

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

Lecture – 38
8051 Programming Examples (Contd.)

Next we will look into another program that computes the LCM of two numbers least common multiple of two numbers.

(Refer Slide Time: 00:20)

The slide is titled "Example" and contains the text: "Write a program to find LCM of two numbers stored at memory addresses 6000H and 6001H, with the smaller number at 6000H." Below the text, there are handwritten notes: "x, y" with arrows pointing to "2x" and "3x", and a small diagram of "x ÷ y" with a checkmark. In the bottom right corner, there is a small video inset showing a man in a yellow shirt. The slide footer includes the IIT Kharagpur logo and the text "NPTEL ONLINE CERTIFICATION COURSES".

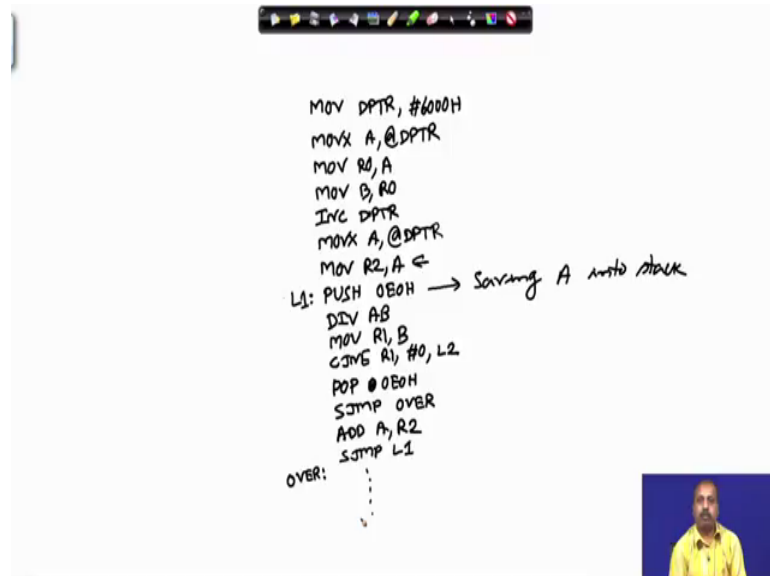
Stored at memory addresses 6000 and 6001H and the assumption that the smaller number is in 6000H. So, see you see if the numbers are say x and y , so if the numbers are say x and y . Then for finding the LCM, we have to divide x by y you have to divide x by y and see whether this y divides x whether y divides x or not.

If y divides x then x is the LCM, if not we have to take the next power of x , x is the larger number y is the smaller number. So, I have to take the next, so we have to take the next multiple of x . So, it is a $2x$ and then again do the same exercise, so I have to see whether this $2x$ is divisible by y or not if it is, so then $2x$ is the LCM. So, then if it is not then I have to try with $3x$.

So, this way I have to just go on, but we increasing the multiple of this x $1x$ $2x$ $3x$ $4x$ $5x$ like that, and whichever at whichever multiple I have got the got this division without

reminder. So, that is the LCM of the two numbers. So, this principle will be used for writing the LCM routine.

(Refer Slide Time: 01:52)



```
MOV DPTR, #6000H
MOVA A, @DPTR
MOV R0, A
MOV B, R0
INC DPTR
MOVA A, @DPTR
MOV R2, A ←
L1: PUSH 0E0H → Saving A into stack
DIV AB
MOV R1, B
CJNE R1, #0, L2
POP 0E0H
SJMP OVER
ADD A, R2
SJMP L1
OVER:
```

So, we will do it like this, so first move, DPTR comma hash 6000 H. So, we have got the DPTR has been loaded with 6000 H, now the 6000 H has the smaller number. So, mov X A comma at the rate DPTR. So, this will be moving the smaller number into a register then we save this value in the register R 0 move R 0 comma A. So, this will save the smaller number in the R 0 register, from there we move it to B register. So, move it to B register, so that in B register the value is stored ok.

So, then we go get the next number INC DPTR to get the next number we increment this DPTR pointer, and move X, move X A comma at the rate DPTR. So, that way, so the next number the larger number is now available in a register and we move the larger number to R 2 move R 2 comma A. So, that moves the larger number to A register R 2 register.

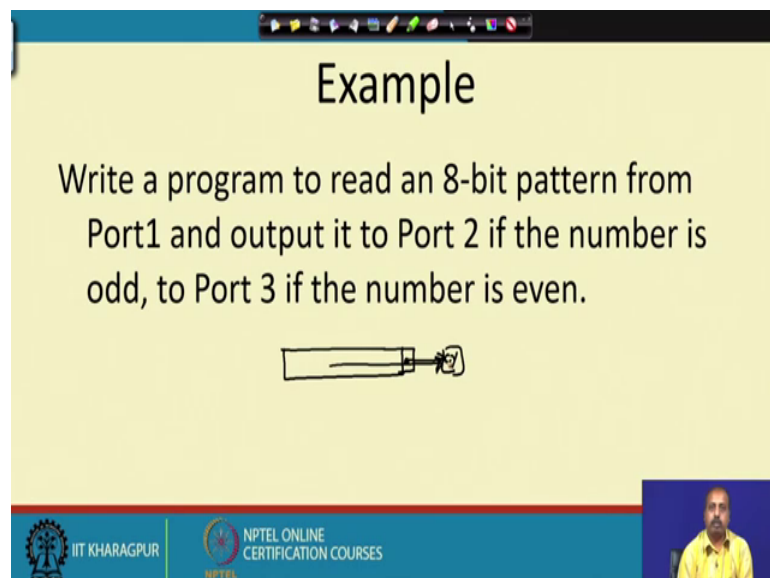
So, next we save this larger number into stack, so we do like push 0 E 0 H, so this will be saving A register into stack fine then we do a DIV AB. So, it will divide A by B, so it will do the division and now we have to check whether, so this remainder is 0 or not. So, remainder is available in the B register, so we move this move the remainder to the B register and see whether the value is 0 or not.

So, CJNE R 1 comma hash 0 comma some level L 2. So, in L 2, so otherwise if the result is 0; that means, it is a the division has been successful. So, whatever was the content of a register? So, that is the correct LCM value, so that we can get back by doing this pop. So, pop 0 E 0 H we do a pop to get the next instruction, and then SJMP over meaning that, so we will that is the end the program has been completed.

So, we have got the number. So, you do not need to do anything otherwise I have to get the next multiple of A. So, that I can do R 2 had a copy of A, so we can just add A comma R 2 can do a add A comma R 2 to get the next multiple of A, and then we just go jump back to this point this push this point. So, we can say SJMP, SJMP L 1 where L 1 is this address.

So, this is our L 1, this is our L 1 and this is over, so rest of anything any other thing that you want to do, so you can do it like this. So, in this way we can using the principle of dividing the multiple power the multiples of the larger number by the smaller number and whichever multiple divides fully with a without reminder. So, that is the LCM of the 2 numbers, so you can use this program to get this thing next we will be looking into another program.

(Refer Slide Time: 06:36)



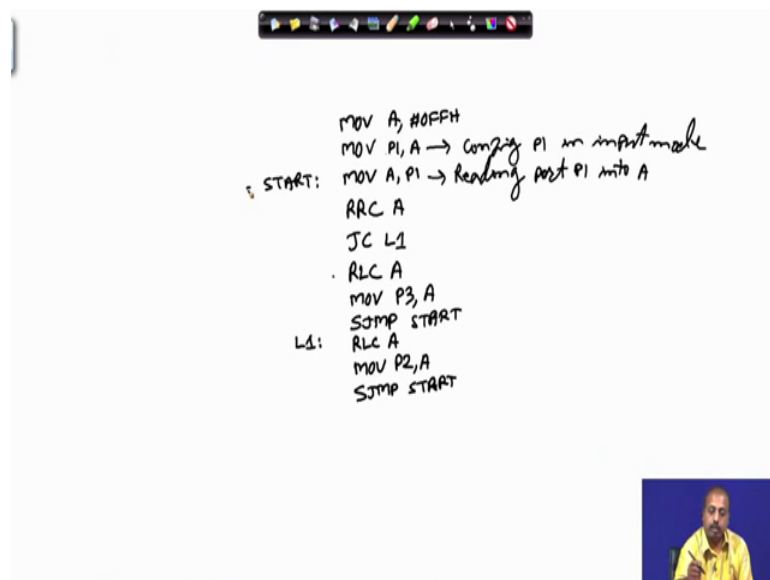
The slide features a yellow background with the title "Example" in black text. Below the title, the text reads: "Write a program to read an 8-bit pattern from Port1 and output it to Port 2 if the number is odd, to Port 3 if the number is even." A hand-drawn diagram shows a horizontal rectangle with a vertical line extending from its right side to a small circle containing the number "2". At the bottom of the slide, there are logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, along with a small video inset of a man in a yellow shirt.

So, this program is simply in the sense that it reads an 8 bit pattern from Port 1 and output it to Port 2 if the number is odd, and it outputs it to Port 3 if the number is even. So, this odd even check can be done by looking into the LSB of the a register or we can

we can just put it through a carry. So, normal strategy is like this whenever you have got this type of problems.

So, we just if this is the register or location that we have, so it is a passed through the carry. So, this LSB, so LSB goes to the carry if you right shift by 1 position LSB goes to the carry. So, we can then do check the carry bit and see whether the result is even or odd, so that can be there. So, whether the LSB was either 0 or 1 we check that way. So, we will look into the corresponding program, so it is like this.

(Refer Slide Time: 07:41)



Say this is, so move A comma hash 0 FFH. So, this is required because we want to read the content from Port 1. So, for that Port 1 has to be configured in input mode, so this will be doing that move A comma 0 FFH, we will put it in the input mode then move A comma P 0. So, when we are doing this then the Port P 0 is this will be reading from Port 1, so this should not be P 0 this should be P 1 ok.

So, this should be P 1, so that we are reading from Port 1, Port 1 is configured in read input mode then move no before this I need to read the content of the port. So, the this is to read an input to the Port 1 has to be read, so and sorry, so this this should be move P 1 comma A this should be move P 1 comma A. So, that Port 1 is configured in the read mode and then I can write move A comma P 1. So, this will be reading the content of P 1 Port on to a register. So, this statement is configuring P 1 in input mode, so this is the that configuration part and this 1 is actually reading Port P 1.

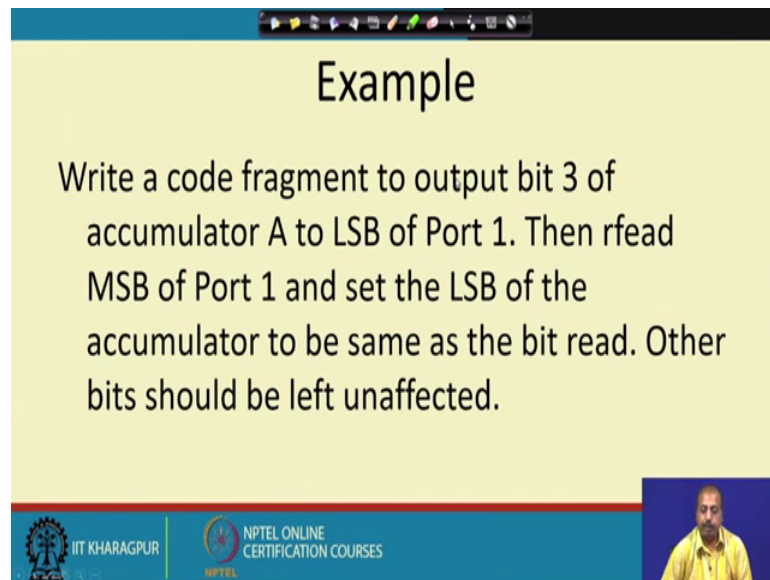
So, reading Port P 1 into A though the statements are just similar, but they have got different meaning then after that I have to see whether the value read is even or odd. So, for that we do a rotate right through carry now if there is a carry; that means, the number was odd, so jump on carry to level L 1 and the otherwise I have to the number is even. So, for that I need to I first restored that, so this is this RLCA, so this is actually restoring the carry, so that way it is done.

And then we will be move P 3 comma A, so this if it is odd in that case this RRC. So, here this carry bit will be set, so it will be jumping to L 1. So, if it was even then this is move to Port P 3. So, so at this point if we are here; that means, the number was even. So, we rotate it left, so that the carry bit comes back to the LSB of A register and then we move the content of A onto the Port P 3. So, that it is output to the number is output to Port P 3.

Then after that the we can after this we can say that the whole process will start all over again. So, if it is a continuous process, so we can do it like this. So, it is I can start from this point I can start from this point again reading Port P 1 of into A register, and this is the point L 1 and in L 1, what we do? In L 1 we rotate left carry through the a register through carry then we move P 2 comma A.



So, that the output is sent to the Port P 2 and then SJMP start, so it will take us back to this point. So, that it will be again continuing to the scan the Port P 1 if the number is even. So, it will be sent to Port P 2 if the number is odd, then if the number is even then it will be sent to Port P 3 if the number is odd it will be sent to Port P 2. So, that way it is the this program will continue next we look into another program.

(Refer Slide Time: 12:40)



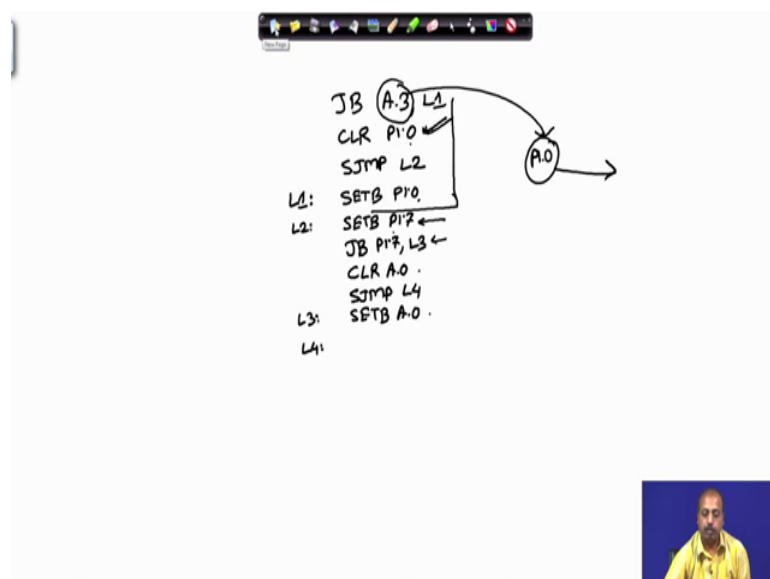
Example

Write a code fragment to output bit 3 of accumulator A to LSB of Port 1. Then rfead MSB of Port 1 and set the LSB of the accumulator to be same as the bit read. Other bits should be left unaffected.



 

Where so we will write a code fragment to output bit 3 of accumulator A to LSB of Port 1, this significant bit of Port 1 and then read the MSB of Port 1, and set the LSB of accumulator to be same as the bit rate, so from Port 1. So, we read bit 3 also from the bit 3 of accumulator will be put on LSB of Port 1, then MSB of Port 1 will be put will be read and will be set the LSB of accumulator same as the bit read. So, other bit should remain unaffected, so we will how to do this thing, so we can write the program like this say.

(Refer Slide Time: 13:37)



```
JB A.3 L1
CLR P1.0
SJMP L2
L1: SETB P1.0
L2: SETB P1.7
    JB P1.7, L3
    CLR A.0
    SJMP L4
L3: SETB A.0
L4:
```

We check this bit A dot 3 and if this bit is set, then we go to the level L 1 then we clear P 1 point 0, and then SJMP L 2 then in L 1. So, this is actually checking the bit number 3 of the accumulator register and it says that if the accumulator bit number 3 is set, then I should output to LSB of Port 1. So, B this bit should be outputted to LSB of Port 1.

So, if this bit is not set then I have done here correctly I have outputted is 0 to LSB or Port 1, if the bit is set then I have to output A 1 at the Port number 1 point of A Port bit 1 point 0. So, this set bit P 1 point 0 is doing that then in L 2 I have to do the other part. So, there I do like first we try this set bit, so this Port 1.7 the MS the, this MSB of Port 1. So, that bit has to be read and we have to set it to the LSB of the accumulator.

So, first of all this MSB of a Port 1 that should be put in the input mode, so for that purpose, so we do set bit P 1.7, so it is configuring that particular bit as input bit then jump on bit P 1.7 comma L 3.

So, if that now this JB, so this will be reading this particular bit and if the bit is L 3 if the bit is set then it will be jumping to L 3, otherwise if it is not set then I can clear the bit the LSB of this accumulator, this LSB of this register LSB of this accumulator. So, that is done here and then SJMP L 4. So, it will be jumping to this L 4 and in L 3 we just, we set that accumulator bit set B a point 0 and this is our L 4 ok.

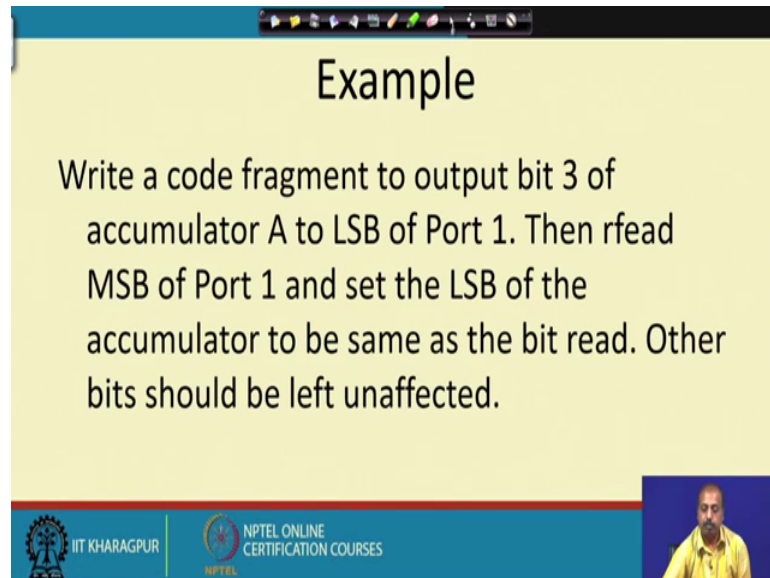
So, what is happening, the problem statement was that our bit 3 of Port 1. So, Port 1s LSB, so Port 1 0 bit, so this bit should be checked and accordingly this sorry this accumulator bit should be checked and then this should be Port 2, this accumulator bit number 3 should be checked and that should go to Port 1 bit number 0. So, this part of the program is doing that.

So, accumulator bit is checked if it is 1 then it is coming to L 1 it is setting Port 1 this LSB of Port 1, if it is not set then it is going to if it is not set then it is clearing that bit Port bit and it is jumping, so the it is skipping over this portion ok. So, up to this much is done and the next part of the program was to read the MSB of Port 1 and set LSB of accumulator accordingly the first statement.

So, this configures that MSB the next bit, so it is next part, so it is reading that value of Port 1.7 this particular bit if the bit is set it is coming to this L 3, where it is setting these accumulators LSB if the bit is not set then it is clearing the accumulator a in this LSB of

the accumulator. So, this way this program can be used for doing the necessary job whatever we have asked for in the problem.

(Refer Slide Time: 17:53)



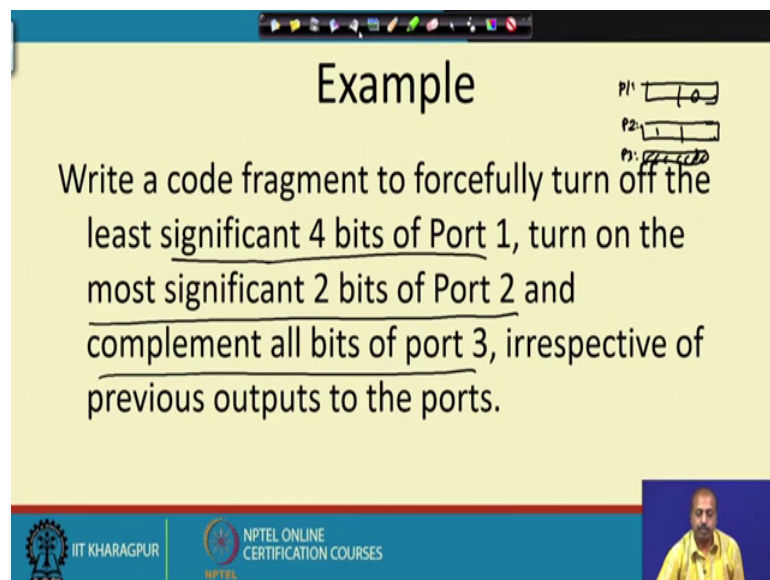
Example

Write a code fragment to output bit 3 of accumulator A to LSB of Port 1. Then rfead MSB of Port 1 and set the LSB of the accumulator to be same as the bit read. Other bits should be left unaffected.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, next we will look into another example program.

(Refer Slide Time: 18:00)



Example

Write a code fragment to forcefully turn off the least significant 4 bits of Port 1, turn on the most significant 2 bits of Port 2 and complement all bits of port 3, irrespective of previous outputs to the ports.

P1: [0000] P2: [1100] P3: [1111]

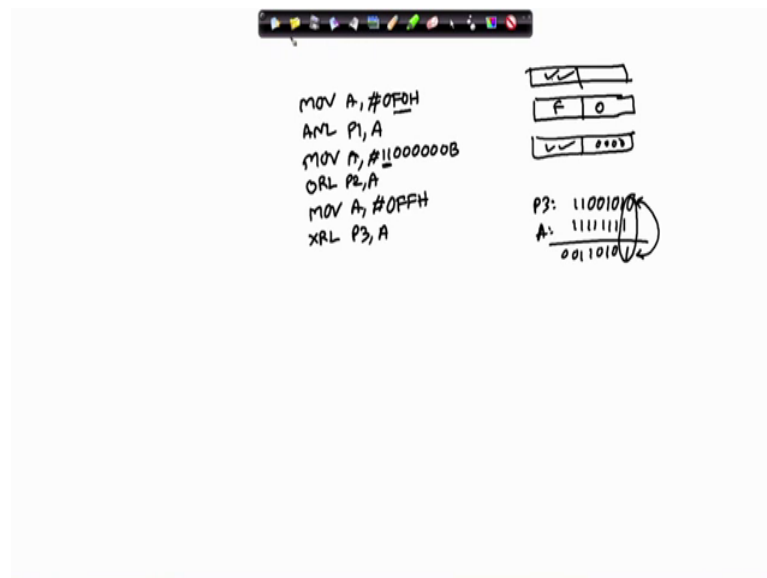
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, this program, so it is, it will forcefully turn off the least significant 4 bits of Port 1. So, least significant 4 bits they should be turned off. So, that is the first part and turn on the most significant 2 bits of Port 2 complement all bits of Port 3, irrespective of their previous setting. So, it should not disturb the settings otherwise, so Port 1, so the

configuration should be if this is the 8 bit. So, this lower bit, a lower 4 bits they should be they should be turned off they should be least significant 4 bits be turned off.

Then the most for 4 2 the most significant 4 bits will be 1, whatever be their previous setting. So, all these bits should be 1 and for Port 3 all bits are complemented. So, they are all of them are complemented. So, this is the problem statement that we have, so how to solve this in a 8 0 5 1 a single language, so we can do it like this.

(Refer Slide Time: 19:09)



So first we say move, A comma hash 0 F 0 and then and logical, A comma sorry and logical P 1 comma A and logical P 1 comma A. So, if we do this what will happen with this F 0 this Port 1 Port 1 content will be ended. So, Port 1 was this 1 and it is ended with f 0, so whatever was the content of Port 1, so this 4 bits, so they will remain as it is in the result. So, the these bits will come as it is, but this least significant 4 bits.

So, they will become all 0 because this is a end operation. So, by this we can do this then after that I have to set this we have to turn on the most significant 2 bits of Port 2, so most significant 2 bits of Port 2. So, how to do that, so I have to set another mask for that purpose, so we can prepare a mask like this move 1 comma A move A comma hash 1 once if I write in binary.

So, it is easily understandable, so this is the this is a binary pattern, so you can write a B at the end to tell that also, so it is a binary pattern. So, the least the most significant 2 bits

are to be set. So, for that the operation that we have to do is all logical and then with P 2 we do a logical oring of A. So, this or logical P 2 A, it will be oring the it will be oring this P 2 with A, so this most significant 2 bits will get set that way.

The last part is to complement the bits of this P 3 A P all bits of P 3. So, that you can do it like this, so move so for that the mask pattern should be all 1. So, move A comma X 0 FFX, and then we XRL, XR logical P 3 with A. So, if you do XR logical P 3 with A, so all bits are X or so x oring means it will be a complementation.

So, if the suppose the P 3 was the content was something like this 1 1 0 0 1 0 1 0. So, and now a register I have got 1 1 1 1 1 1 1 1, so if you do a bitwise x oring, so this x oring will give me 1 the first bit, second bit will give me 0, third bit will give me 1, fourth bit will give me 0, fifth bit will be 1, sixth bit will be 1, seventh bit is 0 and eight bits is also 0. So, you see that these value is just the complement of what we have previously because the x or operation, it will always do the complementation x oring with 1. So, it will always give you the complement operation.

So, this way we can write the program for bit manipulation and you can do all these things, by doing the easy bit manipulation instructions, so you can write this type of programs.

(Refer Slide Time: 22:44)

Example

Write a program to transmit continually the string "Hello" serially at 9600 baud rate with 8 bit data and 1 stop bit. $\tau_H = -3^v$

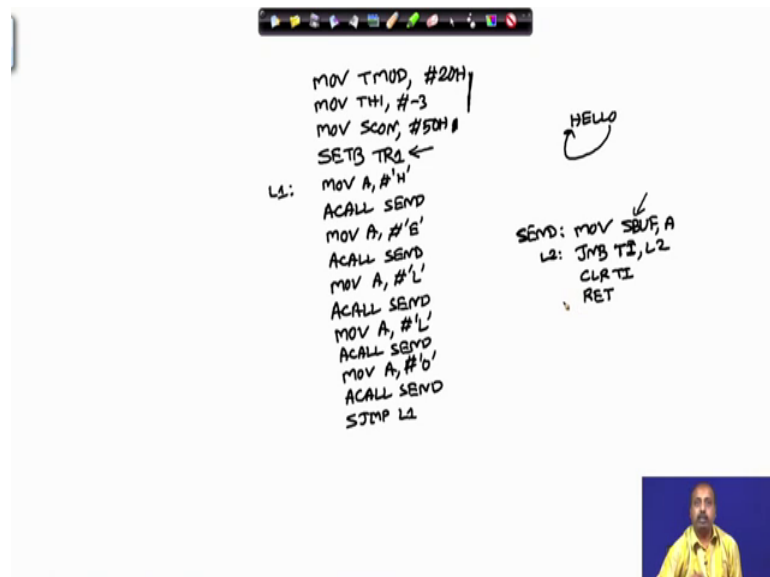
Write a program to operate serial port in mode 1 for receiving serial byte through RxD pin and send the byte to Port 2.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, next we will look into a program, next we will look into a program that will transmit continually the string hello serially at 9600 baud rate with bit data with 8 bit data and 1 stop bit. So, this is the setting, so you see that this requires the usage of this serial transmission and the timers are to be used and if you remember that in 9600 baud rate. So, this corresponds to this programming this TH register with the value minus 3.

So, if you put it in the TH register minus 3 and then and this T mod register accordingly. So, setting those control words we can select this minus 3 this 9600 baud rate for the for this transmission for the serial transmission. So, we will do that, so, we will let us try to see how this can be done, so first we configure this T mod register.

(Refer Slide Time: 23:46)



So, move T mod comma hash 2 0 H, so what it will do it will set this T mod register, so, that you can so it is basically that auto reload and all those things the modes and the other settings.

So, that it will be operating for the, this mode operation. So, the it can be used for the serial transmission and then this TH 1. So, it is using this timer register 1, so TH 1 will load it with hash minus 3. So, this way it is loading the value TH 1 with minus 3. So, that or in the auto reload mode it will operate and it will generate a baud rate of this 9600. So, that calculation we have seen previously while looking into this this timer and this serial transmission.

Then the next thing to do is to set the s con register, so for this particular case. So, you can find out that the s con setting should be 5 0 H you remember that this s con setting. So, this was for setting this the number of bits, 8 bit the 8 bit byte then 1 stop bit. So, whatever is specified? So, 8 bit 8 data bit a 1 stop bit and all these all those settings. So, they that will they will turn out to be the s con setting of 5 0 H.

Then so we have to start the timer, so we say set bit TR 1, so set bit TR 1. So, it will start the timer. So, that now a serial transmission is enabled so it will be able to do that now what we have to do is we have to put individual characters onto this s buff register and those characters will be transmitted. So, what we do first character is H for the hello we want to transmit the string hello. So, we want to and that we want to do continually, so it will continually transmit the string hello. So, move A comma hash H then we write a subroutine send, so which will be used to send the pattern send that string send the character.

So, ACALL send we will do that then we so H is done next is E. So, move A comma hash E and then ACALL send, then H E then L move A comma hash L sorry this should be another ACALL, ACALL send H E L another l. So, a move A comma hash L ACALL send then then the hello then the O, character has to be send. So, for that we say move A comma hash O ACALL send.

So, over now I have to repeat the whole thing. So, I can say SJMP L 1, wherein L 1, so it will do start going from this point onwards again move A comma H, now the send routine we have to write the send routine. So, the send routine will be something like this if this is that send, so it is like move. So, what if the send routine what is required is that that the character should be moved to the SBUFF register.

So, move SBUFF comma A, so it is from SBUFF it is doing it then I have to see whether the transmission is over or not. So, this TI flag, so it is raised when the transmission is over. So, I have to wait here till the TI flag is raised jump or not bit transmit interrupt TI, comma L 2 and L 2 is this 1 only.

So, it is the program will be waiting here till this transmission is over, and then I have to clear this TI, otherwise the next character transmission it will be taken as if it has already been transmitted. So, I need to clear that TI bit from the program and then return from

here. So, this way we can have a program by which some string can be transmitted over the serial lines of 8 0 5 1.

So, essential thing first this timer has to be set for this baud rate selection, then the SCON register has to be set for this configuring the size of the byte, the characters or bytes that you want to transmit and the that about the start stop bit start bit like that. Then this we have to start the timer at this point, then for transmitting individual characters the character should finally, come to the SBUFF register and after doing that after writing into SBUFF the transmission starts immediately.

So, we have to wait for this transmit interrupt flag to become 1. So, till it does not become one the character has not yet been transmitted fully. So, we have to wait here and once it is done, so we come, we clear this TI bit and then we can return, so these are the things to be remembered.