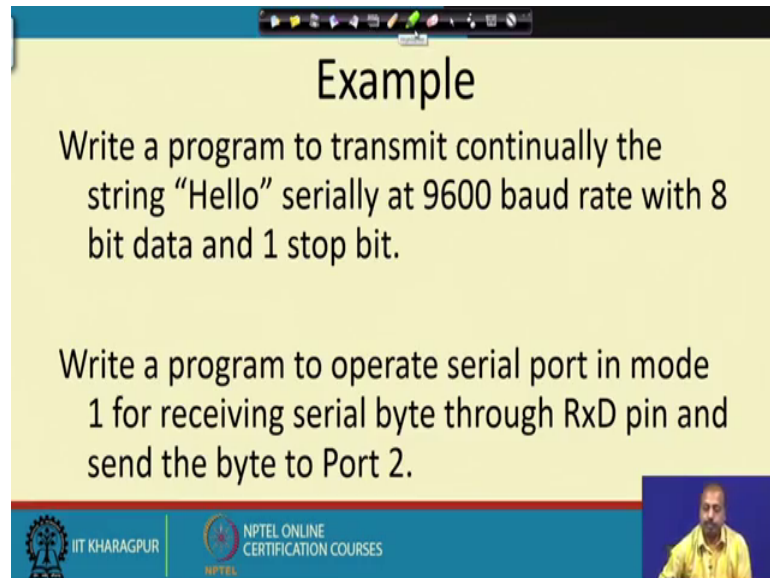


Microprocessors and Microcontrollers
Prof. Santanu Chattopadhyay
Department of E & EC Engineering
Indian Institute of Technology, Kharagpur

Lecture - 39
8051 Programming Examples (Contd.)

(Refer Slide Time: 00:20)



Example

Write a program to transmit continually the string "Hello" serially at 9600 baud rate with 8 bit data and 1 stop bit.

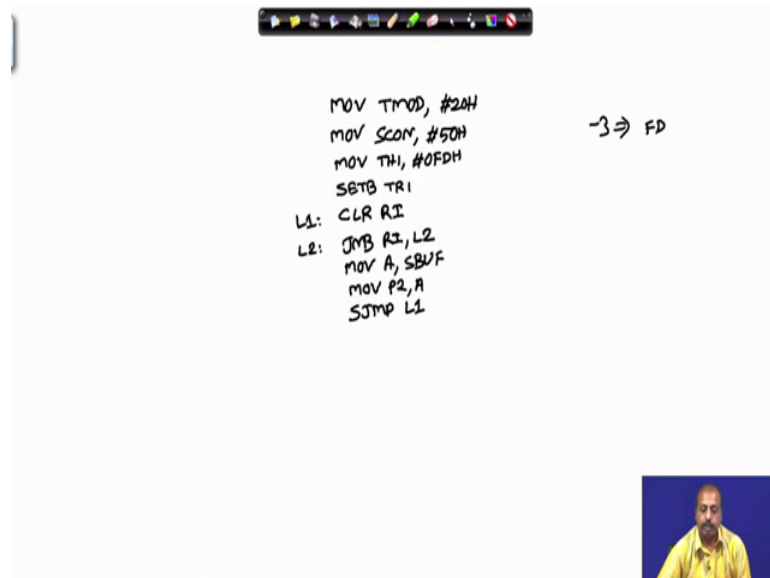
Write a program to operate serial port in mode 1 for receiving serial byte through RxD pin and send the byte to Port 2.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

The next program that we will look into is again on the serial port. So, it will be operating the serial port in mode 1, or receiving serial byte through, are the received R x D pin, and send the byte to port 2. So, it will be receiving one full byte serially from this through the received pin R x D and then the byte will be put onto this port 2. So, how to write this program, so let us see.

So, what we essentially do is, first of all this T MOD register has to be set, as you know that for any serial transmission. So, we have to use this timer 2 to act as the, this providing clock for this serial transmission.

(Refer Slide Time: 01:08)



So, we will start with setting this T MOD register. So, we first move into this T MOD register, the value has to be 0 hex. So, that we will consider, just in like the previous program. So, it will configure this timer 1 to act as auto repeat mode. Then the S CON register has to be configured, MOV S CON comma hash 5 0 hex, and this is again the control word that is required for this 8 bit data then 1 stop bit and all those things. Then I have to set the timer, so that I can get 9600 baud rate.

So, 9600 baud rate, so it was minus 3. So, that minus 3, so if you write in hexadecimal notation. So, it is F D, it is that is, so you can check the minus 3 in hexadecimal in twos complement notation it will be F D. So, instead of writing minus 3 you can also write like F D. So, you can write like MOV T H 1 comma hash 0 F D X, write like this, then we have to start the timer, so set B T R 1, this will start the timer.

Now serially we are getting the bits. So, I have to clear that receive interrupt R I the, you remember that whether you get T I or R I that bit is set whenever you are receiving some data. So, we clear this R I, so that if previously something has been received, it should not be taken into a part of this transmission. So, this is, it is clear there. So, jump or not bit after that we check whether we are receiving any bit or not on jump on not bit R I comma L 2. So, where L 2 is this point itself. So, we are basically waiting for getting the next R I being set, so J N B R I L 2. So, you will be getting the next character received on

the serially. So, the next characters will be coming, and when that entire frame has been received that is your 8 character bits and 1 stop bit, when everything has been received then this R I will be raised again by the processor.

So, now you can, if this is raised; that means, the character is available in the S BUF register. So, you can get the content from S BUF onto a register. So, you can write like MOV A coma S BUF. So, you get the character from S BUF register into A register, and then you put this value onto port P 2, MOV P 2 comma A. So, you will be getting the value from A register on to the port P 2.

So, that is what you are looking into. We were trying to get the content from this serial transmission line R x D and to getting it on to port P 2, so this character has been received. So, I have to go for the next character; so, S J M P L 1 S J M P L 1, where L 1 is this point. For the next character I have to clear the R I and again I have to wait for the R I to become high again. So, that is in the J N B. So, as soon as the next character will be received, so this will be, this R I bit will be set, so we have received the next character. So, this way we can write the program for this serial transmission. So, next we look into another program where that is again on serial transmission but slightly more complex.

(Refer Slide Time: 05:18)

Example

Write a program to get data from ^{Main} P1 and send to P2 continually. Incoming data from serial port be sent to Port P0 simultaneously. _{Interrupt}

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

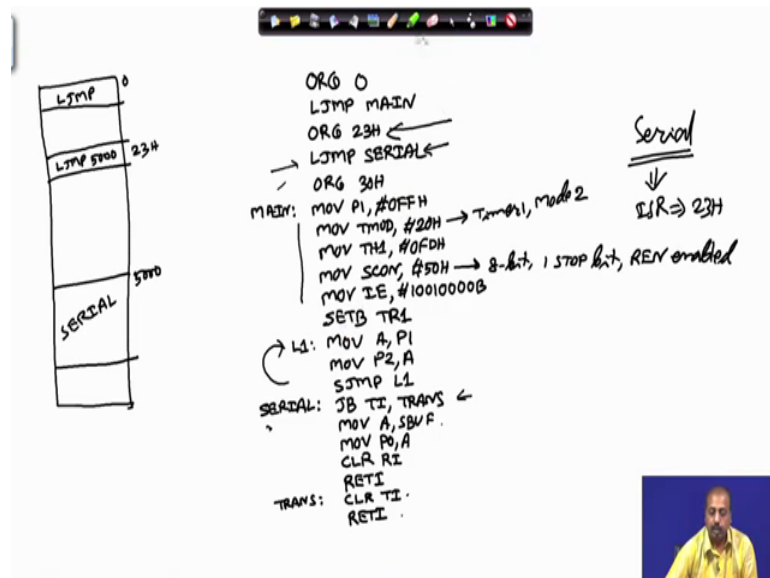
So, this program. So, this is suggested, it is expected that it will do two things together. So, it will get the data from port P 1 and send it to port P 2 continually. So, it will get the data from port P 1, and it will get the data from port P 1 and send P 2. So, this is the main job that it will do; simultaneously it will look into the incoming data from serial port and that will be sent to port P 0.

So, from port P 1 to P 2. So, that parallel I O, parallel read write that will continue. Apart from that there should be a serial communication from port P, from a serial communication. So, whatever comes on the serial port that should be sent to P 0. Now, the point is that serial transmission, so it will take time. So, if you do it in a multi flexed fashion; that is you first transmit 1 byte of from P 1 and send it to P 2, read 1 by from P 1 send it to P 2, then read one character on the serial port and this send it to P 0, then again read the next character from P 1.

So, it is not desirable, because the serial transmission will be slower compared to this port P 1 P 2 communication that we are looking for. So, ideally this P 1 P 2 communication, so this should be treated as the main job. So, this should be the main job, and this signal transmission, so this should be, by means of some interrupt, this should be by means of some interrupt, so that whenever this R I flag is set that receive interrupt will come.

So, this R I flag will be set and then will be able to do, we can read the from S BUF and transfer it to port P 0. So, this is what we will be doing in the program. Now since there are interrupts involved. So, we have to tell specifically where exactly the code should be put, fine. So, the main program can start from say at least 0.

(Refer Slide Time: 07:27)



So, we write it like ORG 0. So, that the assembler we will start putting the next piece of code at address zero of the memory. Here I have got only 1 L J M P instruction L J M P MAIN. So, if you look into this memory map; once the program is loaded, so it will be something like this. at location zero it will have that L J M P instruction. Whereas, this MAIN will be somewhere later, this MAIN will be somewhere later. So, that can be done. Then I have to use the serial interrupts, that serial interrupt has to be used, and for the serial interrupt you know that the corresponding I S R address I S R vector address is 23 hex. The 8051 manual. So, it you can see that the corresponding vector address for serial interrupt is 23 hex.

So, at location 23 hex, I have to put another jump instruction which will, if this is the actual serial transmission routine. So, if this is the serial transmission routine. So, if this is say for examples, this address is say 5000, then I somehow I should write here L J M P 5000 that should be written fine. So, that is done. How it can be done? So, we write another ORG here; ORG 23 hex. So, the assembler will know that the next instruction that you are going to write, it should be put at address 23 hex. Then here I put that L J M P serial. So, in my program serial is a level, so that will translate to some address. So, whatever addresses translates to, in this example I assume that it translates to address 5000. So, there you are putting this. So, we are writing like L J M P serial done.

Next thing is. So, serial routine will write somewhere, but now I can start the main routine, and main routine I want that it should not be immediately after this, there are some more interrupt vectors. So, I can say that my main routine will be starting from location 30 hex. So, from 30 hex onwards we have the main program. So, this is the main, from this point I am starting the main routine.

So, what I am doing. So, first I have to read port P 1 and out put it onto port P 2. So, port P 1 has to be configured as input port. So, for that purpose I have to do like MOV P 1 comma hash 0 F F hex. So, MOV P 1 comma hash 0 F F hex. So, this port, and then this port P 1 is configured as input port, and then for serial transmission I need to do the setting that I was doing previously in the other program.

So, those settings I have to do it here; MOV T MOD comma hash 20 hex, then it is timer 2 MOD 2 and then I have to have this TH register, MOV T H 1 hash 0 F D hex that is at minus 3 value, and then the S Con register has to be set, MOV S CON comma hash 5 0 hex. So, this setting it is timer 1 MOD 2. So, this timer 1 MOD 2 and this setting is 8 bit interrupt data, 1 stop bit and this receive enable, this interrupt is enabled.

So, receive the R E N; receive enable, this is enabled, the interrupt will come; otherwise the interrupt will not come. And then you can, the interrupt is; so make the interrupt enabled register also has to be programmed otherwise the interrupt will not be received. So, I E comma hash 1 0 0 1 0 0 0 0 B. So, this is the interrupt enable. So, it will be enabling the serial transmission interrupt. So, this you said that serial transition interrupt then I start this set B T R 1.

So, this will be starting the timer 1. So, we have set everything for receiving the data, we have to set port P 1 in the input mode and we have done other settings. So, that the serial transmission is enables with interrupt. Then the job is simple like now I have to just read from port P 1 and output on to port P 2. So, we can do it like this. So, we MOV A comma P 1. So, P 1 we read into A and then MOV P 2 comma A that way you can do, and then I can just put a loop here.

So, that this reading is done repetitively. So, S J M P L 1, so L 1 is here. So, it will be just reading from the port 1 and writing on to port 2 done. Now I have to, what is left is, writing down the interrupt service routine for this serial communication fine. So, this is

the address at which I am writing the serial routine and you see that at 23 hex I have got L J M P serial. So, it will come here.

So, first of all I have to see whether see the problem with serial interrupt is that, you see both transmission and D C they will generate interrupts, and both of them generate a single interrupt. So, I need to check in the interrupts service routine, whether it was due to some transmission or receive. So, for this particular program for trans if there is a interrupt is raise gives on transmission. So, that is not meaningful, it may be the processor was doing something else.

So, that transmits interrupt was generated, but I need to for those cases I do not need to do anything, but whenever the receive interrupt is there, so that is valid for us. So, both of them are mapped to the serial interrupt. So, both T I and R I. So, they are mapped to the same interrupt as you know in 8051. So, in that program, the interrupt service routine you need to distinguish between them. So, this is done here.

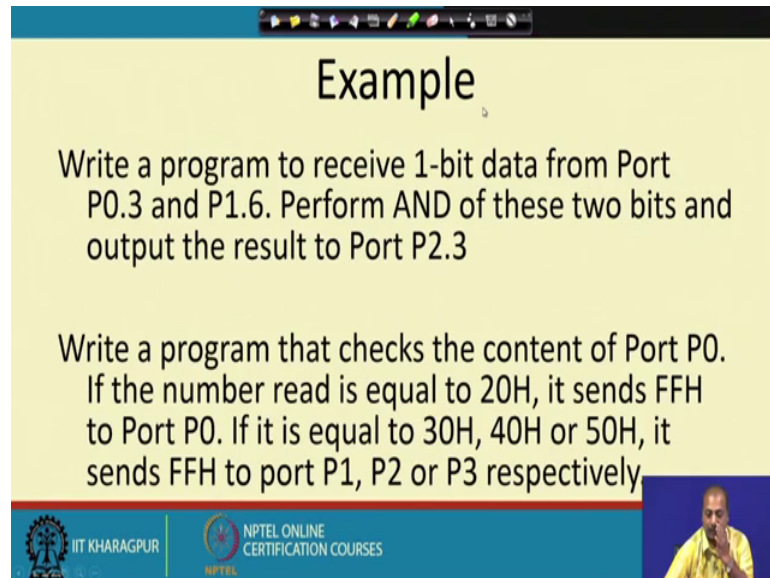
So, this we jump on bit T I to trans. So, this is actually skipping over the part of the code that we have for this time; otherwise, so if this is not the case and the interrupt has been raised; that means, it is due to receiving. So, in the S BUF register we have got the value. So, you can get it from the S BUF register back on to A, you can get and get into A register. And then move this content of A registered to P 0 MOV P 0 comma A and then clear this R I bit, then ret I and this trans part is actually here. So, in trans part I will clear that T I bit whatever be the reason for which it has been set, so that T I bit is reset and then this is ret I. So, you see that in this program how will it operate.

So, the main program is looping at this point. So, this main program after doing the settings it is a looping at this point continually, try reading from P 1 and putting onto P 2 in between the serial trans the serial interrupts will come, when the serial interrupt comes the processor will come to these address 23 hex, it will start 23 hex, it will find this instruction L J M P serial. So, it will jump to this point this serial.

So, it will first check whether it is A, the T I bit set or not. If T I bit is set then it will understand that this is due to some transmission which is not useful for us. So, it is coming here and returning from there; otherwise if this is due to a receive. So, this S BUF will be copied onto A and then A will be moving to P 0. So, this way this whole

program will operate. So, we can use the serial transmission in this fashion. Next we will look into another program which we will deal with mainly with the ports.

(Refer Slide Time: 17:28)



Example

Write a program to receive 1-bit data from Port P0.3 and P1.6. Perform AND of these two bits and output the result to Port P2.3

Write a program that checks the content of Port P0. If the number read is equal to 20H, it sends FFH to Port P0. If it is equal to 30H, 40H or 50H, it sends FFH to port P1, P2 or P3 respectively

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, this is telling that we will receive 1 bit data from port P 0.3 and 0.6 1.6 we perform end up these two and output put the result to put P 2.3. So, from 0.3 and 1.6 whatever you are getting. So, you need to end them and put the result on P 2.3. So, how to do this? So, we can do it like this that first this port P 0 and P 1 there I have to put it in the input mode. So, I do it like this maybe the entire port, I put it in the input mode MOV 0 comma 0 F F hex.

(Refer Slide Time: 18:07)

```
MOV P0, #0FFH
MOV P1, #0FFH
MOV A, P0
ANL A, #08H
RRA
RR A
RR A
MOV R1, A
MOV A, P1
ANL A, #40H
RR A
RR A
RR A
RR A
RR A
ANL A, R1
RL A
RL A
RL A
MOV P2, A
```

Diagram 1: A 4-bit register with bits 3, 2, 1, 0. Bit 3 is 0, bit 2 is 0, bit 1 is 0, bit 0 is 0. A value of 0 is written below.

Diagram 2: A 4-bit register with bits 7, 6, 5, 4. Bit 7 is 0, bit 6 is 1, bit 5 is 0, bit 4 is 0. A value of 0100 is written below.

So, that will read that entire that will make this entire port as input port MOV P 1 comma hash 0 F F hex that will put the interrupt P 1 as input port, then we do MOV A comma P 0. So, this P zeros content is available in a register and what we want is P 0.3. So, the 0.3; so that is if the corresponding if I consider the mask pattern, then I am interested in bit number 3 only. So, I am not interested in 2 1 0 or this part. So, I am interested only in this part this particular bit. So, for that we prepare a mask and that mask is 08 hex. So, the analogical A with hash 0 8 hex. So, if we do this ending then whatever be the values of these bits. So, they will become 0 and only this bit will survive, yes this bit will survive.

And now, we do some rotate right accumulated. So, rotate right accumulator once. So, that will take this bit to position 2 then this rotate right, once more it will take it to position 1 and then another rotate right. So, that will take it to position 0. So, after these 3 rotate rights this bit number 3 will be coming to bit number 0 fine, then we save this value into register R 1, this modified value is saving to register R 1, then we try then we read port P 1.

So, MOV A comma P 1 and there I am interested in bit number 6. So, I am not interested in 0 1 2 3 4 5 I am interested in bit number 6 the 7 and 8, so there if I want to get this one. So, this bit should be 2 for the mask and rest of the bits should be 0. So, the bit number this, this has what have in a meaning. So, this mask is 4 0. So, I do an analogical

similar to the previous case analogical A with hash 4 0 hex and then I need to take this bit to position 0.

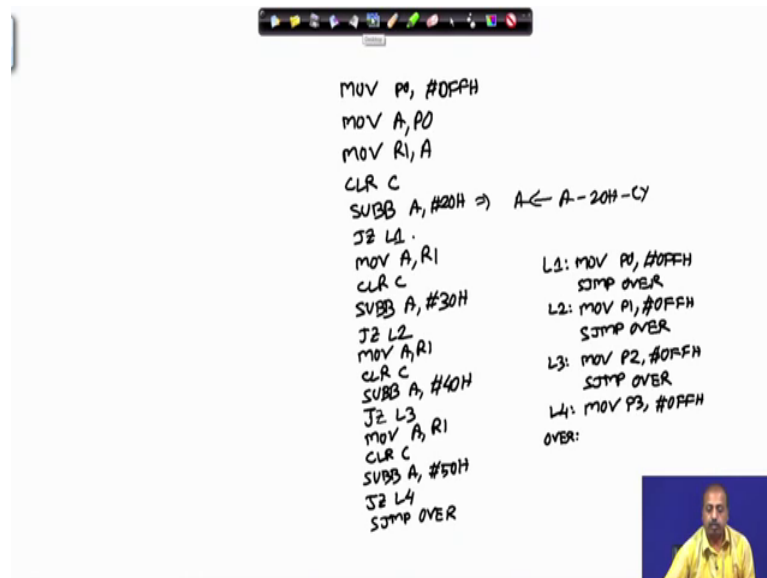
So, for that purpose I need to do, put a number of rotate rights 6 such rotate rights we will be require rotate right a hex into position 5 4 then 3, then 2, then 1 and then 0. So, at this point this sixth bit has come to position 0. Now I do so in R in R 1 register. I have got this port zeroes bit number 3 and in a register I have got port zeros port ones bit number 6. So, I can do one analogical A with R 1 and then I have to put the result on port T 2.3.

So, you can directly put, you can take those, do some bit set reset type of instruction and do that, but another way of doing, it is I want to go to port 3. So, if I want to go to bit number 3. So, I now do a rotate left A by 3 rotate left by 3 and then I put this pattern onto port 2. So, this way without using any bit manipulation instruction like a set B clear B J B J N B etcetera. So, if you are using; so if the byte oriented instructions or word oriented instructions.

Then you see that you can also do this bit setting resetting checking and all that, but of course, the problem is that the program becomes longer. So, you have to do many rotations. So, you definitely, you can write a much simpler version of this program that uses a simple bit set reset type of instructions.

So, this is just to show that this can be written in this form also. Next we look into another program that we will check the content of port P 0. If the number read is equal to 20 hex, it will send F F hex to port P 0, if it is 3 0 hex for 0 hex or 5 0 hex, it will send F F hex to port P 1 P 2 or p 3. So, it reads from port P 0, and if the number is equal to 0, then it goes to port F F goes to P 0. If it is 3 zero, then it goes to port F F goes to P 1, so that way the program is simple. So, we can just write the corresponding code like this. So, first the port P 0 has to be read.

(Refer Slide Time: 23:40)



```
MOV P0, #0FFH
MOV A, P0
MOV R1, A
CLR C
SUBB A, #20H ⇒ A ← A - 20H - CY
JB L1
MOV A, R1
CLR C
SUBB A, #30H
JB L2
MOV A, R1
CLR C
SUBB A, #40H
JB L3
MOV A, R1
CLR C
SUBB A, #50H
JB L4
SJMP OVER

L1: MOV P0, #0FFH
    SJMP OVER
L2: MOV P1, #0FFH
    SJMP OVER
L3: MOV P2, #0FFH
    SJMP OVER
L4: MOV P3, #0FFH
    SJMP OVER
OVER:
```

So, for that purpose I should have move this 4 P 0 has to be configured as input port. So, 0 F F hex then I have to say like. So, I have to get the content of port P 0 into A register MOV A comma P 0 and then we save this value onto R 1 from register in R 1 register, you save the value then we do a clear carry. So, then I can do it like subtract it borrow A comma hash 2 0 hex.

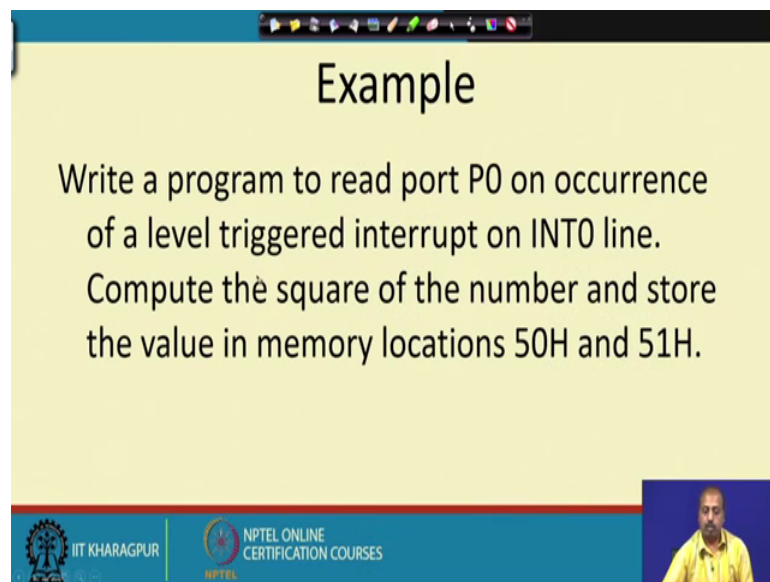
So, again this program I could have written in other ways, also using comparison instructions and all that. So, instead of that what I am trying to show, is the usage of the subtract with the borrow feature, for that a carry flag has to be cleared, because this instruction will do a, will get a minus 20 X minus the carry bit. So, before doing this I said the carry bit 2 0. So, that it is A A minus 20 hex only. So, clear, so if it is 0; that means, the number was equal to 2 0 hex.

So, in that case I go to level L 1 otherwise I need to check whether it is equal to A 3 0 or not. So, MOV A comma R 1 and then again clear carry, and now I subtract with borrow A comma hash 3 0 hex. So, hash 3 0 hex. So, again, so if the number is, if the result is 0; that means, if the number was 3 0 jump 1 0 to L 2, then we can say like say MOV A comma R 1, again clear carry and subtract with borrow A comma hash 4 0 hex jump 1 0 2 L 3 and then MOV A comma R 1 this part is similar.

So, we can just go on repeating it. So, clear carry and then subtract with borrow A comma hash 5 0 hex jump 1 0 2 L 4 and if nothing happens it does not matter, if anything, none

of the setting will be required. So, it will be go to the end of the program which is another and level say over. Now comes the level L 1 in level L 1. So, I have to; this is the point L 1. So, in that case F F should go to port P 0. So, we can see like MOV P 0 comma hash 0 F F hex, then S J M P over, then I can say L 2, here I will be moving to port P 1 and P 1 comma hash 0 F F hex S J M P over then L 3 MOV P 2 comma hash 0 F F hex S J M P over and L 4 which will MOV P 3 comma hash 0 F F hex and this is the point over. So, this is the point over where it will be doing that. So, this way you can write a program by which, again this as you can just by other complex checks can be there. So, this program is actually trying to show the utility of this subtract with borrow type of instruction. So, you can do this thing.

(Refer Slide Time: 28:24)

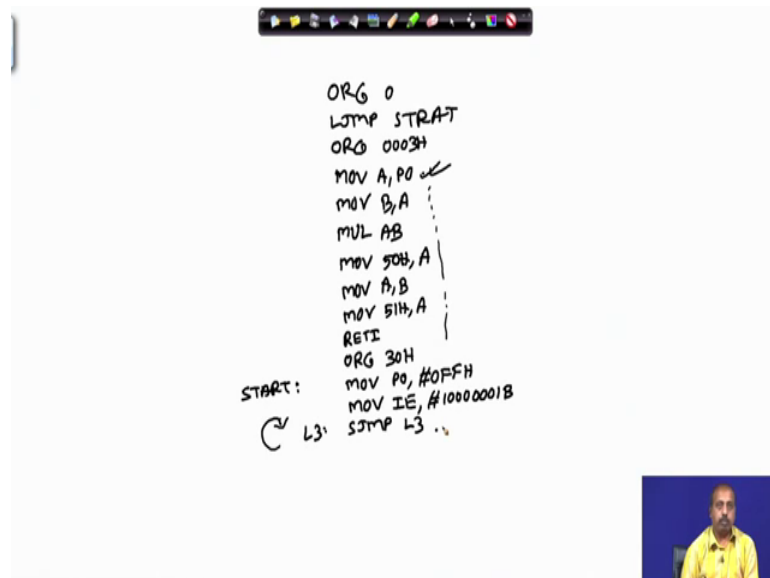


Example

Write a program to read port P0 on occurrence of a level triggered interrupt on INT0 line. Compute the square of the number and store the value in memory locations 50H and 51H.

Next we will look into another program. So, it will program to read port P 0 on occurrence of a level triggered interrupt on I N T 0 line. So, I N T 0 line. So, if there is an interrupt then this, it will lead the content of port P 0, and then it will compute the square of the number and store the value in memory location 50 hex and 51 hex. So, thus tried some interrupt service routine. So, how to do that?.

(Refer Slide Time: 28:56)



```
ORG 0
LJMP STRAT
ORG 0003H
MOV A, P0
MOV B, A
MUL AB
MOV 50H, A
MOV A, B
MOV 51H, A
RETI
ORG 30H
START: MOV P0, #0FFH
MOV IE, #1000001B
L3: SJMP L3
```

So, we can do it like this O R G 0 L J M P start this, the start is the main program from where it will be there. So, any interrupt service routine that you have. So, interrupt service routine have to tell like what how to put that. So, then in O R G then this interrupt service routine for this I N T 0, the corresponding vector address is 0003 hex. So, if they write like 0003 hex.

So, I am not looking into that, if I write that interrupt service routine code here. So, it will overwrite other interrupt service interrupt vector table. So, I am not bothered about that. In this case I am assuming that from this point on itself the interrupt service routine will be loaded. So, what we need to do is, here we need to get the content of this A register. If this P 0, the port P 0 into A register and then we have to get the square of them.

So, for getting the square, the value, the MOV, the value of MOV, the value of A to B and then use the instruction multiply A B. So, we get the square and this A resistor. Now we will have the 1 1 byte and register will have the other byte. So, this A register value you say that memory location 50 hex, and then this B register value I will be saving in 51. So, for that purpose we first move the content of A B register, MOV A comma B, and then save the value in 51 hex and then let i.

So, this is the end of the interrupt service routine. So, then from now, I have to put the main routine. So, suppose the main routine I am putting from where memory address 3 0

hex. So, we can put a `ORG 30` hex. So, there you put the start. So, in the start in the main routine what I need to do is, you see that at this point you were reading the value up port `P0`. So, we have to load `P0` with configure `P0` with a pattern. So, you should put a statement like this `MOV P0, #0FF` hex. So, this is one statement and the other thing is the enabling of interrupt. So, for that the interrupt enabled register `IE` should be program `MOV IE, #10000001` binary.

So, you can check the interrupt enabled register setting. So, this will say enable the system to getting the `INT0` interrupt and then the program has just to wait. So, this is `SJMP L3`. So, it will be waiting at this point and whenever an interrupt will come, the system will automatically be redirected to this point, the `INT0`. If you automatically be redirected to address `0003`.

So, from this point onwards it will execute that port and data, you will take it back to this point. Again sometime later the interrupt will occur, again the value will be read from port `P0` and moved into the memory location, the square of it will be stored in memory location `51` and `50`.