is also there for 3 clock cycles and now the memory getting this the 16-bit address. So, it can generate this value the memory location content in D 7 to D 0, and it will be connected to the data bus line.

So, that way this or if it is that the memory the processor is trying to write something. So, it can put the values of D 0 to D 7 onto the data bus onto these multiplexed pins now, but ale is not generated. So, it will be available on this D 0 to D 7.

So, the ale operates as a pulse during the clock cycle T 1 the first clock cycle, and we will be able to latch the address. And then the when the ale signal goes low the address is saved in this AD 7 to 0 line, and that can be used for are their purpose as bidirectional data line. So, we can use this after it has been latched. So, this part is free to act as the data bus.

(Refer Slide Time: 02:19)



The Overall Picture

So, the overall picture; how does it look like? So, the memory and this processor 8085 like that. So, from 8085 we have got suppose we have got a memory chip which is one kilobyte chip. So, this memory chip is. So, if there are one kilo location and each location is 8 bit wide fine.

So, since it is one kilobyte memory. So, that means, I need 10 address lines to access individual locations within this chip. So, it is A 0 to A 9. So, they are actually needed for accessing this particular memory, because this memory has got 10-bit address line. So, out of that say this A 0 to A 7 will come from this latch, and here A 8 to A 15 are going.

So, from there I need to connect A 8 and A 9. So, those 2 bits will be coming from here, 8 bits will come from here totally making a 10 bits here A 0 to A 9 they will come here.

The remaining lines a 10 to A 15 they will be fed to some chip selection logic circuitry. So, as we have discussed in that memory interfacing classes, there we can have some sort of decoder by which we can generate appropriate chip select signal for this chip. So, whatever be the address range in which we want to put this chip, accordingly we can connect this a decoder output to the chip select line of the chip of the of the memory chip.

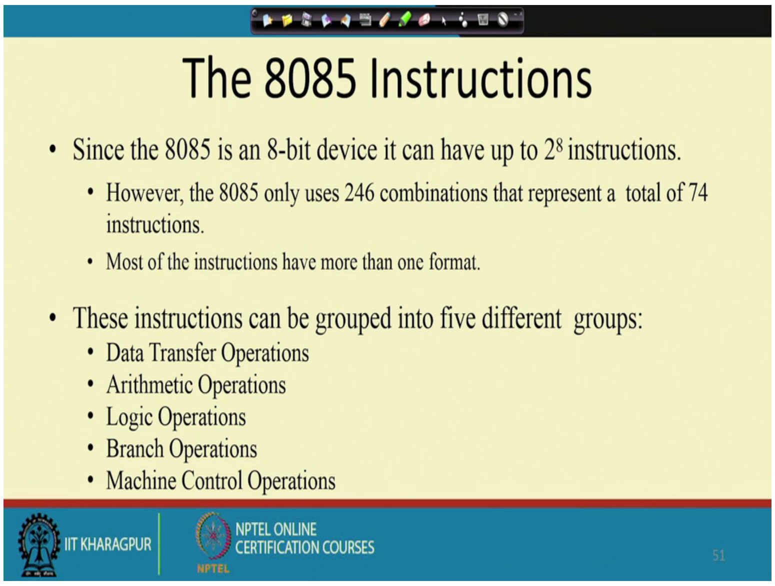
So, accordingly in that address range. So, this is the memory will get selected. Now as far as this AD 7 to AD 0 line is concerned. So, this is bidirectional. So, though it is not shown here explicitly, but this is the bidirectional line because the data bus is also there. So, this ads; so initially this A 0 to A 7 will come here the latch will latch it. So, that will get it here, but the data bus will be connected this D 0 to D 7 line. So, the you will get a you get a you can think that I am getting a clean data bus line here, because the address bus has been separated it out separated out at this point.

So, this is straightaway connected to the data bits of the memory chip. So, and also, we need to access this chip when it is doing a memory read operation or a memory write operation. Accordingly, this ram chip so, it has got a read bar control and a write bar control if you are trying to read something from the memory then this read bar line should be made 0, if you are trying to write something on to this memory, then this write bar line should be made 0.

How these lines are generated? So, these are generated by doing this read bar and I O M bar. So, these 2 lines are inverted and then the it is nand gate. So, accordingly we get the memory read bar line. So, that will go to the read bar connection here. Similarly, this write bar line and this I O M bar line. So, when they are nand gate that they are inverted and then nand gate, and then that generates this memory write bar signal, and that goes to the right bar line of the memory.

So, this way the chip can be interfaced with the 8085 processor. So, if you have got multiple such chips, then this chip selections circuitry will become more complex, accordingly a number of chip select signals will be generated, and they will be given to various processors where various memory chips.

(Refer Slide Time: 05:49)



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

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES 51

Next we will look into the instructions of 8085. So, as to use any microprocessor, we need to understand a few things, the first one we have already seen is the set of general purpose registers that the processor supports that B C D H L in case of my 8085. So, that is one thing.

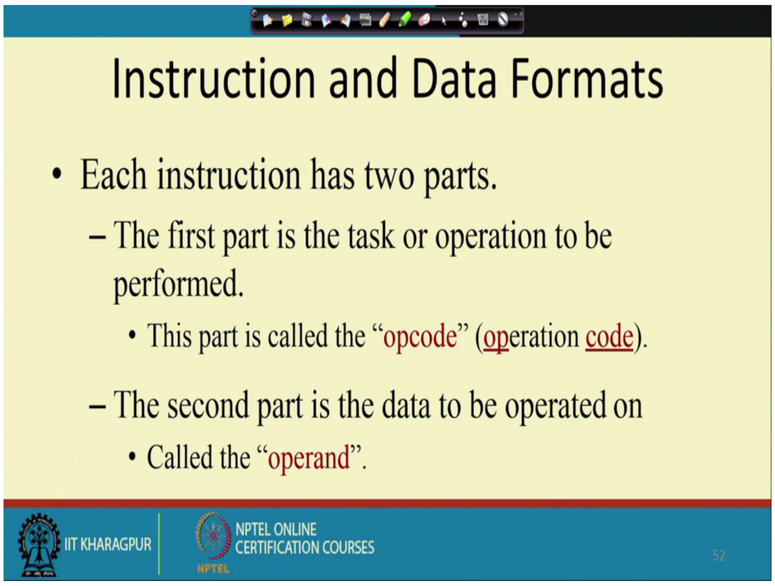
Second thing that we need to know is the set of instructions. So, what are the instructions that are supported? So, these 2 things taken together is known as the processor programming model, or ppm in short; so processor programming model. So, whenever I am talk coming across a new processor, and new microprocessor. So, I should try to learn; what is the processor programming model for this microprocessor. And once I know that I will know about the resources that I have I will know the instructions that I have in my hand, and accordingly I can write down the programs for that microprocessor.

So, since 8085 is an 8 bit device, it can have only 2 power 8 instructions; however, we have already seen that it uses only 246 combinations, and that represents a total of 74 instructions only. And most of the instructions have more than one format naturally. So, 74 instructions and they are using 246 combinations. So, naturally each instruction may have more than one format. And if you do a grouping of the instructions they can be grouped into 5 different categories. Data transfer, arithmetic operation, logic operation, branch operation and machine control operation.

So, data transfer operations they will talk how to transfer data between the CPU registers between memory locations between CPU which register and memory locations like that, how the data can be transferred, from one position one storage to another storage. One storage location to another storage location; then the arithmetic operations they will tell the arithmetic operations that we can do. Logic operations we will be talking about the logic things like and or nand nor etcetera, then the branch operation.

So, they are actually for transferring control. Like if in a program you are trying to implement a loop, or you are trying to implement an if then else type of structure in an assembly language program, then you will need these branch operations, in which you have to jump from one location to the other based on some condition so that way the branch operations will come into utility. And there are some machine control operations by which you can control the way this the processor behaves. So, we will see them later.

(Refer Slide Time: 08:34)



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".

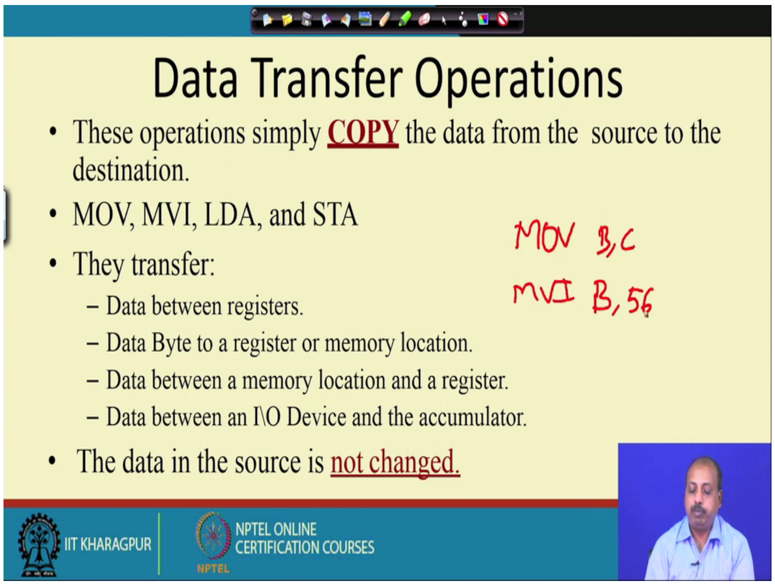
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | 52

So, each instruction can be thought of to be consisting of 2 parts. The first part talks about the task or operation that we want to perform, and this is called the opcode part. So, opcode the word stands for operation code. So, this part tells us like what is the operation that I want to do, and the second part is called the operand.

So, second so, what is the opera on which data we want to do that operation. So, that is the operand, and as you as we have seen that there are 246 different combinations that are supported by 8085 for only 74 different instructions. So, or the 74 are the 74 different opcodes that we that is therefore, supported by the 8085 processor, and by looking into different type of operands that are possible. So, that number reaches 246.

So, for same opcode by varying these operands; so we can get different instructions.

(Refer Slide Time: 09:35)



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 B,C

MVI B,56

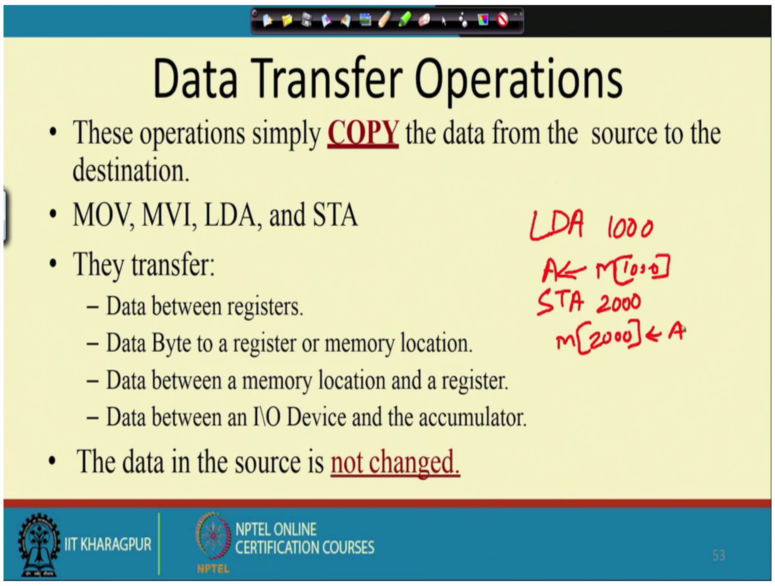
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL

So, first group of instruction is the MOV instruction is a data transfer operation. So, you want to convey you want to copy the data from source to the destination. Like, you can say the this instructing move MOV stands for move shorthand for move, when MVI move immediate, then LDA, which is load accumulator. And there is STA which is store accumulator.

So, what do they do? Say, they transfer between registers, like this MOV instruction like you can say MOV B comma C. So, we can have an instruction like say MOV B comma C. So, what it means the content of this C register will be moved to the B register. Or you can have say you can have say MVI the move immediate B comma 1 byte can be specified say 56. So, that means, I want to value want to move the value 56 into the B register. So, in this way I can think I can talk about various operations that we can do in various movements that we can do in 8085.

Similarly, there is this LDA and STA instruction; so this LDA and STA instructions. So, they talked about how to move values to the accumulator. So, you can from memory location.

(Refer Slide Time: 11:19)



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.

LDA 1000

$A \leftarrow M[1000]$

STA 2000

$M[2000] \leftarrow A$

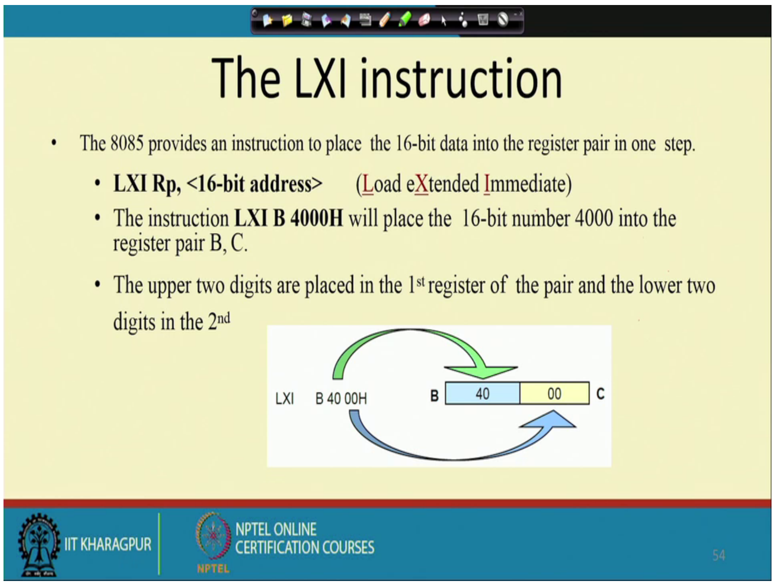
So, you can load that value into accumulator. So, there is this LDA instruction is something like this LDA 1000. So, it essentially means that accumulator will get the content of memory location 1000. So, it will get the content of memory location 1000.

Similarly, you have got STA, say STA 2 or 2000. So, it means that the memory location, 2000 will get the content of the accumulator register. So, this way I can have this data transferred between accumulator and memory locations. Or you can even there are some MOV instructions so that we will see later. So, the by other registers can also be used for similar purpose for doing this data transfer from memory to say the CPU registers. And between registers we have got MOV instruction.

And so, MOV instruction is for between memory locations, and between memory location and from between CPU registers or CPU register and memory. Or the LDA is for either LDA and STA, these are special so, these are with respect to accumulator only. And this MVI is for immediate. So, if you want to move some constant value to some register. So, you can use this MVI instruction.

The data in the source is not changed; so all these MOV instructions or the load or store instruction the store the source from where the data is being moved. So, that is not change. So, in some sense this is not a move this is basically a copy operation.

(Refer Slide Time: 12:58)



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
54

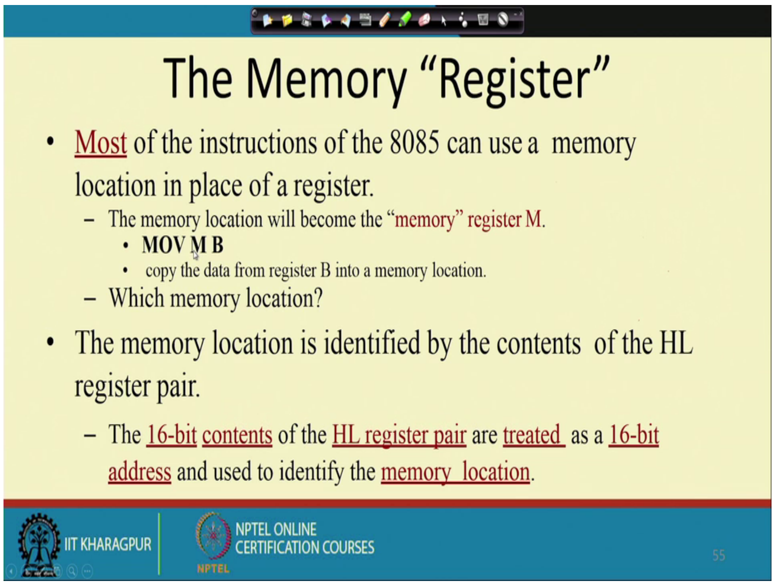
So, that is why it is said that it is simply a copy operation.

So, the next instruction that we will look into is a special instruction which is called LXI. So, it says that it is a load extended immediate. So, the load extended immediate is LXI. So, this is for loading 16-bit value onto some registered pair. So, LX the format is like this LXI Rp some 16-bit value. So, it means that so, in that register pair we will we will load this 16-bit value directly.

So, typical example is say LXI B 4000 hex; so the when I say 4000 and H at the end. So, that means, I am talking about a hexadecimal number. So, in assembly language programs so, normally that is the custom followed. So, if you follow in number by the character H; that means, it is hexadecimal number otherwise it is taken as a decimal number of course, it can vary from assembler to assembler, but this is some common convention that is used in most of the assemblers.

So, these 4000 hex this value will be moved on to this B register pair. So, B and C form a 16-bit register pair. So, this 4000 number will go to the pair BC and out of that this 4 4 0 being the higher order byte will go to the higher at the higher register, that is B register and this 0 0 being the lower order byte will go to the lower register. So, upper 2 digits are placed in the first register of the pair and the lower 2 digits are placed in the second register of the pair. So, that is the LXI operation.

(Refer Slide Time: 14:45)



The Memory "Register"

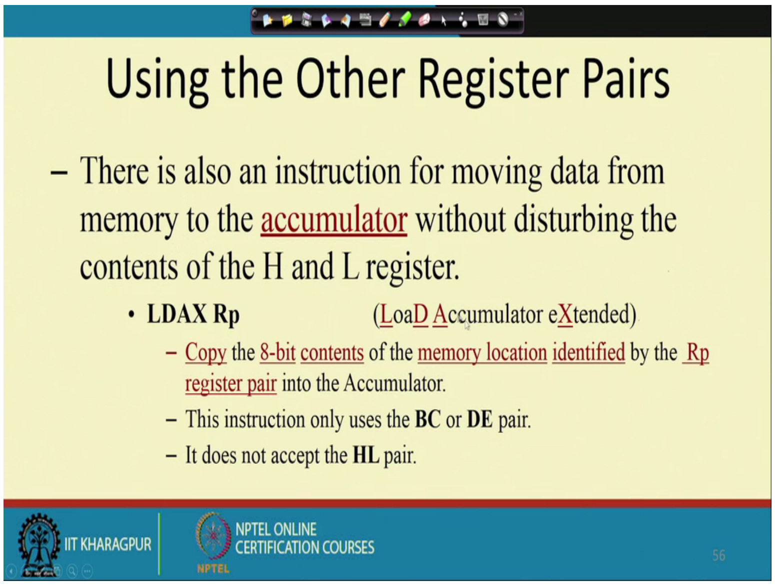
- Most of the instructions of the 8085 can use a memory location in place of a register.
  - The memory location will become the "memory" register M.
    - MOV M B
    - copy the data from register B into a memory location.
  - Which memory location?
- The memory location is identified by the contents 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.

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

Then the once this LXI has been learned. So, we can talk about another very interesting operation we are or another very interesting type of register which is called memory register. So, what it means is that, you can treat as if the some memory location as it a register also. Like most of the instructions in 8085 can cause a memory location can use a memory location in place of a register typical example is MOV M comma B. So, what it means is that we want to copy from data register B into a memory location. So, the data from B register will be copied into memory location M.

But what is this memory location M. So, this is basically indicated by the H L pair. So, H L pair suppose this H L pair contains the value 2000. So, in that case that H L pair value will be used as a pointer, and that was there. So, and the location content of that particular location will get modified. So, if H L pair contains the value 2000, then the memory location 2000 will get a copy of the value of register B. So, that way it becomes a technique by which you can imp implement pointers. So, you can use the H L pair as the pointer.

(Refer Slide Time: 16:15)



Using the Other Register Pairs

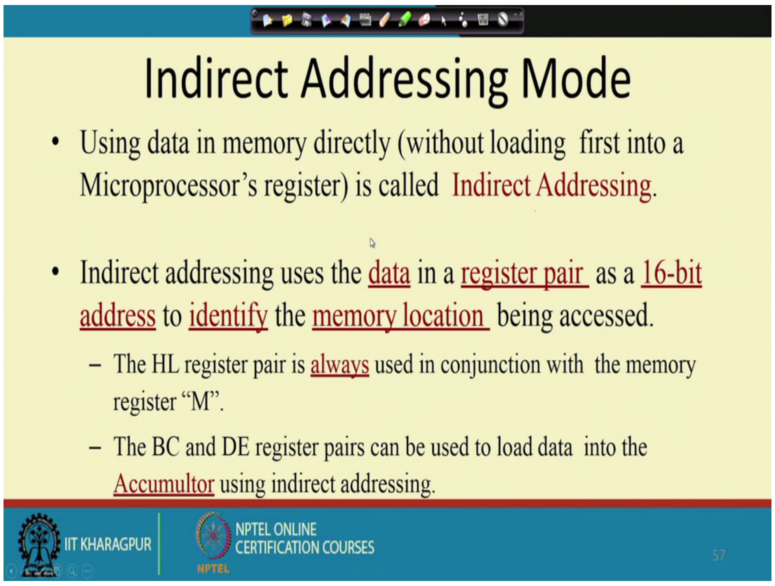
- There is also an instruction for moving data from memory to the accumulator without disturbing the contents of the H and L register.
  - **LDAX Rp** (LoaD Accumulator eXtended)
    - Copy the 8-bit contents of the memory location identified by the Rp register pair into the Accumulator.
    - This instruction only uses the **BC** or **DE** pair.
    - It does not accept the **HL** pair.

So, apart from H L so, this H L pair is modified by that LXI H type of instruction. So, you can also use other registers for that. So, you can say that the you can use an instruction to that that LXI is sorry, that MOV a comma M that instruction the previous instruction that we are talking MOV M comma B or whenever the M is the operand, then it is the H L pair which acts as the pointer.

So, if you want that I if you if you want to use some other register pair as the pointer you can do that by using this LDAX instruction; so LDAX Rp. So, this means the load accumulator extended, and whose the location is pointed to by this Rp. So, you can have LDI LDAX B or LDAX D. So, that way you can say that. So, if you say LDAX B the operation is that the accumulator will get the content of this memory location pointed to by the BC pair. If it is LDAX D; that means, the accumulator will get the content of memory location pointed to by the D E pair, but it will not accept H L pair. So, LDAX H is not a valid instruction, it does not accept the H L pair.

So, H L pair is used for indirect access, and by which you can you can get the content of some memory location to some register. And this LDAX is another indirect access where this content of B and B C and D E pairs, they can be used as the indirect entries.

(Refer Slide Time: 18:06)



Indirect Addressing Mode

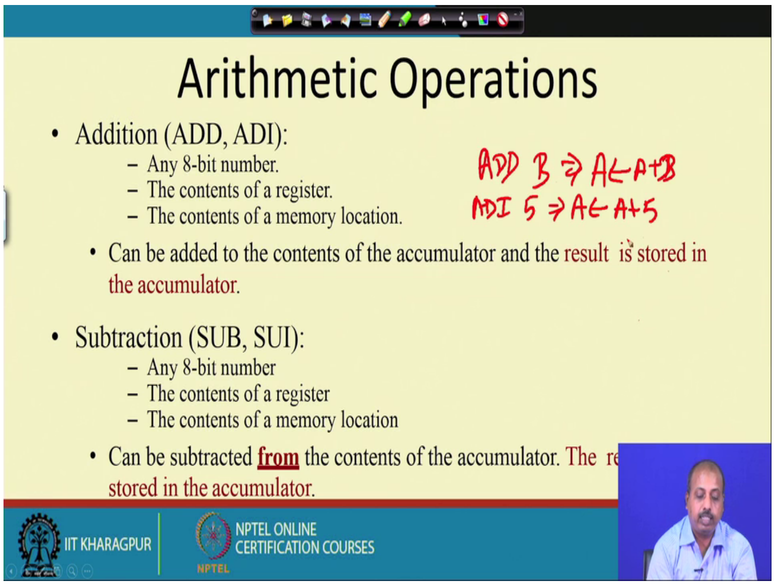
- Using data in memory directly (without loading first into a Microprocessor's register) is called Indirect Addressing.
- Indirect addressing uses the data in a register pair as a 16-bit address to identify the memory location being accessed.
  - The HL register pair is always used in conjunction with the memory register "M".
  - The BC and DE register pairs can be used to load data into the Accumultor using indirect addressing.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, this gives rise to something called the indirect addressing mode. So, without so, we it is actually use the data from the memory directly, and we do not do not load it into the CPU registers; so for doing the addition and all this operation. So, it is required that there are 2 operands we have to give we have to get 2 operands these one possibilities are in the CPU registers, but if you do not do that. So, you can get the second operand from the memory as well. So, this is called indirect addressing. So, this uses data in a register pair as a 16-bit address to identify the memory location being accessed.

So, the if you are using H L pair then it is with registered M. So, it is when whenever you are telling this register as M. So, it is the H L pair, and if it is B C or D E pair then you have to use the you have to load the value into the accumulator, and then do the arithmetic operation.

(Refer Slide Time: 19:09)



Arithmetic Operations

- Addition (ADD, ADI):
  - Any 8-bit number.
  - The contents of a register.
  - The contents of a memory location.
  - Can be added to the contents of the accumulator and the result is stored in the accumulator.
- Subtraction (SUB, SUI):
  - Any 8-bit number
  - The contents of a register
  - The contents of a memory location
  - Can be subtracted **from** the contents of the accumulator. The re stored in the accumulator.

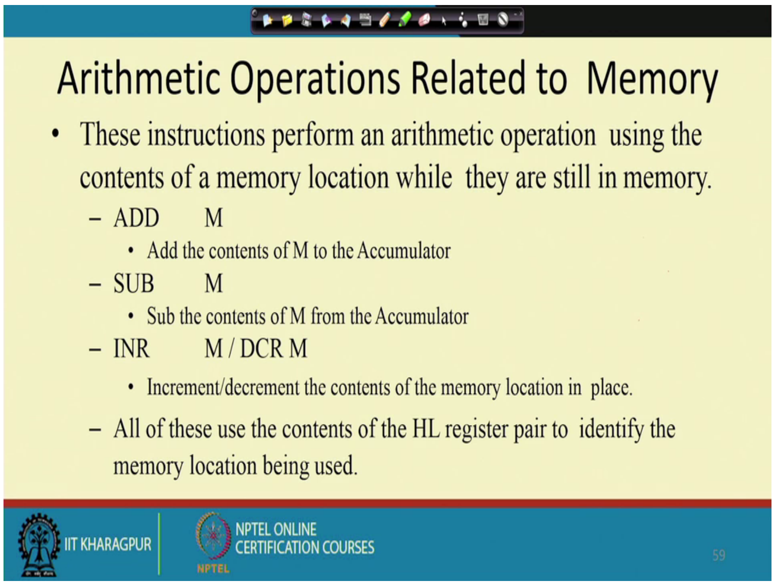
$ADD\ B \Rightarrow A \leftarrow A+B$

$ADI\ 5 \Rightarrow A \leftarrow A+5$

So, these addition operations like say add or add immediate. So, any 8-bit number can be added. So, the content of so, I have this add instruction. So, naturally it operates on 8-bit data. And one operand is the accumulator the other, other operand is the other operand is the operand that we have specified. For example, you can say like say add you can say like say ADD B. So, which means that the a register will get A plus B. So, this is possible or you can say add immediate 5. So, that means, the a register will get a plus 5. So, this way we can have this immediate operand, and we can have this addition of numbers.

So, other possibility that we have is that we can have the subtraction operation. So, the subtraction operation is similar to addition, but it will be subtracting the number, from one from the other the second operand from the first opponent. And otherwise it is same as the thing. So, it can be subtraction the accumulator and the result is stored in the accumulator. So, we both add and sub instruction the destination is the accumulator. So, after that the second source operand, may be something else, but the destination is always the accumulator.

(Refer Slide Time: 20:42)



Arithmetic Operations Related to Memory

- These instructions perform an arithmetic operation using the contents of a memory location while they are still in memory.
  - ADD M
    - Add the contents of M to the Accumulator
  - SUB M
    - Sub the contents of M from the Accumulator
  - INR M / DCR M
    - Increment/decrement the contents of the memory location in place.
  - All of these use the contents of the HL register pair to identify the memory location being used.

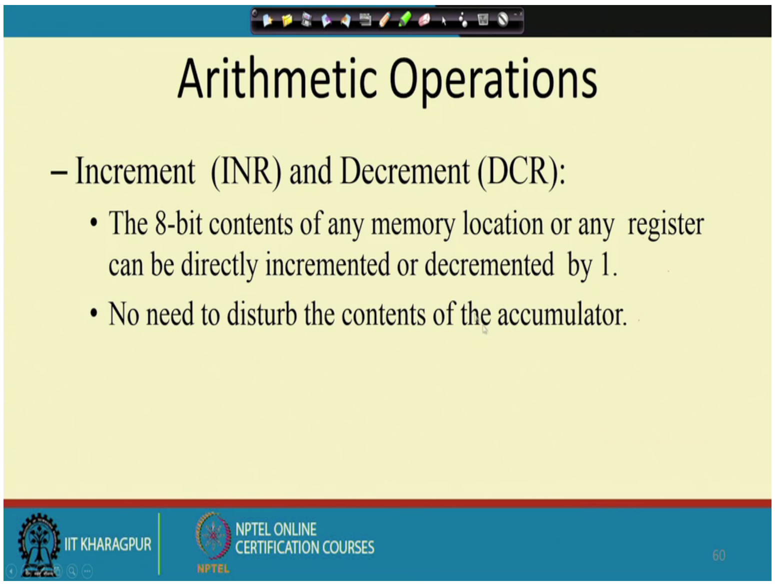
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

59

So, if you are trying to use the memory as one of the operand, then a first of all you need to ensure that the H L pair points to the correct memory location which will be used in the instruction for the addition subtraction etcetera. So, you can have this add M instruction. So, add M what it will do it will add the content of memory location pointed to by H L register with the content of accumulator, and the result will be stored in the accumulator.

Similarly, we have sub M which will subtract the value of memory location pointed to by H L from a register, and the value will be stored in the accumulator. Similarly, your INR M increment M decrement M. So, these are also possible. So, H L pair will be used as the indirect address.

(Refer Slide Time: 21:35)



Arithmetic Operations

– Increment (INR) and Decrement (DCR):

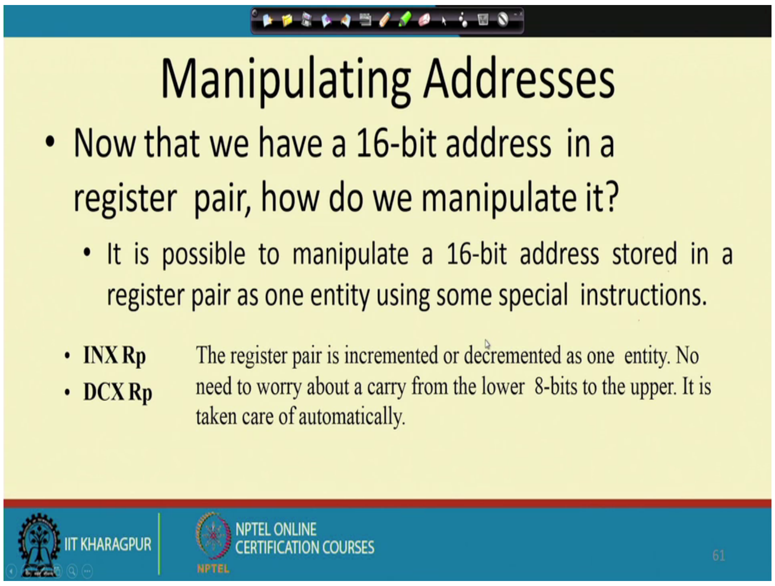
- The 8-bit contents of any memory location or any register can be directly incremented or decremented by 1.
- No need to disturb the contents of the accumulator.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL
60

So, other operations other arithmetic operation like we have to have this increment decrement operation. So, you can do it directly with respect to some register or memory location. And we do not need to disturb the accumulator. So, other operation like add operation sub operation what is happening is that the result is stored in the accumulator. And nows now suppose I just need to increment B. So, if I if this is not possible directly then what will be required is that B content will come to the accumulator, then it will be incremented and the value will be stored back to the B register. So, that will that will destroy the previous content of a register which is not always advisable for a program that will make the control flow difficult.

So, what is better is that for this increment decrement operation we do not involve the accumulator directly until and unless it is explicitly specified that INR A where I want to increment the a register itself, but I can if I have got this register specified directly like INR B INR C dcr D, like that then this should be done directly without affecting the accumulator.

(Refer Slide Time: 22:56)



Manipulating Addresses

- Now that we have a 16-bit address in a register pair, how do we manipulate it?
  - It is possible to manipulate a 16-bit address stored in a register pair as one entity using some special instructions.

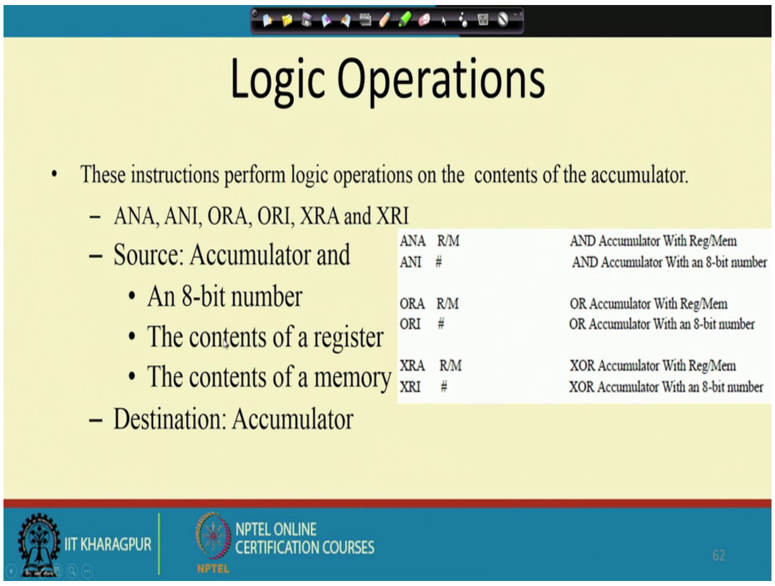
- **INX Rp**
- **DCX Rp**

The register pair is incremented or decremented as one entity. No need to worry about a carry from the lower 8-bits to the upper. It is taken care of automatically.

Now how to manipulate this address part, like we have got this register pair to hold this address that is fine, but how can we change it? Can we change the content of this register pair? So, this is possible by do using this instruction INX. So, when it is x it is extended. So, extended means it is 16 bit. And you remember that 8085 is an 8-bit processor, because it generally operates on 8-bit data. So, it is not that it cannot operate on 16-bit data, but internal implementation wise it is all 8 bit. So, that is why this is an 8 bit processor.

So, this is register pair INX Rp. So, this register pair will be incremented by 1 and this DCX Rp will decrement this register pair by 1 and we do not need to worry like this Rp register pairs suppose this BC. So, when this C value becomes equal to 0, and we decrement it further, then a carry is being generated and it is automatically taken care of. So, we do not need to if this INX and DCX instructions were not there. Then you need to write an elaborate routine by which you do you can do this 16-bit increment decrement operations, but that is not necessary because these instructions are there; this INX and DCX operation.

(Refer Slide Time: 24:22)



Logic Operations

- These instructions perform logic operations on the contents of the accumulator.
  - ANA, ANI, ORA, ORI, XRA and XRI
  - Source: Accumulator and
    - An 8-bit number
    - The contents of a register
    - The contents of a memory
  - Destination: Accumulator

ANA	R/M	AND Accumulator With Reg/Mem
ANI	#	AND Accumulator With an 8-bit number
ORA	R/M	OR Accumulator With Reg/Mem
ORI	#	OR Accumulator With an 8-bit number
XRA	R/M	XOR Accumulator With Reg/Mem
XRI	#	XOR Accumulator With an 8-bit number

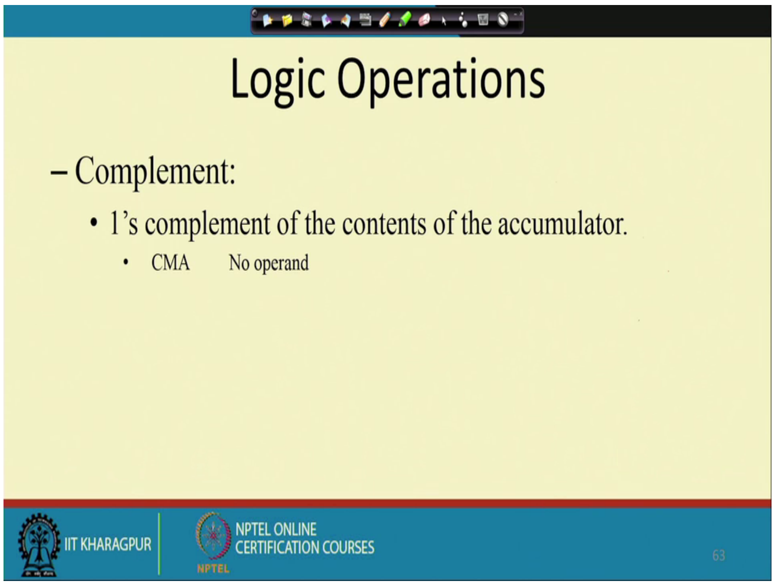
IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

62

Now, there are logical operations. So, after this arithmetic operations, we have got logic operations, and we have got this instructions like; and so, this ANA then ANI. So, ANA is the logical and operation ANI is and immediate. ORA is or accumulator. ORI is or accumulator with some immediate value.

Then we have got XORA for the xor operation and XORI for again xor operation with some immediate operand. So, we have got this source as accumulator and we have got 8-bit number. So, that is the content of a register or contact of memory location as the second operand, and the destination is always the accumulator. So, otherwise they remain same.

(Refer Slide Time: 25:11)



Logic Operations

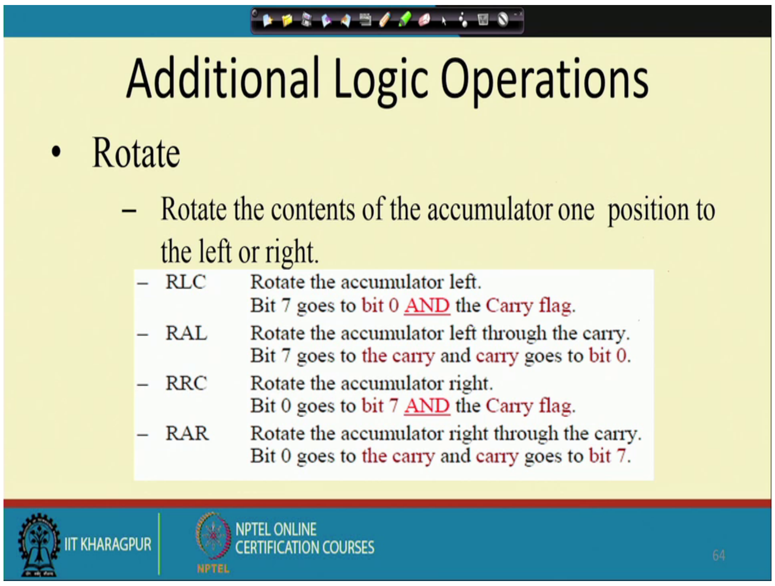
– Complement:
  - 1's complement of the contents of the accumulator.
    - CMA No operand

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
63

So, another operand the operation that is there in in logic category there is a compliment; so this complement operation. So, this is one's compliment of the content of the accumulator. So, it is it will just invert all the bits of the accumulator. So, as such so, no other operand need to be specified. So, this is 0 operand instruction because the operand is implicit.

So, this is CMA complement accumulator. So, it is a ones complement it is not 2s complement. So, it should not take that if I do this I will get a negative of a number. So, it is just B twice in a logical operation that is why it is not an arithmetic operation the logical operation. So, it will be in complementing the bits of individual of the accumulator register other logic operations there.

(Refer Slide Time: 26:01)



Additional Logic Operations

- Rotate
  - Rotate the contents of the accumulator one position to the left or right.

–	RLC	Rotate the accumulator left. Bit 7 goes to bit 0 AND the Carry flag.
–	RAL	Rotate the accumulator left through the carry. Bit 7 goes to the carry and carry goes to bit 0.
–	RRC	Rotate the accumulator right. Bit 0 goes to bit 7 AND the Carry flag.
–	RAR	Rotate the accumulator right through the carry. Bit 0 goes to the carry and carry goes to bit 7.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

64

So, we have got the rotate instruction. So, it will rotate the content of accumulator one position to the left or right. So, we have got 2 categories one is rotate left another is rotate; RL and RR. Now this when we do this rotate left then this rotate the accumulator left bit 7 will of the accumulator will go to bit 0 and the carry flag. So, bit 7 will go to bit 0 as well as the carry flag. So, it is RLC stands for rotate left with carry, and then this one this RAL. So, it will also rotate the accumulator left through the carry. So, it is with carry and this it is through the carry.

So, in this case the bit 7 will go to the carry, and the carry will go to bit 0. So, this is RLC and RAL. Similarly, we have got rrc and RAR rrc is same as RLC, but now it is right. So, it is rotate the accumulator right bit 0 goes to bit 7 and the carry flag, and RAR it will rotate the accumulator through the carry accumulator, through the carry bit 0 goes to the carry and the carry goes to bit 7.