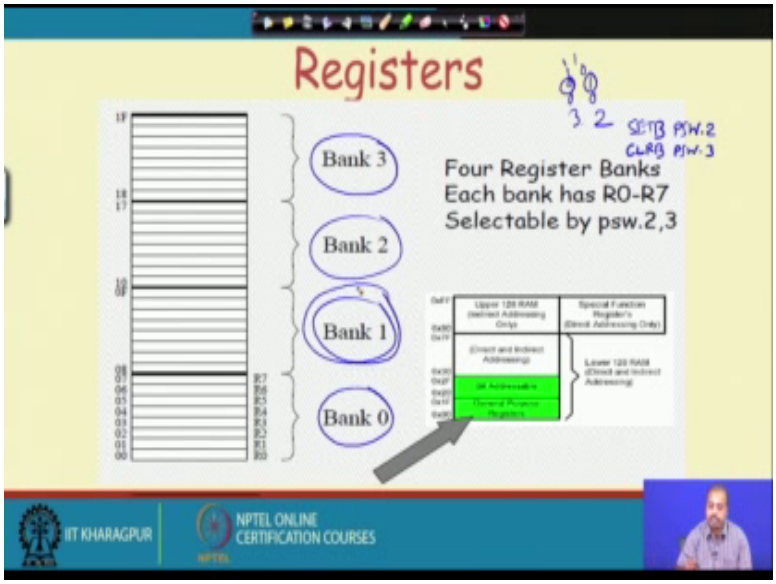
Microprocessors and Microcontrollers
Prof. Santanu Chattopadhyay
Department of E & EC Engineering
Indian Institute of Technology, Kharagpur

Lecture – 26
8051 Microcontroller (Contd.)

We can control this bank selection by this PSW register.

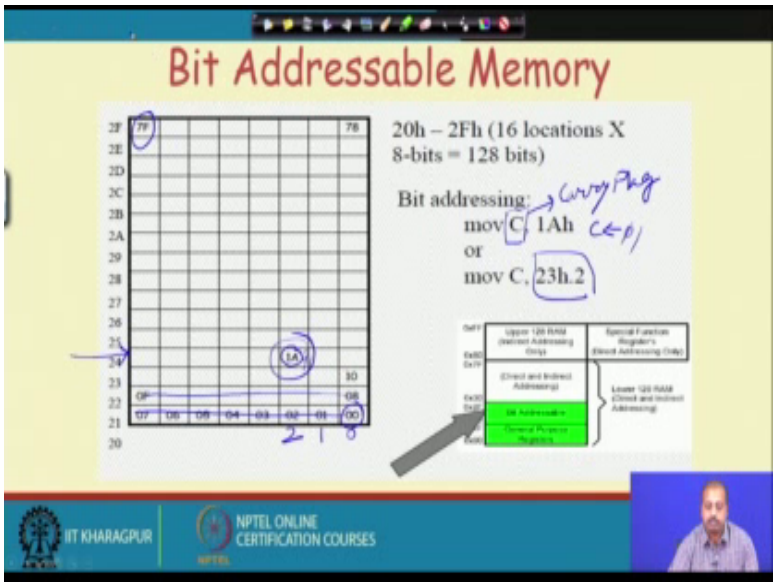
(Refer Slide Time: 00:20)



So, well see this processor status over register later, but what happens is that you can we have got individual bit set reset type of instructions in 8051 and this bank selection so this this 2 bits 3 and 2 3 and 2. So, they being 0 0 it will select this bank 0 this 2 bits being 0 1 will select bank 1. So, 1 0 will select bank 2 and 1 1 will may select bank 3.

So, you can simply have the instruction like SETB PSW 0.2. So, that will be setting the bit of PSW 2 and then if you use say clear bit PSW dot 3. So, what I am doing I am setting these bit as 0 and these bit as 1. So, effectively I am selecting this bank 1. So, this way we can use this PSW bit settings for selecting or switching between the banks. So, next we will look into this bit addressable memory locations.

(Refer Slide Time: 01:31)



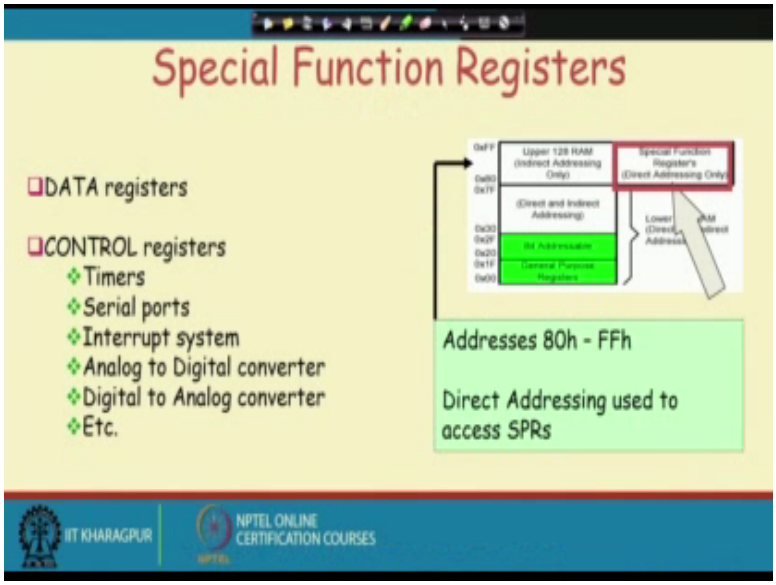
So, there are 128 bits location which can be accessed individually. So, their physical memory address is 2 0 to 2 F I am sorry there is a shifting here this 2 0 is actually here. So, there is a misalignment of this numbering. So, this to 0 is here then 2 0 location to zeros bit number 0. So, that is given the location address 0 0. So, this bit number is. So, if you say if you if you are writing like set bit sorry if you are writing like set bit 0.

So, if you write something like set B 0. So, what it will do. So, this particular bit will be set to 1 similarly if you set say like set B say 5. So, this particular bit will be set to 1. So, this way we can access individual bits in 8 0 5 1 instruction. So, we will see them in detail later. So, like say moves moves C comma 1 A hex. So, what it will do this C stands for the carry flag. So, this is the carry flag. So, this is the carry flag and you see this if you count from the here. So, 0 0 1 words like this. So, this location becomes this location address is 1 A this particular bit address is 1 A.

So, this carry will get the value of this particular bit. So, if this bit is say 0 then carry will get a 0 if this bit is 1 then the carrying will get a 1 or you can. So, you can write the same thing in this fashion also. So, 23 x dot 2. So, this is the location 23 as I said there is a misalignment of lines numbers. So, this is the location this is the memory location 23 and then 20 threes bit number 2. So, this is bit number 0 1 2. So, bit number 2 we want to access. So, that is also the other way by which we can do this we can access this individual bits.

So, this way this we have got bit addresses running from 0 0 to 7 F total 128 locations we can use for in the in the bit addressable in this bit addressable fashion. So, they are very very much useful like many of the interfaces that we have in embedded applications. So, they have got a digital input output and these digital inputs outputs are being say a single bit input. So, they can be very easily put into associated with these locations.

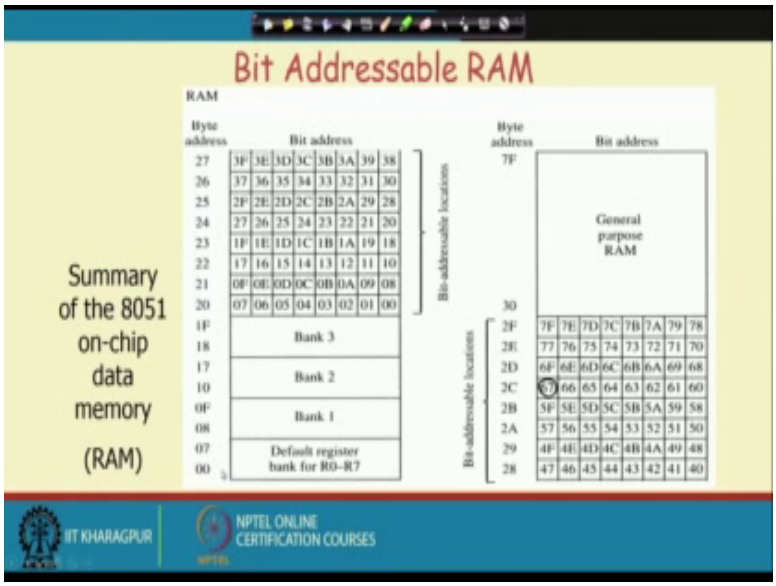
(Refer Slide Time: 04:21)



Then well look into the special function registers. So, there are many special functions registers ranging whose address is 8 0 to FFh. So, they are used for direct addressing and. So, we have got data registers and control registers. So, we will see what are they. So, they have got in different context they have been used as I said that 8051. So, it has got within a timers already built in timer counters then serial ports are there then interrupt mechanism is there then some of the advanced version of 8051 they have got ad converters da converters also integrated.

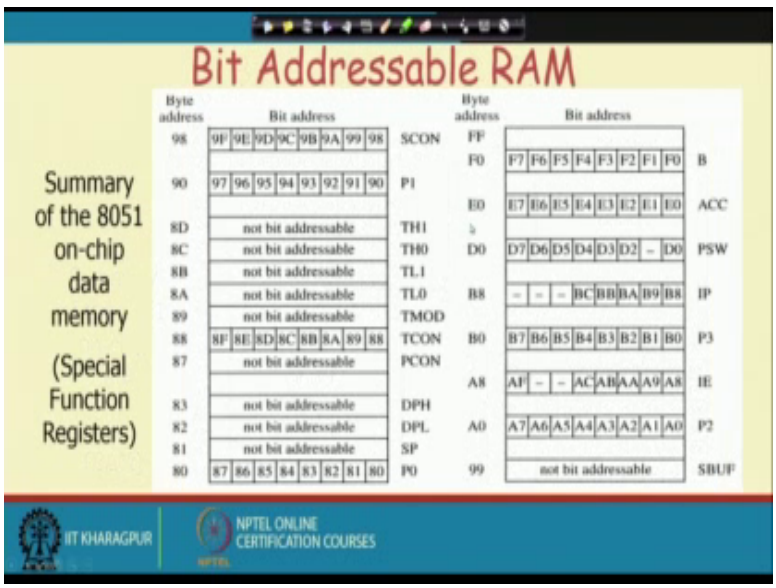
So, controlling those peripherals so, we need to have some additional registers for setting for controlling them and checking the status of them. So, we need some additional registers. So, they are the control registers. So, this special function registers. So, we will be using we will see that there are a number of control registers. So, throughout our lecture we will find that many such control registers will be mentioned and this 3 0 to 7 F. So, this is basically the ram space. So, you can use some you can store some temporary data there you can use them for stack. So, all these can be done ok.

(Refer Slide Time: 05:37)



So, what is happening? So, bit address overall summary of this 8051 on chip memory. So, you see that if you start with 0 0 first we have got the registered banks 0 0 to 1 F. So, is the first we has the default registered bank R 0 to R 7 then bank 1 bank to bank 3 then to 0 to 2 f we have got this bit addressable locations. So, stand bit number 0 0 to 7 F after that from 3 0 to 7 F, we have got this general purpose RAM. So, this is the general purpose. So, this is the scratchpad sort of thing there are they can be they are not bit addressable they are byte addressable only, but we can use them for storing some 8 bit values there.

(Refer Slide Time: 06:23)



Now, if you look 880 onwards. So, if you look into 8 0 onwards then what happens is you have got this these are the special function registers. So, some of the special function registers are like every port. So, you remember that there are 4 ports in 8051 P 0 P 1 P 2 P 3. So, P 0 is associated with address 8 0 then P 1 is associated with the address 9 0 P 2 associated with A 0 and P 3 associated with B 0. So, we have got these ports also they are they are also associated in memory location.

So, in your instructions so you can either say port 0 or you can say memory location 8 0 they are all the same similarly you can say P 1 or memory location 90, they are all the same. Now this stack pointer so this 81 so location 81 so that is that corresponds to the stack pointer register then 8 2. So, is either DPL and 8 3 is the DPH so this DPH DPL pair. So, this makes that DPTR register for external memory axis and they are not bit addressable. So, you see that these locations. So, they are not bit addressable then this location is not documented what it is doing now 8 7 is the PECON register. So, this is for power control. So, you can put that 8051 in the power down mode by setting this PCON bits accordingly.

So, you can use this PCON register there then this TCON is another register which is used for timer control timer and timer and interrupt control then TMOD is for the timer mode control. So, these are all different register then this TL0 TL1 TH0 TH1. So, these they are for timers. So, this TL0 and TL1 they hold the timer value of our timer 0 and TH0 and TH1 they hold the current time value in timer 1. Then this SCON register is for serial communication control.

So, this bit. So, the what does this bits mean. So, we will see that later when you go to this individual portion. Then S buff is also for serial communication and what happens is that if you want to transmit something serially you have to copy it into S buff after setting this SCON properly. So, if you copy it into S buff then the character will be sent serially through the transmit bit and TXD and RXD lines.

Then this is P 2 this is interrupt enable register. So, this interrupt enable register. So, this is for enabling interrupt. So, we will see that later again then IP is the interrupt priority. So, you can modify the priorities of different interrupts in 8051 there are a number of interrupts like this the 2 timers. So, they have they can give interrupt then the serial

transmission. So, that can generate another interrupt then we have got this INT 0 and INT 1 pins. So, they can also generate interrupts.

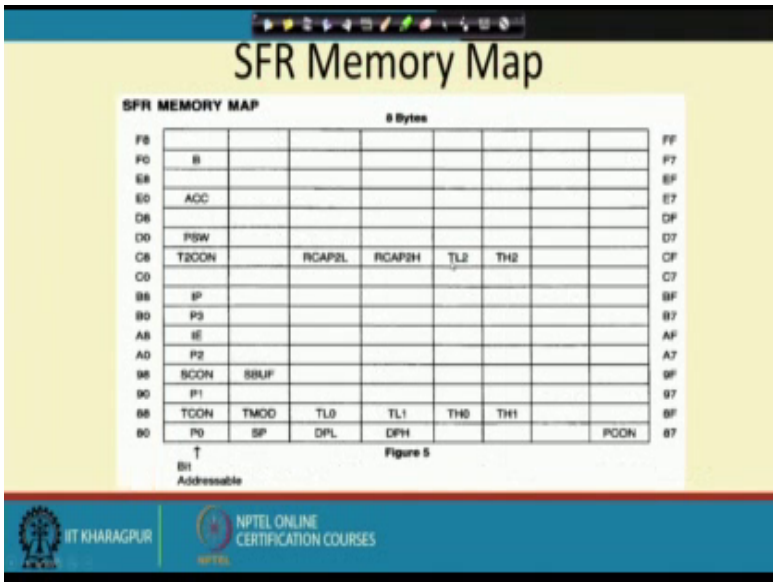
So, that way we have got a number of sources of interrupts and these interrupt priorities can be modified there is a default priority, but beyond that you can also modify the priority then this PSW is the processor status word. So, this D 0 to D 7 they have got some special meaning when we go to PSW register we will see then this accumulator ACC. So, this has got E 0 to E 7. So, these are the addresses. So, you can you can write this you can get this value in the accumulator as an 8 bit pattern or you can access as individual bits.

So, you see that bit number. So, P 0 is at location 880. So, the bit numbers it represents is 8 0 8 1 8 2 8 3 8 4 like that up to 8 7 after that TCON is bit addressable and it starts that location 8. So, it is at location 8 8 and the bit numbers also start with 8 8. So, in this way at whenever we have got whenever we have at a boundary of 8. So, that at that point we have got this bit addressable feature like we have got the bit addressability at location 8 0, then we have got bit address ability at 8 8, then at 9 0, then 9 8, then A 0 A 8 B 0 B 8.

So, then C 0 is of course, not there then we have got this D0. So, after B 8 you see that there is a nothing. So, B 8 to B F if we say then there is nothing like C 0. So, it start with D 0 D 0 to D 7 and then E 0 E 0 to E 7. So, bit numbers are the same.

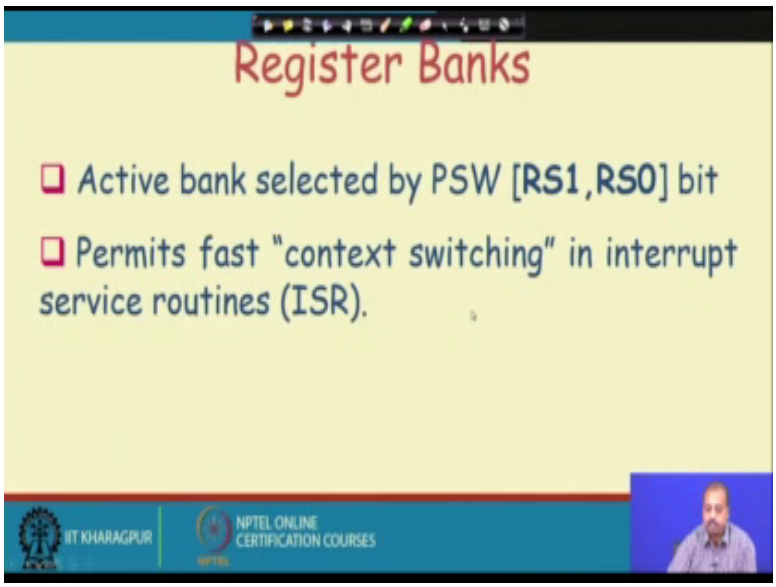
So, these are the byte address and these are the bit numbers they are the bit addresses.

(Refer Slide Time: 11:17)



So, this special function memory. So, you can we can just review it once more. So, this 8 0 is for P 0, then this 9 0 is for 1, then this B 0 is for P 3 and. So, then P A 0 is for P 2 like that then this SP is at 8 1, DPL is at 8 2. So, the same diagram that we had previously. So, that is shown in another fashion here.

(Refer Slide Time: 11:52)

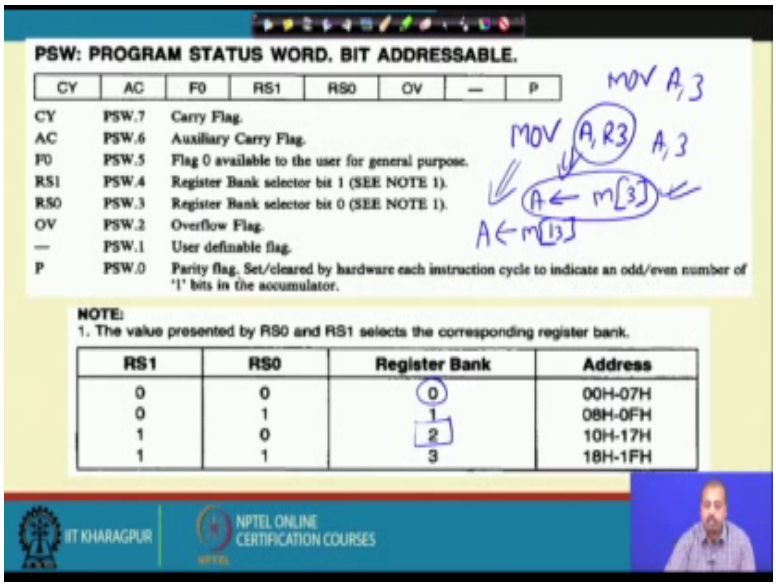


So, this registered banks. So, as we say that the active bank is selected by PSW RS1 and RS0 bits. So, this register bank select 1 and register bank select 0 bits. So, this will permit context switching in interrupt service routines. So, as I was telling that you can

switch the context very fast. So, without storing all the registers that the ISR is going to use you can just switch over to the switch over the bank and if you follow a policy that all the main all the interrupt service routines or a will be using the registered bank say 2 and the main routine will be using registered bank 0.

So, that way there is no confusion in the saving of registers. So, that is not required at all. So, you can just switch the registered bank.

(Refer Slide Time: 12:46)



So, if you set this R S 1 and R S 0. So, this is the bit R S 0 and this is the bit R S 1. So, 0 1 2 so this is the PSW register. So, with this is bit 0 1 2 3 so and 3 and 4 so PSW 3 and PSW 4 so previously we said that it is 2 and 3, but it is actually 3 and 4 though. So, PSW 3 is the R S 0 bit and PSW 4 is the R S 1 bit.

So, this is this this is if this 2 bits are 0 0 then it will be accessing it will be setting registered bank 0 as the current registers 2 current set of registers to be used similarly these 2 bits are 0 1 then it will be using this bank 0 1 and the corresponding addresses will be 0 0 2 0 7. So, if you are writing an instruction like say move sorry if you are writing an instruction like say move a comma up something like if you are writing like say move a comma R 3 then depending upon the current registered bank selection. So, it will be accessing 1 of the locations like if the current bank is say registered bank 0 then when you say R 3. So, this is actually will be accessing the register. So, this registers as

we know that this will be starting with the this registers will be starting with R 1 I think yeah.

So, register starts with number R 0. So, if you say R 3 then the corresponding number that we have is so R 3 so the 0 1 2 3. So, this is the location memory location memory location 0 3. So, that will be used here so this instruction. So, this will get the into the accumulator content of memory location 3. On the other hand if you are if provided you are using the registered bank 0, if you are using registered bank 4 in state then the same instruction. So, this will be taking the accumulator will get the content of memory location that is that is 1011 1213 it will get the content of memory location 1 3.

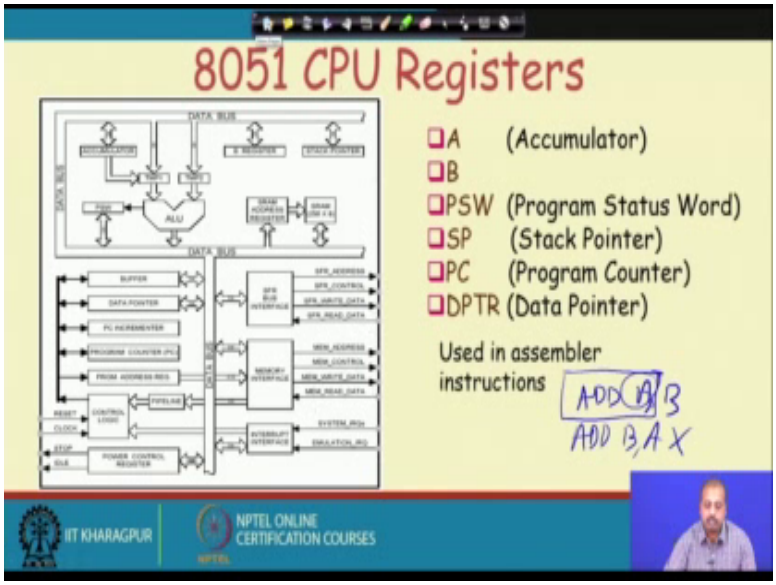
So, this way can see that depending upon this registered bank selection. So, you can do it like this. So, you can also write it as instead of writing like a comma R 3 you can write like move A comma 3. So, when you say A comma 3 then the register bank is not in question. So, in this case it will execute this 1 only. So, the move instruction when you are writing the move instruction you can write either in this 4 move A comma R 3 in that case depending upon the registered bank selection. So, it will either be accessing say location member the 3 or the location 13 in our example; however, if you are using the instruction like move A comma 3 move A comma 3 then whatever be your bank selection. So, it will move the content of memory location 3 to A.

So, it is the registered bank independent. So, that way we can have these registered banks useful apart from that we have got this P as the parity bit. So, that is set or clear by hardware each instruction cycle to indicate odd even number of ones in the accumulator. So, that that can be useful for parity check then this is flag the user can put some values there. So, this is also bit addressable, but it is not used. So, you can use it for your own purpose. So, you can use it as A 1 bit location.

Then this OV is the overflow flag then this R S 0 and 1. So, they are for the registered bank selection F 0. So, this is available for the user for general purpose. So, this is also a. So, you can use it for your own purpose those are name is F 0, but it is available then this ac is the auxiliary carry the if from the when you are doing say addition operation or the subtraction operation. So, from bit number 3 if a carry is generated towards bit number 4 then this auxiliary carry will be set and the carry flag is CY. So, that is ah. So, if the

overall 8 bit addition or subtraction type of operation generates a carry then this carry flag will be set. So, this is the PSW register.

(Refer Slide Time: 17:58)



Now, other registers that we have in 8051 A or accumulator so, this is the most important register that we have and all these arithmetic logic operations. So, they will be using a register as 1 of the source and as well as the destination. So, you can say that. So, you can have an instruction like say add A comma B. So, in that case the content of A and B registers will be added and the value will be stored in A register. So, there is no instruction like say add B comma B. So, this is not there.

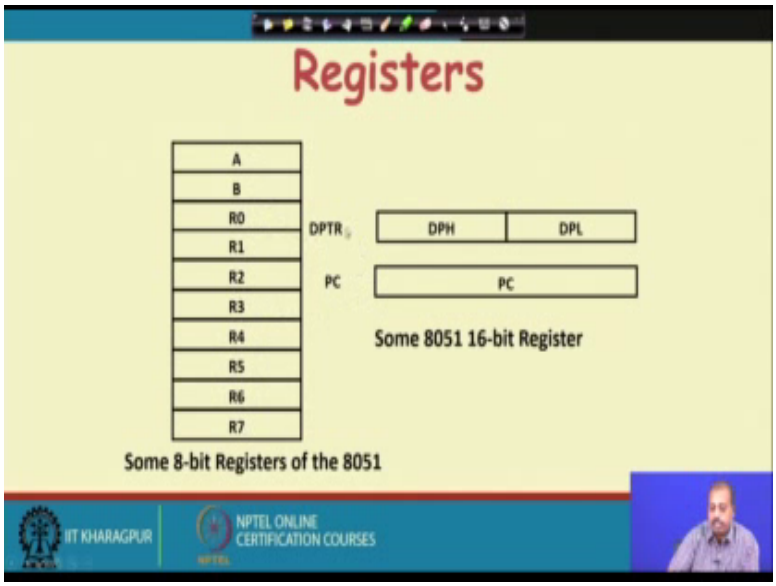
So all instructions so they will have a as the as the first operand. Now apparently it seems that why do we mention this a separately. So, this ADD A is actually the full mnemonic you can say. So, most of the instruction you will find that though we are mentioning a separately, but that is only for human understanding as far as the machine is concerned. So, there is nothing you say you cannot have after ADD you cannot have any other register name it has to be A ok.

So, this way this accumulator is 1 of the very important registers that we have then we have got this B register. So, this is another very important register that is there then the PSW is program status word SP is the stack pointer this is the stack pointer then we have got this program counter like say. So, program counter is there. So, this is the program counter register accordingly and there is a DPTR or data pointer. So, data pointer is this

1. So, DPTR so for program memory axis, it will go via this program counter and for data memory access it will go by this data pointer for a for the external memory axis. So, it will use the DPTR register for internal memory access it will use program counter or some addresses, but for external memory axis and external data memory axis. So, it will use the DPTR register.

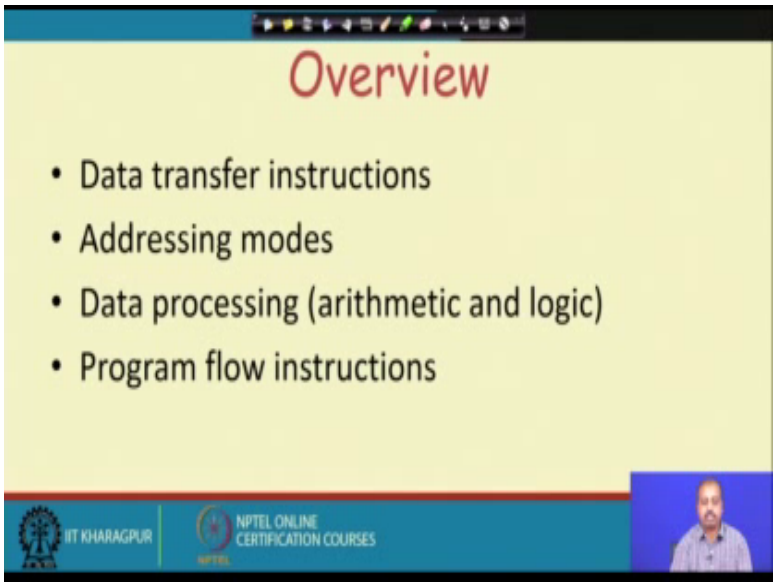
So, apart from that we have got these special function registers memory in terms of a memory interface. So, this address generator address data then control. So, they are there. So, they, but these lines are going, but they are actually all internal to the chip because this side I will be connecting some memory chip and some memory and that memory will have will also internal. So, that is not shown here. So, explicitly, but it is just here anyway.

(Refer Slide Time: 20:46)



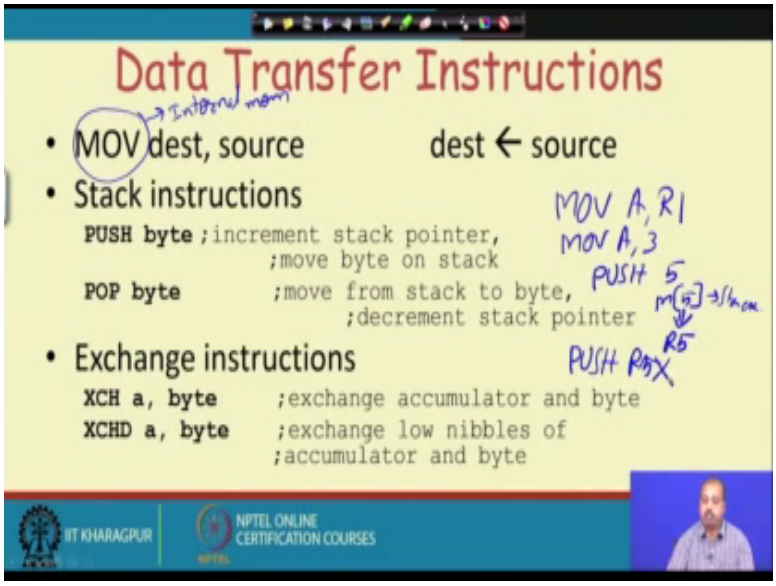
So, next so the to summarize the registers that we have is are like this we have got this A B registers plus the registers R 0 through R 7 then. So, and they are all 8 bit registers and the important 16 bit registers are like DPTR and PC. So, DPTR has can be considered as 2 8 bit registers DPH and DPL and this PC is a 16 bit register.

(Refer Slide Time: 21:15)



So, going towards the these instructions of 8051 the assembly language instructions of 8051. So, you can value we can visualize them in from different angles like the data transfer instructions, then the addressing modes in which this data are accessed, then the data processing instructions like arithmetic and logic operation and there can be some program flow instruction that will control like jump call this type of instruction which will control the operation of individual operation of the flow of the program.

(Refer Slide Time: 21:49)



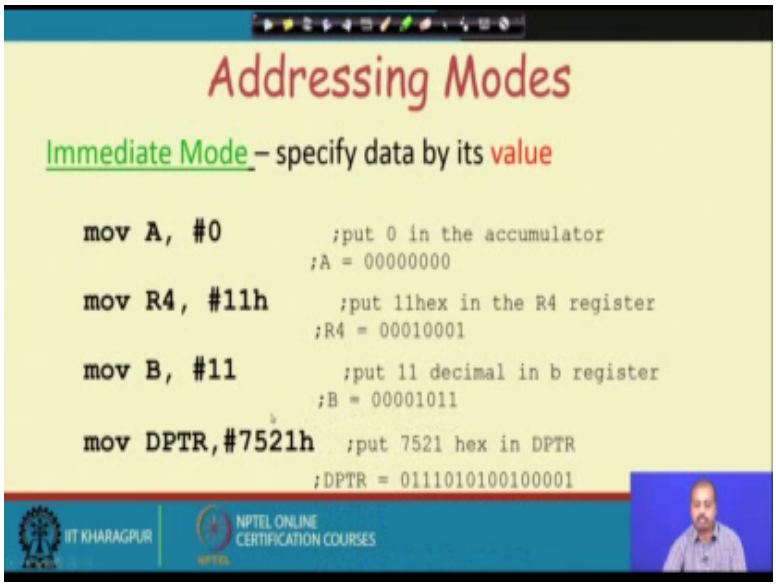
So, the first category is the data transfer instruction. So, 1 instruction maybe like move destination comma source. So, destination gets the value of source then we have got stack instructions like push byte and pop bytes. So, increment the stack pointer move byte on the stack and pop byte it will move the move from stack to byte and then it will increment the stack pointer. So, it is like this I think there are server.

So, this move destination comma source like you can have instructions like say move as I said I can have an instruction like move A comma R 1. So, R ones value is moved to a then we can have like move A comma 3. So, memory location 3 is content will be coming to A. So, they are so this move instruction. So, this is always with the internal memory. So, it does not access the external memory. So, they are all with internal memory then the stack instructions. So, stack instructions are like this.

So, you can have instruction like push 5. So, what it what it will do the content of memory location 5 will be pushed on to the stack. So, this will go to stack and this as we know that memory location 5 actually it corresponds to the register R 5 in the. So, if you are. So, it will give in the register bank 0. So, this is the register R 5. So, it will be saving the content of register 5 on to the stack provided you are using registered bank 0. So, if you are using some other registered bank then accordingly this number has to be modified, but we cannot write like push R 5. So, that is not possible. So, this is not possible.

So, we have to tell the byte address directly and that way it has to be done then there is an exchange instruction. So, that can be used for exchanging the accumulator with some byte so XCH a, byte. So, it will exchange the accumulator and byte and XCHD. So, it will exchange the nibbles of the accumulator and byte. So, we will see some example.

(Refer Slide Time: 24:09)

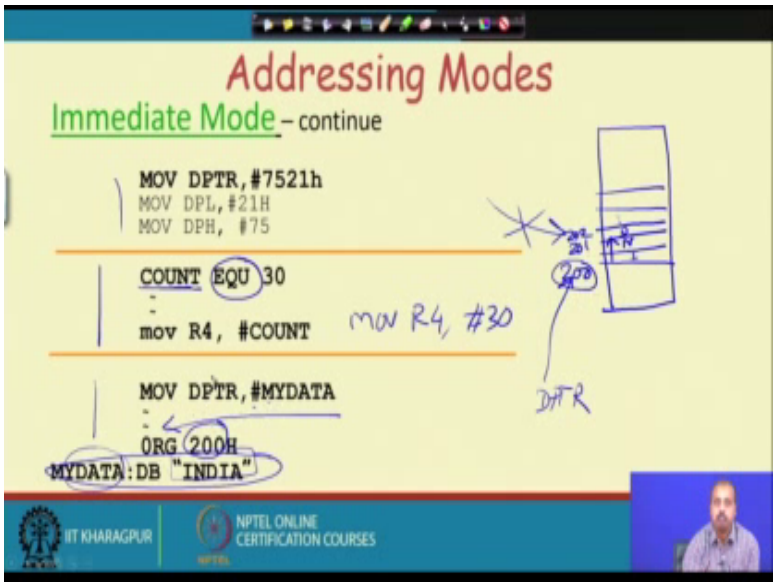


Like say move a comma has 0. So, that this. So, whenever we put A, hash. So, most of the assemblers we will take it as immediate value, if you do not put this hash then the meaning is we will get the memory location 0 content will come to the A register, but when you put this hash. So, we mean that it is the memory it is the immediate value 0 that will come to a register.

Then move R 4 comma hash 11 hex. So, this 11 hex will be put into the R 4 register move b comma hash eleven. So, your decimal 11 will be put in the B register again this is the convention followed by most of the assemblers that if you follow the number by 8. So, it will be taken as hexadecimal number if you do not write anything or if you write ad after the number. So, it will be taken as the decimal. So, you can also initialize that DPTR. So, you can say like move DPTR has 7 5 2 1 hex. So, this 7 5 will go to that DPH register and 2 1 will go to the DPL register as a result this DPTR will have the value 7 5 2 1.

So, this is the thing a gets 0. So, here R 4 gets the pattern 11 hex and in this case it gives the number 11 decimal and in this case DPTR gets the 7521 hex.

(Refer Slide Time: 25:38)



So, we can we can also put it in this format like say move do a move DPTR comma hash my data where my data is. So, this is defined as DB. So, this is defined as DB. So, this is. So, this is again used in some many of the assemblers as the as some space at which you can we are we are defining some constant data.

So, it is like this so this ORG statement. So, ORG statements means that the assembler will start, assembling the program, from that address onwards. So, this is just telling the assembler that the next program the next instruction or data whatever it is. So, it will be put at this particular address onwards. So, if this is the memory. So, you can say that at memory location 200 onwards it will be if this is the location 200. So, from this position onward so it will be putting the value in there.

So, I will be put here then if this is 2 0 1. So, N will be going there then this is 2 0 2 D will be going there. So, this way the numbers will be stored that individual characters will be over. So, DB stands for define byte. So, this my data DB India. So, this will be defining this 5 byte space and in that 5 byte space it will be writing the value 200. So, when you say that DPTR are move DPTR hash my data. So, this my data where if whatever be the address so 200 so the 200 will come to my data.

So, we can have this DPTR 7521 h this movement. So, it can also be executed like move DPL 21H and move DPH hash 75 H there should be A H here. So, that way we can do it and we can also do it like this that say this is another way of doing this like say this this

is this is another way of the move instruction like count EQU 30. So, EQU is another assembler directive. So, where I am defining a constant count whose value is equal to 30 and move R 4 comma hash count. So, this will this is same as the instruction sorry this is same as the instruction move R 4 comma hash thirty the same as this.

So, this EQ is another assembler directive that tells the similar that in the remaining program instead of writing constant that way. So, I may write I may use this symbol this name count and while assembling the program or generating the machine code for the program. So, wherever it finds the word count it will replace by the value 30 there.

So, this way we can. So, this is another way of doing the movement this is one way and also this is another way at which this DPTR can be initialized. So, we can have some the if DPTR has to point to say this address for external memory sorry not this one say 200 it has to point to this address. So, it can be done in this fashion. So, we define first this number this constant India and give it the name my data so later on when this when I am writing hash my data. So, this is basically the value of my data and value of my data is nothing, but the address from where the my data starts. So, that way this DPTR will get the value 200 in that case.