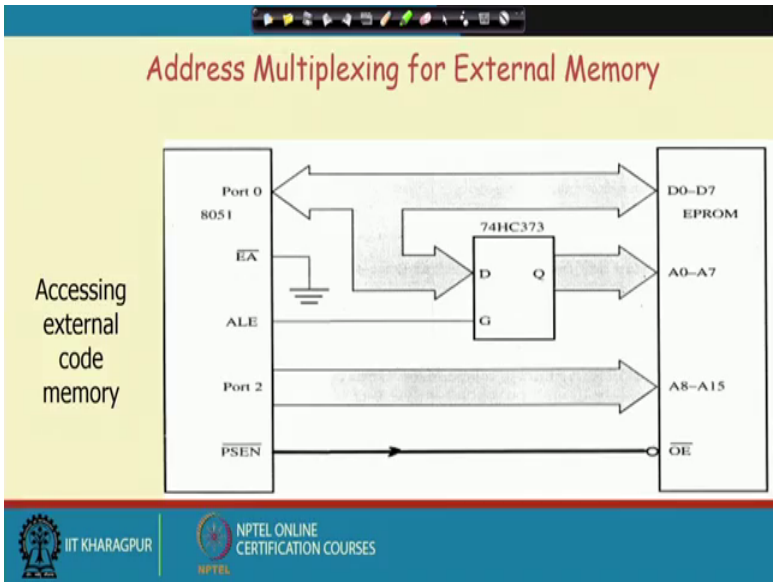
Microprocessors and Microcontrollers
Prof. Santanu Chattopadhyay
Department of E & EC Engineering
Indian Institute of Technology, Kharagpur

Lecture – 25
8051 Microcontroller (Contd.)

As far as address multiplexing is concerned we have got 8051 when it is connected to one external program memory that is EPROM.

(Refer Slide Time: 00:23)

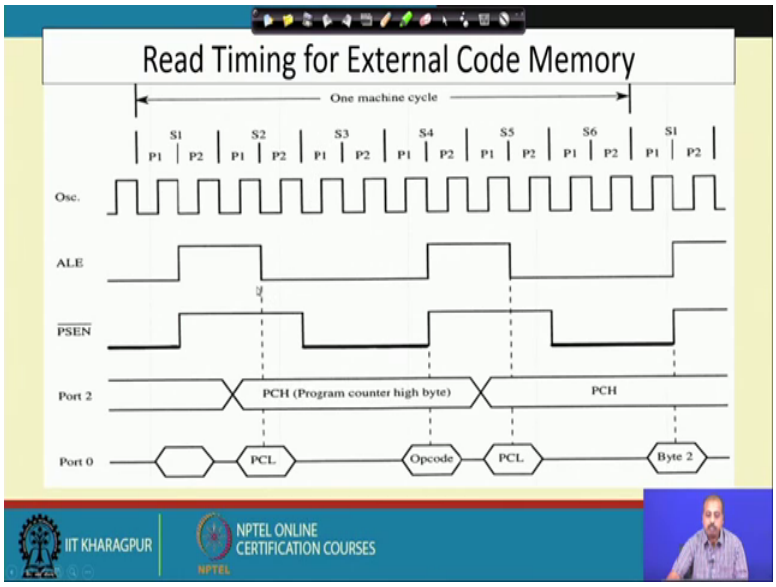


So, this is the external address bus from Port0. So, we have got this Port 0 we. So, Port 2 is providing the higher order address bus A to A 15 bits. Port 0 is the multiplexed addressed data bus. So, this is passed through one 7 4 3 7 3 latch for multiplexing for de multiplexing the lower order address bus and the ALE signal is connected to g pin of 7 4 3 7 3 by which we can when this ale signal is activated address is available in the bus and that gets latched here.

So, as far as this EPROM is concerned, it continually seize the address bus address bits 8 0 2 a 7 a to a 15 and a when this PSEN bar line is activated. So, it is connected to the v bar pin of this EPROM chip. So, v bar is same as read bar as we said. So, it is. So, output enable. So, read bar line. So, when this lines are given. So, EPROM will put the content on the data bus and the data bus through port 0 will reach 8051. So, in this way we can connect external program memory to the 8051 chip.

So, we also call it external code memory because this external program memory is generally holding the program of course, in some cases you can connect you and put the data they are as well, but for general generally. So, that will also be treated as code only.

(Refer Slide Time: 01:56)

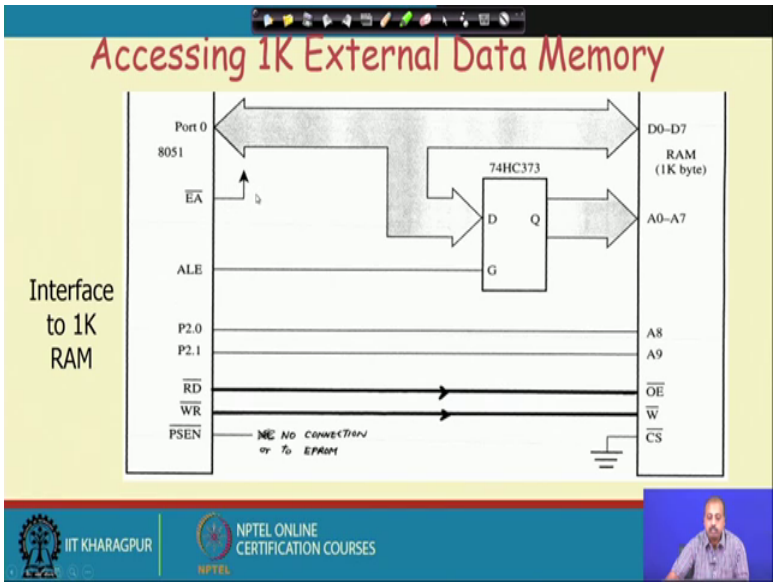


Ah. So, if we look into the timing diagram. So, if this is the oscillator clock that we have. So, we have got this ALE signal. So, they are activated at this point. So, when the ale signal is of following. So, at that time the pc the Port whatever on Port 0, this lower order address bits will be put and that will get that will be sensed here. So, that will that will get latched at this point and when after that this PSEN bar line will go low. So, when is PSEN bar line goes makes a transition from low to high. So, that is taken as the point at which the data bus will have the Opcode available.

So, at this point; in between this activation of pc low and this activation of this PSEN bar. So, memory is expected to put the Opcode on to the data bus. So, Opcode will be available any time like this and when is PSEN bar line makes a transition from low to high then only the Opcode will be taken of by the memory. So, it is the protocol the memory should keep the Opcode available for that much amount of time and then this 8051 will sense this Opcode through this PSEN bar line and this value is read here again later sometime the ale signal will go low and when it is going low. So, it will be used as the value to latch the lower order address bus to a lower address on the Port 0 that lower order address bus.

So, you see that this signals when they are falling. So, they are used for they are used for sensing the values or similarly this ale signal is when it is making the transition high to low. So, they are actually taken as the triggering point and this PSEN bar line. So, when it is since it is PSEN bar then it is the this complementation is there. So, when it makes the transition from low to high. So, that point is taken as the triggering point for the memory to put the data.

(Refer Slide Time: 04:08)



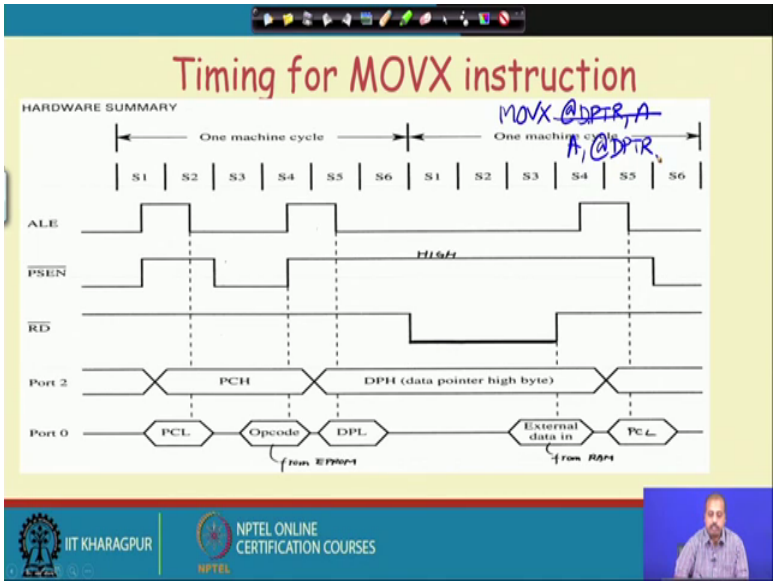
So, this way this whole operation takes place the external code memory access reading takes place this in this fashion. On the other hand if we look into the data memory connection external data memory connection then we have got the connection like this. So, here also. So, we have got one say suppose we are going to connect one Kilo Byte of RAM suppose the RAM we are connecting is only one Kilo Byte of long. So, we have one Kilo Byte means. So, they it the RAM will have 10 address lines.

So, this lower address bus that A0 to A7 they should be have connected and this A8 and A9. So, this memory chip has got 10 address lines A0 to A9. So, A0 to A7 will come from Port 0 the through this multiplexing through this demultiplexing of the address bus by 7 4 3 7 3 and this A8 and A9. So, this two are coming from the Port 2's bit number 0 and bit number 1. So, the rest of the bits are not connected. So, as we know that it may lead to folding and all that. So, we will not go in to that. So, other bits being do not care. So, there that can lead to folding.

Then the read bar line that we have. So, read bar line is connected to the v bar or output enable of this ram chip and the right bar line is connected to the right pin of the a RAM chip. So, the there is no other decoding done. So, this cs bar pin is grounded. So, this RAM is always enabled. So, whenever the address is put on to the address bus and this signals read bar or write bar is given the content of the memory cell will be available on the data bus to be read by Port 0.

So, apart from that this PSEN bar line we will have no connection. So, if you are not having any external EPROM then this PSEN bar will never be connected and this if it is if the EPROM is there then that PSEN bar line may be connected to the EPROM and this ea bar line also the external access. So, that is that is connected to high because this is not doing an external access as far as the code is concerned this is not code access. So, this is data access. So, there is no problem and.

(Refer Slide Time: 06:23)



The timing diagram again. So, this external data memory access is done by the MOVX instruction we will look into this MOVX instruction in detail later, but essentially it is format is something like this.

So, it is like MOVX at the rate DPTR sorry MOVX at the rate DPTR comma A or another option is the other way. So, other the it is for getting the data from A register into a memory location pointed to by the DPTR register pair and there is another version for

storing for getting the content of memory location to the accumulator. So, there you just write it as MOVX A comma at the rate DPTR ok.

So, those are 2 versions of the MOVX instructions. So, they are for external data memory access for external data memory access. So, it uses the MOVX it uses the MOVX comment or MOVX instruction. So, how does it operate. So, you see that first it has to get the instruction from memory; for that purpose, assuming that this Opcode this instruction is in the external memory. So, this is the PSEN bar line is activated. So, that you get the Opcode from the external ROM external EPROM.

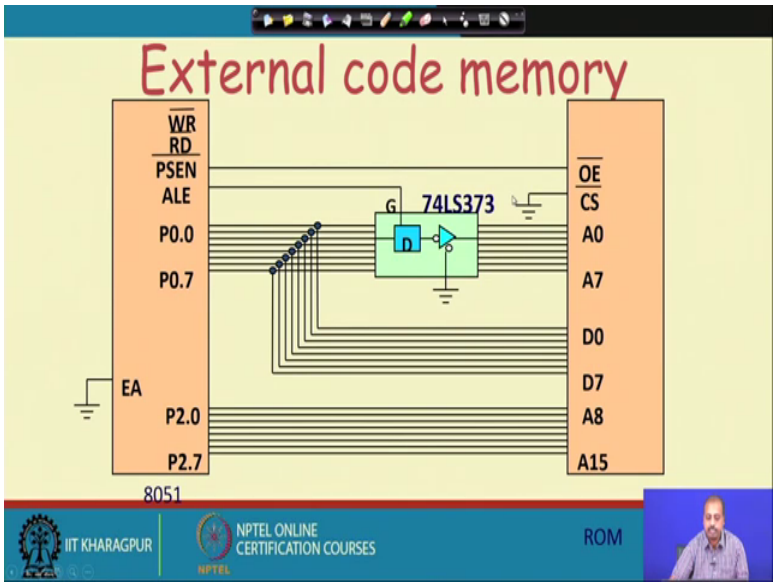
So, that is though this apart we have already discussed how to access this external ROM ale signal is activated the address is given lower order address given the higher order address is put on the pc high and this PSEN bar line is given when PSEN bar is going high the data the Opcode MOVX Opcode is available a on to the data bus then the decoder will decode that MOVX instruction and now it will go in to the second stage second stage and in that second stage it has to it puts the read bar line.

So, DPHigh the DPTR register it is consisting of two pairs DPHigh DPH and DPL to 8 bit register whereas, DPTR is the 16 bit register. So, this DPHigh. So, that will be put onto Port 2 because through that this external address bus will be connected to the external RAM. So, it will be put there and this DPL. So, this will be this is again multiplexed with data bus. So, that is the Port 0. So, these value this. So, lower order address bus is multiplexed. So, in the lower order address bus put the value DPL.

So, high order address bus has got the content of DPH lower address bus has the control of the content or DPL and then this read bar signal is activated when this read bar signal makes a transition from low to high by that time from the ram the external ram the data should be available on to the data bus. So, on this rising edge the value will be picked up by the 8051 chip and again the. So, so that is the. So, MOVX instruction is complete. So, data is since it is it is basically reading the content of memory location into A register. So, read bar signal is given similarly if you want to write like if you want to write the content of A register on to external dat[A]- on to external memory.

So, in that case right bar signal should be given and here instead of coming from external data input it will be from the A register the accumulator.

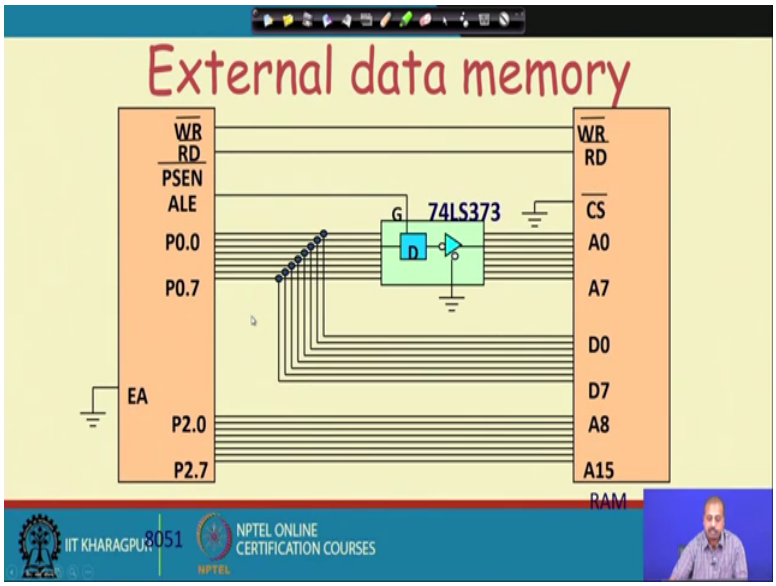
(Refer Slide Time: 10:14)



Register. So, this how this connection is done like external code memory so for external code memory connection. So, we have got this um this higher order address bus P2 provided by Port 2. So, 2 point 0 2 to 2.7 they are connected to A to A 15 then Port 0 provides the multiplexed address data bus.

So, this 8 bits they are connected to the data bus of the RAM chip and a data bus of the ROM chip and also it goes to the latch 7 4 here 7 3 latch where it is de multiplexed by the ale signal and. So, this is. So, this is the this is the demultiplexed address bus this is the demultiplexed data bus ALE signal is connected to the g pin of 7 4 3 7 3 PSEN bar line is connected to OE bar and CS bar line is grounded and this EA bar line is grounded. So, that is it does not have any meaning because it is not the data access. So, it does not have any meaning.

(Refer Slide Time: 11:22)

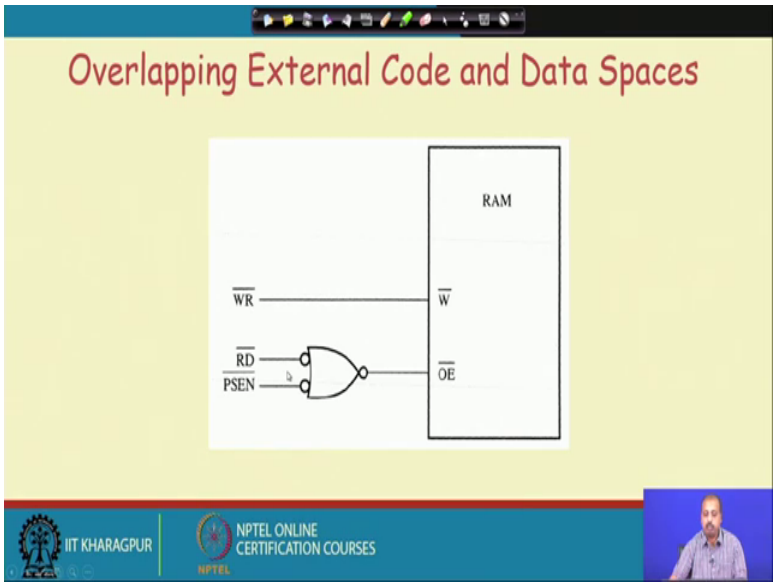


So, for external data memory access, this is the thing that is a right bar line is connected to write bar read bar line is connected to read bar PSEN bar. So, PSEN are is not necessary in this case. So, if the connection is not shown here PSEN bar is not meaningful ALE is ALE signal is connected to G. So, rest of the thing is unaltered. So, P 0 to 7 goes to this latching of latching of this lower order address bus here and the data bus is connected to D0 to D7.

So, this EA bar. So, this is connected to ground meaning that is the external memory access. So, this EA bar line is grounded in the previous case also this A bar line was grounded because we were doing memory external memory access. So, in both the cases external memory, there will be some more detail the where will see how this EA bar line makes difference between this external line access and internal access.

So, since here we are doing external access. So, EA bar pin should be grounded and for that code memory access also since we are doing external memory access this EA bar line should be grounded.

(Refer Slide Time: 12:40)



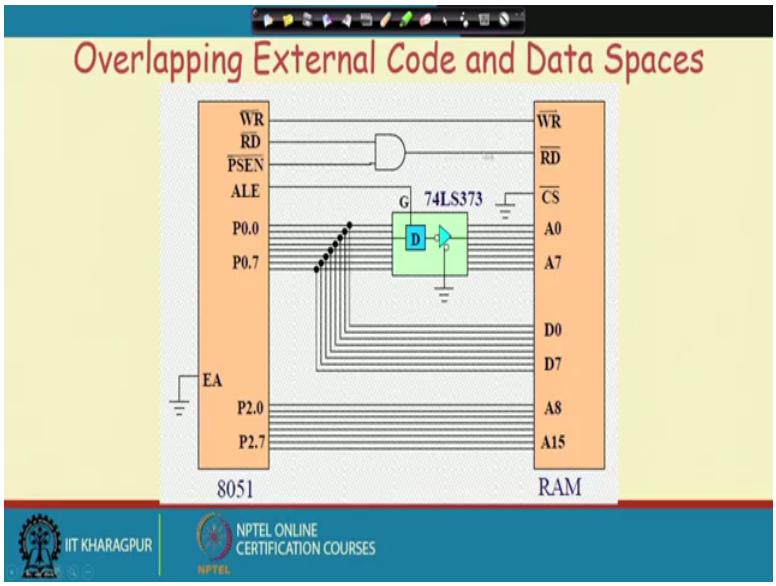
So, next we look into how can we overlap this external code and data spaces. So, this is very simple because if we if like instead of using a separate EPROM for storing the program. So, many cases what happens is that we put with the micro controller our single externally you put a single RAM chip and in that ram chip. So, it acts both as program memory and data memory.

So, for that purpose what we have to do is we take the RAM chip and then this right bar line of the processor is connected to the right bar line of the RAM and this RD bar and PSEN bar they are odd and they are that is connected to the OE bar line. So, which ever signal is activated. So, RD bar or PSEN bar. So, both the cases, OE bar line will get activated. So, this RAM will be providing the data.

So, read bar is activated when it is doing when it is doing say data access and PSEN bar is activated when it is doing program access since, program and data both of them are residing in the memory in the RAM. So, we need to access RAM in both the cases. So, this is a very common practice for systems that have got limited amount of external memory. So, we do not put separate ROM and RAM. So, we put a RAM that also serves the purpose of the ROM.

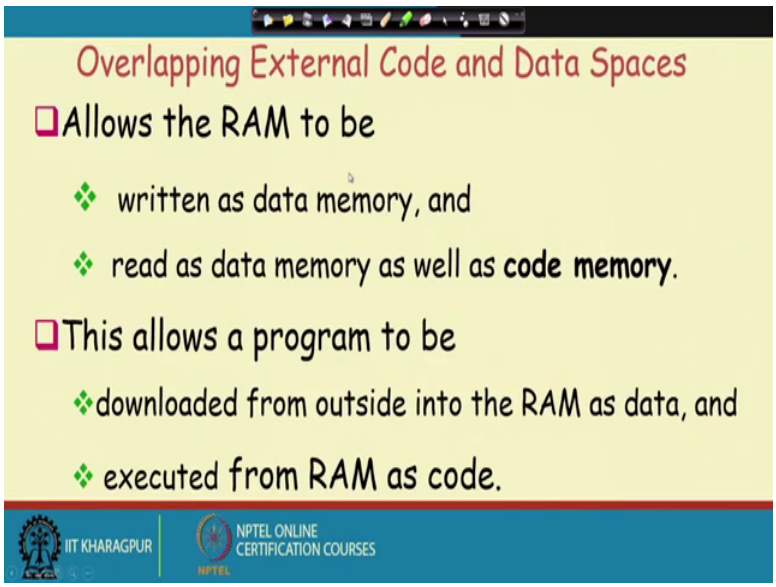
So, this is how we do that overlapping. So, this read bar and PSEN bar. So, those two lines are ended and it is given to the read bar line. So, any of them being

(Refer Slide Time: 14:14)



Low this read bar line will be low. So, we will get that corresponding signal here telling that it is a it is read operation similarly write bar line is connected to the write bar pin of the RAM. So, rest of thing we have already discussed. So, this is the only additional thing that we have to do this putting off and of read bar and PSEN bar for doing the overlapping of code and data spaces.

(Refer Slide Time: 14:38)



So, what is the advantage first of all this overlapping of code and dataspace is it allows the RAM to be written as data memory, and read as the data memory as well as code

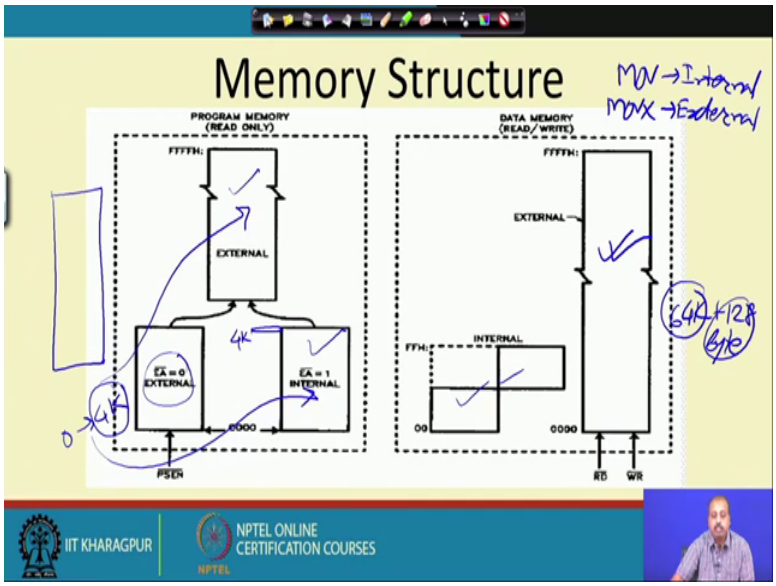
memory. So, while. So, no code memory is in general ROM. So, we do not write there we just read the data we just read the code from there and the constants that we have in the program. So, they can also be treated as code and those constants are also kept in the ROM ok.

So, if you are using RAM then for write for operations. So, you can use the right bar pin access and then. So, we can write the values on to the RAM as data memory and if you are trying to access the program then you can use that PSEN bar or if you are trying to access the data part of the RAM then you can use the read bar line. So, we can. So, what is the advantage of doing all this things? So, this will allow a program to be downloaded from outside into the RAM as data and executed from RAM as code.

So, many times what happens is that with this microcontroller based systems. So, they have got interface with program with personal computers and then that through a download cable we download the program into the kit. So, the program is first developed on the pc. So, we do the simulation and all that. So, when we are satisfied we generate the object code for that program on the pc and then that object code is downloaded onto the 8051 kit and when we are doing this. So, if it is ROM then we have to go for programming of that ROM and that is costly because again if we want to change something erasing will also become bit difficult compared to RAM.

So, instead of that if we have got RAM to be used as the code memory as well then we can just write the we can just write the values on to the RAM. So, that way the entire program can be downloaded into the RAM and then we can use that RAM both for program as well as data.

(Refer Slide Time: 16:55)



So, if you look into the memory structure that we have. So, the program memory structure is like this. So, we have got this the total address space total address line that 8051 has is 16 bit. So, that is address bus is 16 bit. So, the address total address range is 0000 to FFFF.

Now,. So, you are your the access is from it starts with 0000 onwards now if you make this a EA bar equal to one so; that means, we are accessing from the internal memory and if you make EA bar equal to 0 it will access from the data memory. So, in case of if you do it like this then what happens is that you have got this part. So, you can put your entire program in the external memory so that in that case we will make A bar equal to 0 and this if you have got for K address space then this 4K sorry; the when A bar is 0 your entire memory external memory will be acting as the external EPROM that we have.

So, externally we connect 64K. So, type of program memory and then this through this PSEN bar line activation. So, this entire external chip will be accessed whereas, if you make this EA bar line equal to one then for initial access. So, it will be using the internal internal memory, but after that it will be using the external memory once it finishes once it finishes this maximum address for example, in 8051 we have 4 Kilo Byte of internals memory internal ROM. So, when this is over when this is over from the next address onwards. So, it will be accessing this eternal memory.

On the other hand, this for data memory part, we have got this 00 to FF. So, that much will be internal memory and from FF onwards. So, it will be accessing the external memory. So, from because only 128 Bytes are there for the internal memory, first this 00 to 127, it will be internal memory and 128 onwards. So, it will be accessing external memory and that is detected by the MOVX instruction.

So, whenever you are using MOVX instructions. So, it will be using the external memory and whenever you are using MOV instruction. So, it will be using internal memory. So, will see that there are two data movement instruction one is sorry one is MOV and the other is MOVX. So, MOV is for internal access. So, this is internal and there is a MOVX instruction that we have seen. So, MOVX is for external.

So, if you are using MOVX then it will be using this external memory if you are using MOV it will be using this internal memory. So, that is the 8051 policy. So, for int. So, you can say in some sense I can say that the total data memory that 8051 will have in that case is 64K here plus this 128 Byte internal.

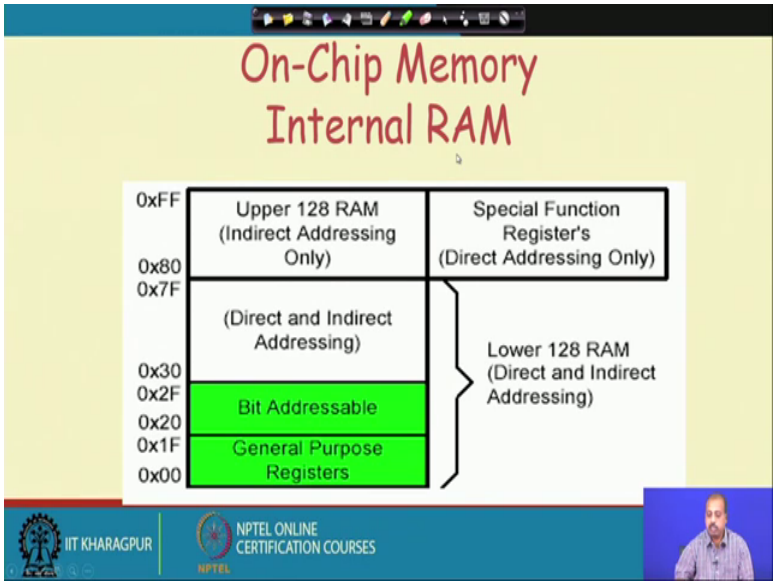
So, this 64K external plus this 128 Byte internal, that is the total data memory we can have for the program memory side it happens if a bar is equal to 0 then the total at total then it will be accessing for whenever it is accessing may programs. So, it will be accessing this external memory; however, if this A bar line is equal to 1 and the this maximum address is say 4K then for the address range 0 to 4K for 0 to 4K it will be accessing this memory and when the address goes beyond this 4K it will be accessing this external memory ok.

So, if EA bar line equal to one then for address between 0 and 4K it will access internal memory internal ROM and beyond 4K it will use the external ROM whereas, if A bar line is equal to 0 in that case for any address it will go to the external memory for getting the Opcode or getting the instruction. So, that makes it very flexible. So, we can have we can we can configure this system as per our requirement.

In many of the microcontroller development kits, they do it like this. So, they just map this external a entire memory as the external memory only and you can do that or in many or in some other cases what is done is this a bar line equal to it will be made equal to one. So, the when it is accessing the internal memory, that is made to hold the

operating system code monitor programs and all that and the user programs are all in the external memory. So, it is also done that way.

(Refer Slide Time: 22:14)



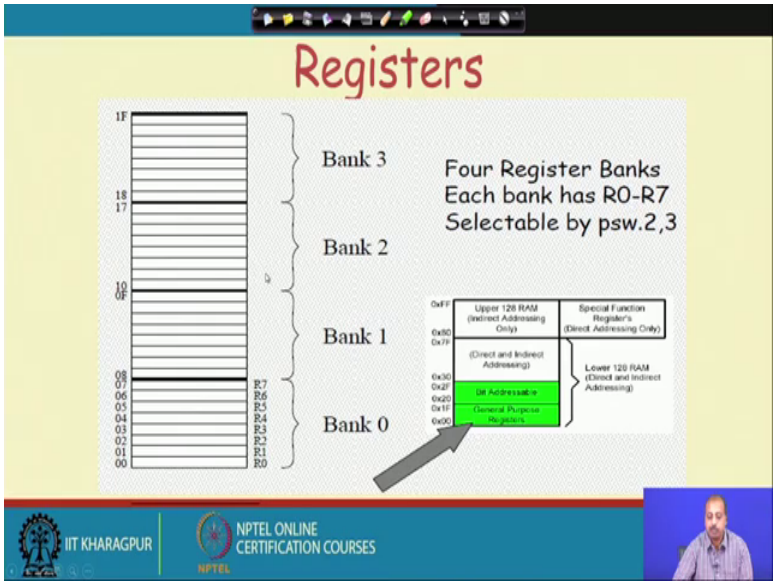
So, if we look into this one chip memory in more detail. So, you can find that. So, there are there is a total we have got two fifty six Bytes. So, out of that, 128 Byte that is 0 to 7F, we will have some special feature and remaining 80 to FF. So, we will have some other feature first of all 0 to 1F. So, we have got some say general purpose registers. So, they are the CPU registers. So, unlike your microprocessor like 8085, we do not have separate CPU registers because now the memory that the internal memory is also a part of the of the chip. So, there is no point keeping something in the CPU separately compared to the memory.

So, that is why this registers. So, they are nothing, but some locations in the internal RAM. So, 0 to 1f. So, that is we have got this general purpose registers then the location 20 to 2F they are they are bit addressable. So, int you can you. So, you can access individual bits as we know that the memory has got a word size. So, that word size is in whatever we have discussed they are eight bit word. So, if you are accessing a location means you are accessing eight bits at a time, but in many cases we want to access individual bits separately and for that purpose we there is a portion of memory which is bit addressable.

So, this 20 to 120, if there are some bit numbers will be given. So, we will see that. So, this location 20 is eight bit wide and each of this locations can be accessed separately they can be set and reset separately then 30 to 7F we have got this is the scratch pad that we have. So, this is for this is for some other purpose. So, you can use them as temporary locations sometimes we use them as stack location etcetera. So, it is done like that and after that we have got some eighty to f those locations.

So, upper 128 RAM. So, for, that is some extra locations that we have. So, this where is this lower part? So, you can use both direct and indirect addressing. So, we can have only indirect addressing here or we can have direct addressing for the special function registers. So, we will explain them as you proceed.

(Refer Slide Time: 24:48)



So, this is the first part the CPU registers that we talked about. So, the registers, we there are actually 32 registers and this each this thirty two registers they are divided into four banks Bank 0, Bank 1, Bank 2 and Bank 3 at any point of time only one bank of registers is active.

So, this is the lowest bank having the registers R0 to R7 then we have got bank one going from R0 to R7. So, like that we have got different registered banks. So, this is the general purpose register Bank 0 to 1F and at you can control this bank selection by two bits in the processor status word register which are known as PSW 2 and 3. So, PSW is

the processor status word register it is bit numbers 2 and 3 they can be used to select the registered bank.

So, this is helpful because what happens if previously while discussing on 8085 we have seen that if we are going from one program we executing on program and an interrupt occurs then it is the responsibility of the interrupt service routine to save all the registers that it is going to use in the body before doing any operation because the if program which got interrupted may be using those values and again while going back it should restore all those register values before the write 10 instruction.

Now, that is costly because. So, many steps are to be done. So, in this case in 8051 we have got the flexibility that we can just switch between the banks may be the main program is using Bank0. So, when we are in the interrupt service routine. So, we can simply switch over to Bank 1. So, that all the manipulations that this is the interrupt service routine will be doing. So, they will be affecting bank one registers. So, Bank 0 registers they will remain unaltered

So, later on while coming back from the sub program or sub routine, you can just switch the banks again. So, that the original bank gets restored. So, this helps in the context switching process. So, context switching becomes faster. So, since embedded applications, we want to have faster contexts which to respond in real time fashion. So, this is help full.