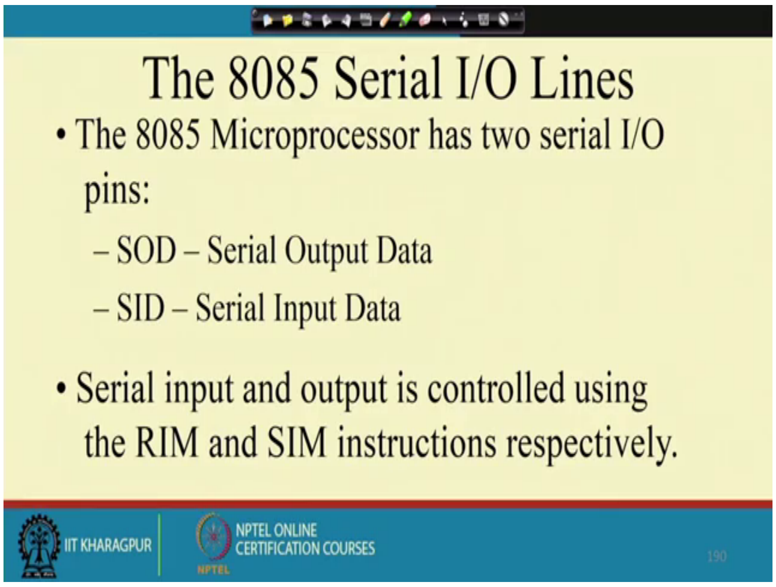
Microprocessors and Microcontrollers

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

Lecture – 21
8085 Microprocessors (Contd.)

In 8085 this serial input output operations are performed by 2 pins.

(Refer Slide Time: 00:21)

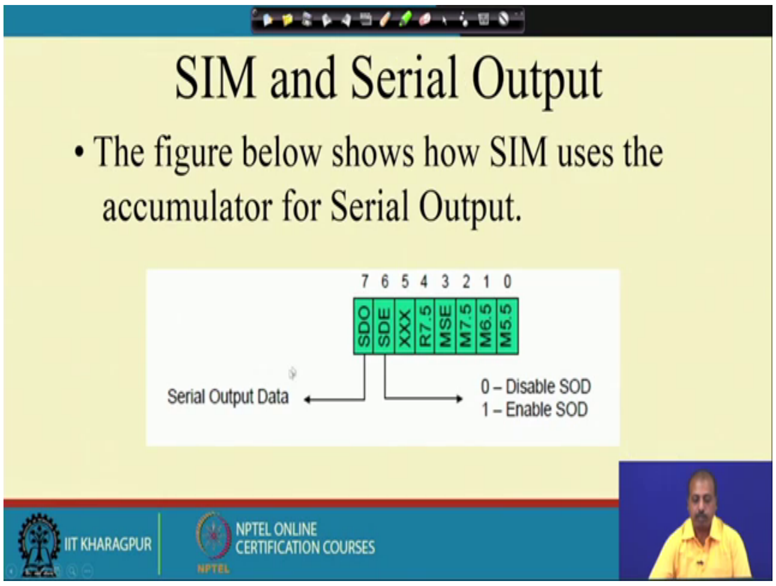


The 8085 Serial I/O Lines

- The 8085 Microprocessor has two serial I/O pins:
  - SOD – Serial Output Data
  - SID – Serial Input Data
- Serial input and output is controlled using the RIM and SIM instructions respectively.

One is known as the serial output data or SOD other is the serial input data or SID. So, these 2 are for serial input and output, and this as far as the operation is concerned operation is controlled by this instructions RIM and SIM, read interrupt mask and set interval mask. As we have seen previously that this RIM and SIM instruction. So, they have got one bit configured in the accumulated that will be acting as the serial input and output.

(Refer Slide Time: 00:53)


SIM and Serial Output

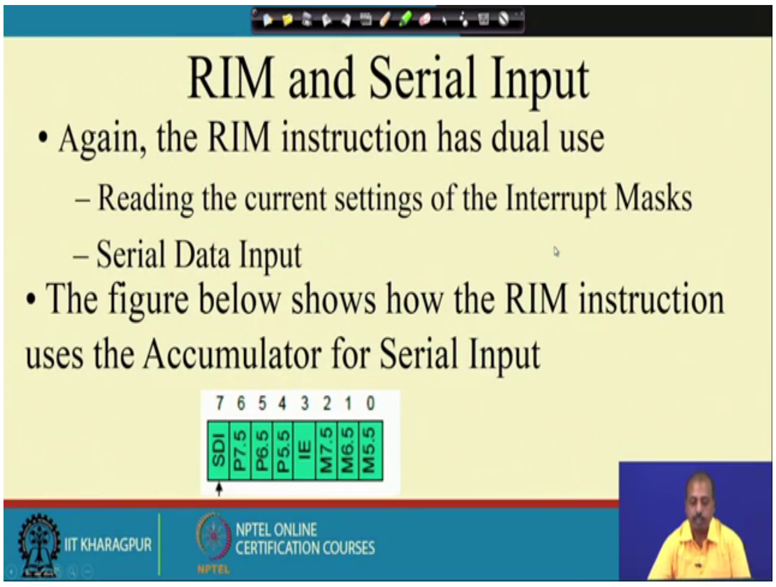
- The figure below shows how SIM uses the accumulator for Serial Output.

So, this is how is it takes place, first of all we will discuss about the SIM instruction and the serial output part. So, this shows that how this can be done.

So, this diagram if you remember. So, we have got this accumulator setting where this bit 0 to bit 4. So, there are for this interrupt masking and this R 7.5 bit, 5 bit 5 is not used and this bit number 6 is for disabling and enabling serial output data. So, SDE serial data enable you can say. So, if it is 0 then this serial output data, this is disabled and if it is 1 then this is enabled 0.5 and s or SDO bit. So, this is for the serial output data. So, that will hold the data that should be sent to the outside through this pin. So, if you want to transmit some bit serially.

So, we can put the bit in this parti bit number 7 of the accumulator and enable this SDE bit and then execute the SIM instruction so that the bit that is told it will be sent through this line. Next you put another bit here and sitting that SDE in the enable mode. So, again the next bit we will go through this serial. So, this is how this SIM can be used for serial transmission of data.

(Refer Slide Time: 02:16).



RIM and Serial Input

- Again, the RIM instruction has dual use
  - Reading the current settings of the Interrupt Masks
  - Serial Data Input
- The figure below shows how the RIM instruction uses the Accumulator for Serial Input

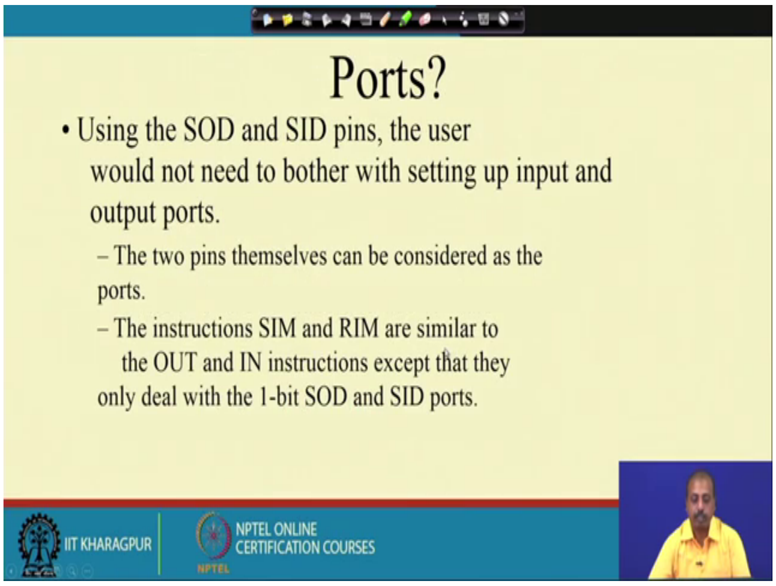
7	6	5	4	3	2	1	0
SDI	P7.5	P6.5	P5.5	IE	M7.5	M6.5	M5.5

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

Similarly, you can for serial input parts, we can take care take help of the RIM instruction.

So, RIM accumulator configuration is like this. So, rest of the bits we have discussed while talking about this we talking about the interrupts and this bit number 7 is the SDI serial data input. So, this particular bit can be used. So, when you are using this RIM instruction, this rest of the bits that is bit number 0 to 6 they are set according to the interrupt mass settings and all that, but this SDI bits. So, this is coming from the SID pin of the 8 0 8 5 processor, and this bit will be having the value of the SID pin. So, whatever is coming serial on the SID pin. So, that will be put into this particular bit. So, we can use this RIM and SIM instruction for doing this serial data input and output operation.

(Refer Slide Time: 03:14)


Ports?

- Using the SOD and SID pins, the user would not need to bother with setting up input and output ports.
  - The two pins themselves can be considered as the ports.
  - The instructions SIM and RIM are similar to the OUT and IN instructions except that they only deal with the 1-bit SOD and SID ports.

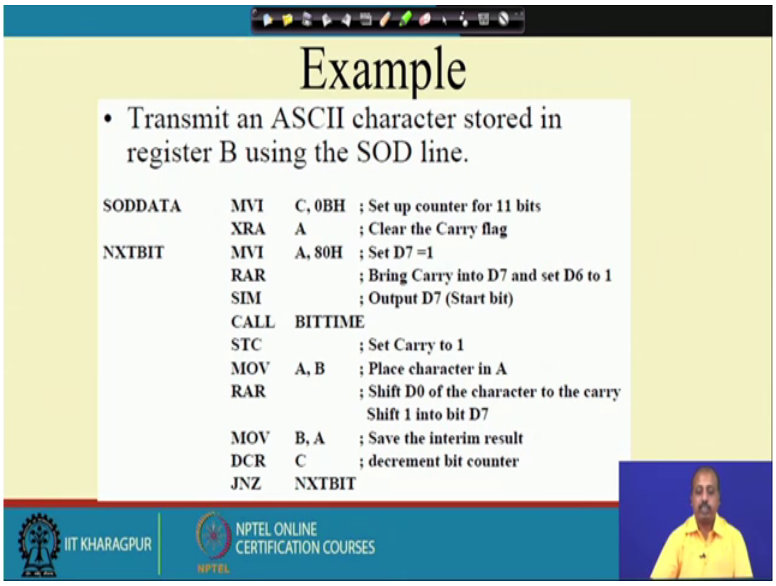
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

What about port selection. So, since this SID and SOD, they are single bit pin and we can we do not have to have different devices, number of devices connected to it and all the so that concept is not there.

So, you do not need to bother about the setting up of input and output port. So, there SID is for input and SOD for output. So, they may be connected to the same device and the (Refer Time: 03:45) it is an device is responsibility to figure out whether it is a input operation or output operation. So, that way, this 2 pins SID and SOD. So, they are by themselves they are the ports for serial ports. Now this is SID this SID, SIM and RIM instruction. So, they are similar to this out and in instruction. So, we will see this out and in instructions later. So, this out and in instructions they are for parallel output. So, you will see that.

So, for parallel output for 8 bit outputs we can use out instruction for 8 bit input we can use in instruction. So, only difference is that in case of out it is 8 bit and in case of SIM it is one bit, in case of in it is again 8 bit input, in case of RIM it is single bit input. So, that way this SOD and SID pins are used as ports.

(Refer Slide Time: 04:36)



Example

- Transmit an ASCII character stored in register B using the SOD line.

```
SODDATA   MVI   C, 0BH  ; Set up counter for 11 bits
          XRA   A       ; Clear the Carry flag
NXTBIT    MVI   A, 80H  ; Set D7 =1
          RAR           ; Bring Carry into D7 and set D6 to 1
          SIM           ; Output D7 (Start bit)
          CALL  BITTIME
          STC           ; Set Carry to 1
          MOV   A, B    ; Place character in A
          RAR           ; Shift D0 of the character to the carry
                          Shift 1 into bit D7
          MOV   B, A    ; Save the interim result
          DCR   C       ; decrement bit counter
          JNZ   NXTBIT
```

So, this is an example that talks about how this serial communication can take place. So, let us take an example of transmitting one ASCII character, store in the register B using this SOD line. So, we have to first, how many bits to be transmitted all together. So, 1start bit 2 stop bits and 8 bits of data. So, total 11 bits are to be transmitted ok.

So, we can keep a counter in the sea register for the 11 bits. So, in MVI C comma 0 BH. So, that will set the set the C register to the decimal value 11, then the next introduction XRA A. So, this is x or accumulator with accumulator register,r if you XRA the accumulator with itself. So, what is the continent will become 0. So, the accumulator contain will becomes 0 and carry flag will also be cleared. Then we have to move this we move the pattern 8 0 to this A register that will set that D 7 bit to 1. So, D 7 bit.

So, then we rotate right. So, bring the carry into D 7 and the that this bit shifts and D 6 becomes 1 then we excite the SIM introduction. So, this actually output the start bit. So, if you if I just go back and see that seem. So, for putting the start bit. So, this bit should be 0 because start bit is 0, and this line should be 1. So, this is done by the first few instructions that we have discussed. So, here this 8 setting this r (Refer Time: 06:24) doing this RAR rotate accumulator right. So, this will be bringing the carry flag from the carry bit from the carry flag into the D 7 bit, and the D 7 bit will get shifted to d 6 bit for D 7 bit will act as that start bit and D 6 bit will be the setting of that SDE the serial data enable and in the SIM instructions.

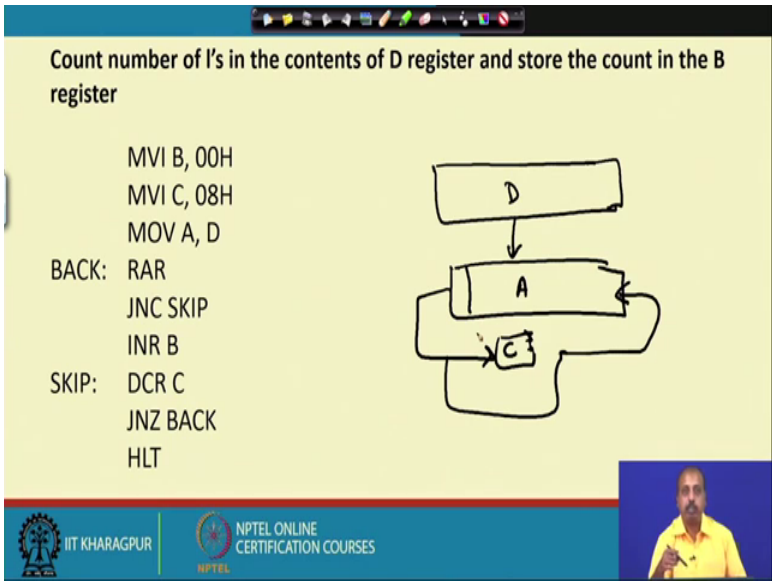
So, this D 7 bit 0 will go to the output. So, that will be the start bit then we have to wait for some bit time. So, you have some delay routine for the bit time. So, which is not shown here explicitly. So, you can write a small delay routine by you are finding out what is the bit time for the type of crystal that is connected to 8 5 processor it is the frequency and then you can find out what should be the timing; and you have to have this borate sitting between the device and the processor and all that. So, though based on that you can find out what is the bit time.

So, you can put a delay routine which is equivalent to this bit time delay, then we set carry to one and then move A comma B. So, this will place the charter in that we had in the B register into A register, then we do a RAR. So, that way this D 0 that we have the first charter that is there in the accumulator. So, that the D 0 bit, that that will go to carry and then the carry was set to one as you see here. So, that carry will become that carry will now get shifted to D 7. This intermediary result is save in the B register because after that the content will be changed when we execute this SIM and all that. So, this is set to value is set into the B register.

So, at the end of it we have got the bit that we want to transmit the next bit that we want to transmit is available in the carry, and the D 7 bit is one by this set carry and then when we do this RAR. So, this carry bit comes to the D 7 and then this the bit that we want to transmit comes to the carry after this RAR instruction, than we save this interim value and then we decrement C. So, we will decrement C. So, one has been transmitted the start bit has been transmitted. So, it is decrement by one and jump on not zero. So, come to the next bit. So, in the next bit you see that again the D 7 bit will be set to one and it rotate right.

So, the D 7 will go to the d 6 bit enabling the SDE, and this SIM instruction will transmit that. So, carry bit will be coming to the D 7 bit and carry bit is actually holding the least significant bit that we have in the charter to be transmitted now. So, that will be coming to this thought this SIM. So, that will be outputted to the serial output data. So, this way we can excite this program, and by this individual characters will be transmitted one after the other and of course, what is not shown here is the finally, we need to transmit to stop bit. So, that is not shown here explicitly. So, after transmitting this 8. So, we have to transmit the carry bits. So, that part of the program is not shown here. So, you can add that to the bottom of the program.

(Refer Slide Time: 09:54)



Count number of I's in the contents of D register and store the count in the B register

```
        MVI B, 00H
        MVI C, 08H
        MOV A, D
BACK:   RAR
        JNC SKIP
        INR B
SKIP:   DCR C
        JNZ BACK
        HLT
```

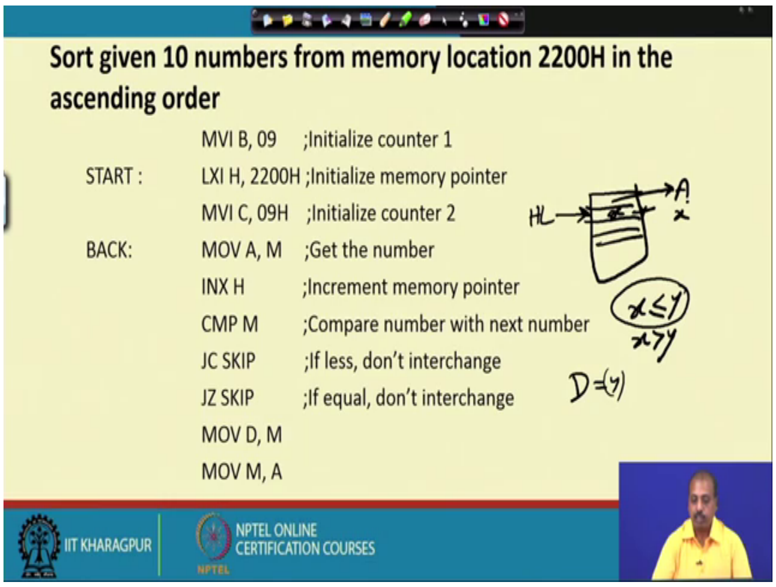
Next we will look into some more examples of this 80 85 programming. So, so we have seen many statement. So, we will use those statements to write some sample program. So, you can just practice them or you can write some similar such programs to get confidence about the programming features of 80 85. The first program that we consider is to count number of ones in the content of D register, and store the count in the B register. So, how do we do this thing? So, in the B register we store 0 0, that is the initial count of number of ones that we have in the B register in D register that count is initialized to 0 and MVI C comma 0 8 H. So, this C register is holding like how many times I need to check the policy that we will follow is basically we will check the carry bit. So, policy that follow is if this is the content of this B register, then what we do? We transfer this content to the A register first and then in the A register we do some rotation; rotation through carry ok.

So, through this carry bit we rotate it, and if the carry bit is set if the carry bit becomes set by this fashion sorry it is not from here it actually goes from this point. So, the same the bit the most significant bit it goes to the carry as well as the least significant position, now if this when this bit is rotated. So, if the carry become set; that means, the bit was 1, and if the carry is not set then the bit is 0. So, we do 8 such rotations, and in whichever if the in the rotation if the bit becomes 0, then we do not increase the B register, but if the bit is 1 the carry bit is 1 we increment the register so that we can get the value there ok.

We can get you can count that one. So, this is done here in the program, you see that this rotate right rotate accumulator right and then jump on (Refer Time: 12:31) to skip. So, in that case if the carry is not set, then we do not count it as the bit that we have as a 1. So, B register is not implemented. So, then you decrement the C value, because we have done one rotation and jump on not 0 back. So, you will go back to this point again rotate the accumulator and we will do this thing. So, this way after this look competes. So, in the C register in the B register will have the number of ones that we have in the original in the D register. So, that will be available in the B registered now.

So, if we look into another program.

(Refer Slide Time: 13:13)

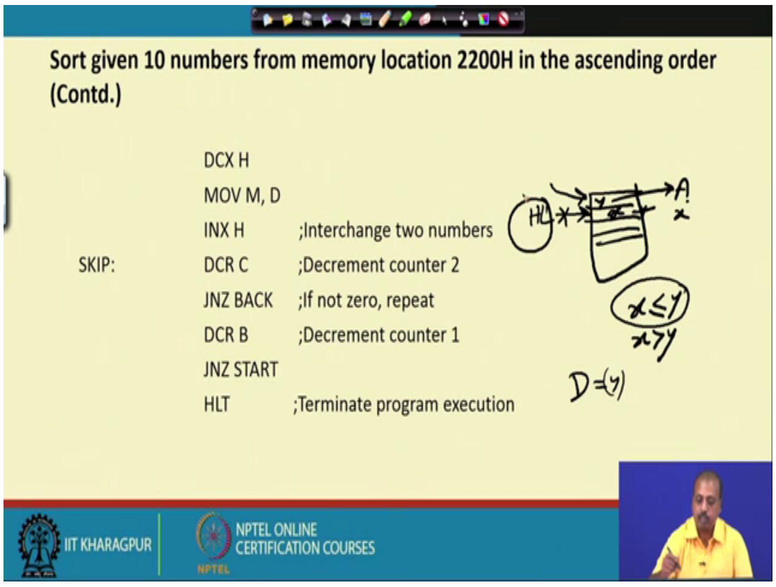


Sort given 10 numbers from memory location 2200H in the ascending order

```
          MVI B, 09    ;Initialize counter 1
START :   LXI H, 2200H ;Initialize memory pointer
          MVI C, 09H   ;Initialize counter 2
BACK:     MOV A, M     ;Get the number
          INX H        ;Increment memory pointer
          CMP M        ;Compare number with next number
          JC SKIP      ;If less, don't interchange
          JZ SKIP      ;If equal, don't interchange
          MOV D, M
          MOV M, A
```

So, we want to sort 10 numbers given stored from stored in memory location memory location 2 2 0 0 onwards and we want to solve them in ascending order.

(Refer Slide Time: 13:20)



Sort given 10 numbers from memory location 2200H in the ascending order (Contd.)

```
            DCX H
            MOV M, D
            INX H       ;Interchange two numbers
SKIP:       DCR C       ;Decrement counter 2
            JNZ BACK    ;If not zero, repeat
            DCR B       ;Decrement counter 1
            JNZ START
            HLT         ;Terminate program execution
```

So, sorting we use the simple bubble sort the type of algorithms. So, we just compare 2 successive numbers, and if they are out of order which change the order of those 2 numbers and if they are in proper order, then we do not alter them and this process is repeated for the all possible pairs of numbers, this is the algorithm that we fallow. So, we initialize this B registered to the value 9, because the first index it can go from1 to 9. So, and the second index will be going from one 2 to 10 so that way we do it. So, LXI H 2 2 0 0 H. So, that will be stor that is the initialized memory pointer. So, it will be initialized and then we move this C comma 0 9 H, this we initialized the counter 2 the other counter is also initialize a move A comma M. So, we will get the first number into the accumulator then the INX H it will increment the memory pointer. So, we have come to the next number. So, in the. So, we have got the numbers stored in this fashion. So, if this is the array, where we have got the numbers stored.

So, the first number is available in the A register now then the H L pair is incremented, and we compared. So, compare M. So, this HL pair is pointing to this address now. So, we compare the content of this location with the content of the accumulator register. So, if the content of the accumulator is less than content of this memory location or they are same suppose this value is say x and this value is say y. So, the possibilities are x is less than y or x is less or equal y other possibility is x is greater than y.

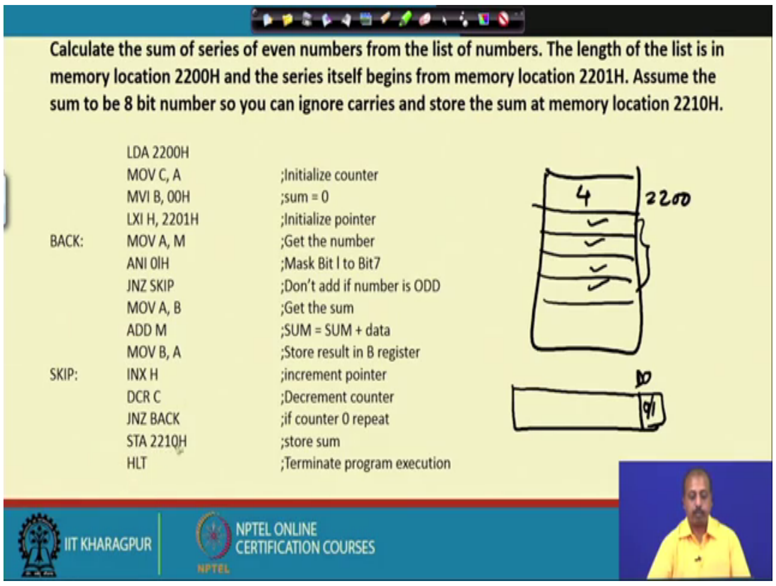
So, if x is less or equal y we do not need to do anything, they are already in the proper order if x is greater than y in that case you need to change the order. So, on jump on carry and jump on 0. So, these 2 cases we do not do anything. So, we do not know if they are less do not interchange, if they equal then also do not interchange otherwise we have to interchange them. So, what do we do? So, move D comma M. So, this instruction. So, this will move the content of this memory location on to the D register. So, the D register now has the value D register has got the value y and then it says move M comma A.

So, this location it was containing y previously now this value becomes equal to equal to x move M comma A this will make it equal to x. Next now we need to value store the value of y on to the previous memory location. So, for that purpose we decrement H. So, H L pair now sorry H L pair now goes back to the point that we have here. So, H L pair goes there one position back by DCX H and then move M comma D. So, now, D register is having the value y so that will be moved here.

So, this location gets the value y. So, as a result this x and y values have got interchanged and then we again take that H L pair back to this position. So, this INX H. So, this will take the H L pair back to the next position with decrement C. So, this one iteration of the inner loop is over jump (Refer Time: 17:37) not 0 to back, it will take us to back and if it is, so that is back is here. So, we get the next number. So, that way we can we will go back to that position and if it is otherwise. So, if it is 0, we have completed one iteration one complete iteration of this counter 2, we decrement this D register and then if it is b registered is not zero. So, we go back start the next iteration of the outer loop. we start the next iteration outer loops stating with this LXI H. So, this process is repeated for 9 times. So, as a result the entire area will get sorted.

So, this way we do this algorithm of sorting 10 numbers in ascending order.

(Refer Slide Time: 18:42)



Calculate the sum of series of even numbers from the list of numbers. The length of the list is in memory location 2200H and the series itself begins from memory location 2201H. Assume the sum to be 8 bit number so you can ignore carries and store the sum at memory location 2210H.

```
        LDA 2200H
        MOV C, A        ;Initialize counter
        MVI B, 00H      ;sum = 0
        LXI H, 2201H    ;Initialize pointer
BACK:   MOV A, M        ;Get the number
        ANI 01H         ;Mask Bit l to Bit7
        JNZ SKIP        ;Don't add if number is ODD
        MOV A, B        ;Get the sum
        ADD M           ;SUM = SUM + data
        MOV B, A        ;Store result in B register
SKIP:   INX H           ;increment pointer
        DCR C           ;Decrement counter
        JNZ BACK        ;if counter 0 repeat
        STA 2210H       ;store sum
        HLT             ;Terminate program execution
```

next we will look into another program that calculates the sum of a series of even numbers from the list of numbers. So, in the list we have got both even numbers and odd numbers, and the numbers are stored from the location 2 2 0 0, and that is and there how many numbers are there in the set. So, that is stored in the location 2 2 0 0 that is a length of the list, and the least actually starts at location 2 2 0 1.

So, at that location 2 2 0 0 we have got the number of numbers. So, if this the memory. So, at 2 2 0 0 we have got the number of number. So, if I have got only say 4 entries, then this value will be 4 and here I have got the actual numbers. This individual numbers maybe even or odd. So, the program is to sum up this even numbers, and there is an assumption that the sum you can be that is why the total output sum is 8 bit wide only. So, there is there will be no overflow after we have added the all the even number. So, we can ignore that carries and store the sum at memory location 2 2 1 0 finally, the sum has to be stored at location 2 2 1 0.

So, how we do this program? So, we first load accumulator 2 2 0 0 H that will have the number of numbers in the A resistor, then move C comma A it will move the count value to the C register then MVI B 0 0 H. So, the sum (Refer Time: 20:18) B is holding intermediary sum so that is initialize to 0, LXI H 2 2 0 1 H. So, this will initialized the pointer to the first location of the array and then move A comma M. So, this will get the first number that we have, and it will get it into the A registered then a 9 0 1 H. So, we need to check whether the number is even or odd ok.

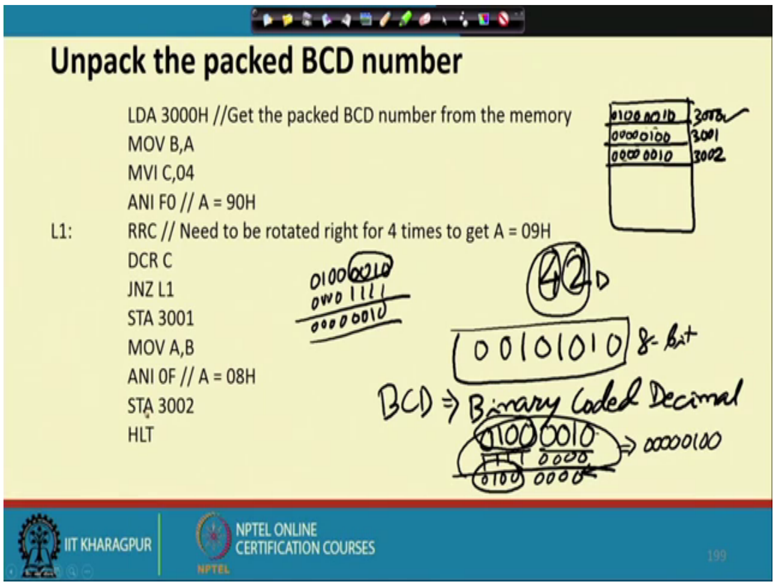
So, for that purpose, how to do it? So, we need to check if the number is even, then this the least significant bit that is a bit D 0 will be 0, if the number is even. On the other hand if the number is odd, then this bit will be 1. So, just exactly what is checked here. So, we see whether the we do a masking of that number with 0 1 H. So, only the least significant bit is 1 and as a result if there if the result becomes 0 of this due to this ending the result becomes 0; that means, the number is an or even number.

So, in that case sorry if the number is not zero then it is an odd number. So, in that case we do not need to edit, but if the number is after this ending if the value becomes 0; that means, the number is even. So, we need to add it. So, we first moved this B register content to A register because B was holding the intermediary sum so that is moved into the register then add M.

So, add M will add the contain of the memory location pointed to by this H L pair with the A register, and the value is in the A register and then the value is again move to the B register the intermediary res result is stored back onto the B register. Then this H registered is incremented this C value C registered is decremented, because we have already processed one number. So, this there C registers is decremented and as long as this C register content is not 0 will go back to this jump on not 0 back. So, it will come back here now it will access the next memory location and edit.

So, once this C register value becomes 0; that means, we have added all the even numbers. So, the value has to be stored at location 2 2 1 0 H. So, that is done here. So the store accumulator 2 2 1 0 H. So, this will be storing the number at location 2 2 1 0 H ok.

(Refer Slide Time: 22:56)



Unpack the packed BCD number

```
        LDA 3000H //Get the packed BCD number from the memory
        MOV B,A
        MVI C,04
        ANI F0 // A = 90H
L1:     RRC // Need to be rotated right for 4 times to get A = 09H
        DCR C
        JNZ L1
        STA 3001
        MOV A,B
        ANI 0F // A = 08H
        STA 3002
        HLT
```

Next we look into another program that will unpacked the packed BCD numbers. So, packed BCD numbers are like this, like when we are stored is when we are storing say number say numbers 42, 42 in decimal. So, 42 in decimal, if you want to store it. So, if you when you are storing in the computer. So, this 42 will be converted into bit pattern like. So, this is. So, this is 1 2 3 4 5 6 7 8. So, this is the 42 representation that we have in the binary number system.

So, if you are storing the number in a computer system then what will happen? So, this will be stored as an 8 bit pattern. So, there is another number system which is known as BCD which stands for binary coded decimal. So, here what is done is that this individual digits of this decimal number system they are coded into binary. So, we have got these 2 number 2 digits here 4 and 2. So, they will be coded the individually and they will be stored in the into 4 bits of a byte. So, what will happen is that this number will be stored as first the digit is 4. So, this will be stored as 0 1 0 0, and the next digit is 2. So, this is 0 0 1 0.

So, this is how the number will be stored. So, first 4 bit and the next 4 bit. So, that is the BCD coding of the number 42. Now if the number is stored like this and then suppose the number is stored at memory location 3000. So, at memory location 3000 we have the number 0 1 0 0 0 0 1 0. So, this is the 4 bit pattern 8 bit pattern that we have now what the program will do. So, it will break this number into 2 8 bit numbers and store results at 3 0 0 1 and 3 0 0 2 at 3 0 0 1. So, it will be storing the first digit that is it will be storing this 0 1 0 0.

And the next location and it will store the number 2 that is the next digit 2. So, that is 0 0 1 0 and rest of the bits are 0. So, this when I say it is 0 0 1 0. So, this most significant bits are all 0. So, the actual number that is stored is 0 0 0 0 0 1 0 0 0 similarly at the next location the actual number that is stored is 0 0 0 0 0 0 1 0. So, this is the thing that we want.

So, I have got at location 3000 in the number like this. So, I have to convert it into 2 digits separately and stored them in an unpack form. So, this is called packed BCD form the 3000 local packed BCD formed, because we are packing 2 digits of BCD number into 1 bit. So, each digit takes 4 bit. So, I can put 2 such digits in a location that is why it is called packed number. So, you can unpack it.

So, how to do this? So, first we get the number into the accumulator. So, this LDA 3000 H. So, it will get the packed BCD number from memory in to the accumulator, then we store the number in the B register. So, move B comma A. So, this will save this value of A registered into B registered; then this C is made equal to 4 and then we and immediate with F 0. So, if we and immediately with F 0; that means, this F 0 H.

So, what you what happens is that? That from this number I want to get only this part will be the; this part I want to convert to 0. So, if I end with F 0 that is ending with say 1 1 1 1 0 0 0 0. So, the result is 0 1 0 0 0 0 0 0 after this ending. So, and then we do a rotate right through carry. So, this will be rotated 4 times and we decrement C. So, this rotation. So, C is initialize to 4. So, this loop will be repeated 4 times. So, JNZ L1. So, this will be repeated 4 times as a result rotate right.

So, if we rotate right then this part this result. So, they will be coming to the they will be rotated and coming to this side. So, after this rotation the result will be 0 0 0 0 0 1 0 0. So, that is what we want to store now. So, the accumulator content is like this. So, that value will be stored at location 3001 now we do again. So, the one part is done now the next digit that we have. So, we move the content of registered B into register A and again and now we end immediately 0 HX. So, if we (Refer Time: 28:35) with this content with the content 0 1 0 0 0 0 1 0. So, if we end with 0 F X 0 0 0 0 1 1 1 1. So, we will get this pat will become all zero. So, this will become 0 0 1 0. So, you will get back this part here and all other things will be 0.

So, now here we do not need to rotate anything because this is the pattern that I want to store. So, this will be stored in the location 3002. So, this will be the value will be stored in the location 2 so that will be stored there. So, then the this holt instruction this is for terminating the program. So, halt instruction will stop the processor as we know. So, in your case. So, you may have some other loops etcetera here to make the program (Refer Time: 29:31) or you may some cases we transfer the control to the monitor program of the heat of the microprocessor heat on which you are working.

So, that we can check the content of different registers and see whether the program has been done correctly or not.