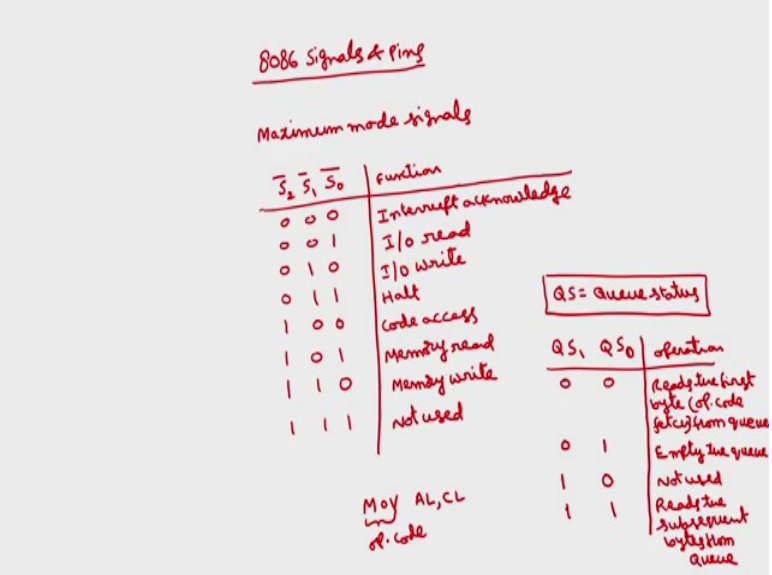
Microprocessors and Interfacing
Prof. Shaik Rafi Ahmed
Department of Electronics and Electrical Engineering
Indian Institute of Technology, Guwahati

Lecture – 05
8086 Maximum Mode Signals

So, in the last class we were discussing about the 8086 signals and pins.

(Refer Slide Time: 00:34)



So, we have discussed about this common 32 signals and then 8 exclusive signals for the minimum mode configuration. So, there are 8 exclusive signals for the maximum mode configuration. So, the 8 signals are S2 bar, S1 bar, S0 bar. So, the function of this one will be 000. So, it performs interrupt acknowledge operation. So, whenever interrupt occurs to the microprocessor.

So, during the acknowledgement of that particular interrupt S2 bar S1 bar and S0 bar will be 000, if this is 001 then this represents IO read operation. So, we know that we have microprocessor can perform 4 I mean microprocessor initiated operations. So, out of this 4 one is memory read, memory read, IO write, IO read ok. So, for 001 this is I O read, for 010 this is IO write, for 011 this is halt operation and for 100 this is code access; it access the code, 101 memory read.

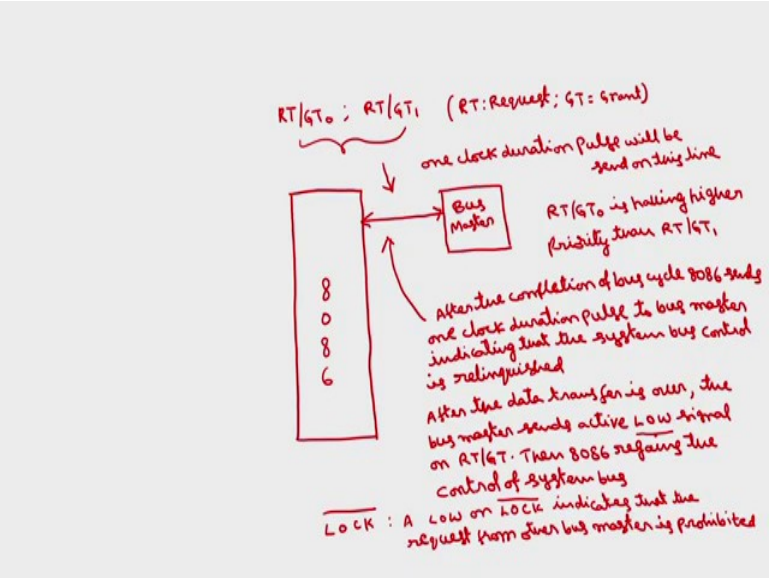
Now, this is IO read and here it is memory read 110 memory write because 111 is not used. So, this is the function of the 3 maximum mode signals S2 bar, S1 bar, S0 bar they are active low signals. Then coming for the other signals so, we have Queue status 1, Queue status 0 signals. So, QS stands for Queue status. So, we know that we already discussed in the architecture 8086 contains 6 byte instruction queue. So, by using this instruction queue we can overlap the fetch and execute operations thereby you can increase the speed of the operation which is called the pipelining.

So, the status of this queue will be indicated by this QS1 and QS0 So, these 4 combinations of the QS1 and QS0 be the corresponding operations are as follows, 00 represents the first byte read the first byte which is called opcode fetch that is opcode fetch from queue. So, always you take any instruction you take any instruction the first term in byte will be opcode fetch the code correspond each instruction for example, if I write say for example, MOV AL, CL.

So, correspond to this MOV there will be a code which is called opcode. So, this is fixed for this 8086 microprocessor there is a format to generate that code op code. So, always in order to execute any instruction the first byte will be the opcode, the next byte will be the registers or it can be data it can be address.

So, when this QS1, QS0 are 00 it reads the first byte which is by default op code fetch from the instruction queue, 01 it empties the queue, 10; 10 is not used, 11 read the subsequent bytes from the queue after this first op code fetch, then it may require the subsequent bytes. There are some instructions where only the opcode fetch is there. So, in that case it will not read any subsequent bytes. So, this is about the queue status signals. So, we have discussed about the 5 signals. So, remaining 3 signals of maximum mode operation of 8086 are.

(Refer Slide Time: 06:47)



There are 2 signals called request by grant 0 and request by grant 1 this RT stands for request, GT stands for grant. So, we know that in case of maximum mode operation it is a multi user configuration. So, several users can use the same microprocessor. Suppose, if I have 8086 microprocessor, suppose if any external bus master want to use this address buses.

I mean data bus, in general, the system bus which is called a system bus together address bus, data bus together is called as system bus. So, whenever this bus controller or bus master external bus master wants to use the address and data busses of this 8086 microprocessor, it will request through one of this signals either this can be anyone of this signals through this signals it will request.

So, in order to request this what it will do is it will send one clock duration pulse will be send on these line. So, this can be request by grant 0 or request by grant 1. So, once what happens is between these 2 so, RT /GT0 is having higher priority than the other one. So, whenever the request the simultaneous request comes on both the signals, then this RT by GT0 will be served first that is meant by the higher priority. So, whenever this bus master request through this RT /GT bar through one clock duration pulse this should be one clock duration pulse.

Then what this microprocessor will do is it will complete the execution of the current bus cycle current operation after that, this will inform this bus master through this same

signal; through same signal using one clock duration pulse. Saying that the 8086 is going to relinquish the system bus to the bus master this is after the completion of the current bus cycle, 8086 sends one clock duration pulse to bus master indicating that the system bus; system bus means both address and data bus control is relinquished.

Now, the bus master can use this system bus for it is own purpose. So, whenever this data transfer is over, after the data transfer is over, the bus master sends active low signal on RT/GT this can be RT/ GT0, RT/ GT1 after that the 8086. Then the 8086 regains the control of system bus. This is the complete process which takes place for request by grant 0 or request by grant 1 by external bus master. So, while requesting it will be one clock duration pulse. Similarly the acknowledgement will be on the same line this is bidirectional.

So, on the same bus the 8086 after completion of the current bus cycle it will again send one clock duration pulse to the bus master indicating that the control of the system bus is relinquished to the bus master. Now, the bus master can use the system bus for transferring of the data, once the transfer of the data is over then the bus master will send active low signal it is not one class duration it is just for only for short duration.

So, low signal on the same RT /GT line so, then the 8086 regains the control of the system bus. This is the entire process for request in the system bus of 8086 by the external bus masters ok. So, between these 2 RT/ GT naught will be having higher priority ok. So, if both the request comes simultaneously the 8086 grants RT /GT0 request.

So, while the bus master is requesting this so, the other bus master to avoid the other bus master to request for the system bus we have a another signal which is having maximum hour signal which is call LOCK bar signal. A low on LOCK bar signal indicates that the request from the other bus master is not granted is prohibited means already so, this 8086 is processing the request of one bus master.

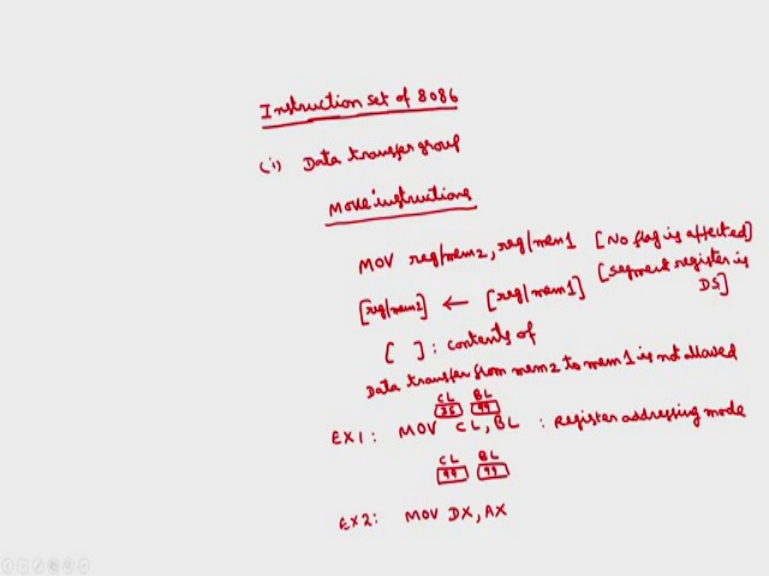
So, at that time if another master request for the system bus it simply places a active low signal on the LOCK bar signal indicating that bus master that the second bus master that the request is prohibited until the service of the first bus master. Once the first bus master uses the system bus and after the data transfer it will give the control to the 8086 that time LOCK bar single will be high so, that the second bus master can request for the

same system bus. So, is there about these 3 signals RT/ GT0, RT/ GT1 and LOCK bar and we have here 5 signals, S2 bar, S1 bar, S0 bar, QS1, QS0.

So, see about the maximum mode signals of 8086. So I have discussed about the complete architecture of 8086 the functions of all the signals ok. So, this is what about this architecture of 8086, then coming to the second portion of this course. So, one portion is architecture then instruction set using this instruction set we can write some programs starting from simple programs to some complex programs, later the third parties how to interface external devices such as IO and memory. How to interface to the 8086.

So, this total course is divided into 3 parts architecture part, then instruction set followed by the programming, then interfacing. So, now, coming for this instruction set of 8086.

(Refer Slide Time: 16:19)



So, the total instructions of 8086 can be divided into several categories. So, like we have this data transfer group, mathematic group, logical group, some string operations ok. First I will start with the data transfer group. The data transfer group of instructions. So, in the data transfer group one of the important operation is MOV operation, MOV instructions. The general format of a move instruction is. we can have the mnemoic will be the code will be MOV.

MOV So, here we will be having register slash memory 2 comma register, slash, memory 1. Simply here the operation will be the contents of register slash memory 1 will be transferred to the contents of register slash memory 2 ok. So, in micro processer terminology this represents this square bracket represents contents of.

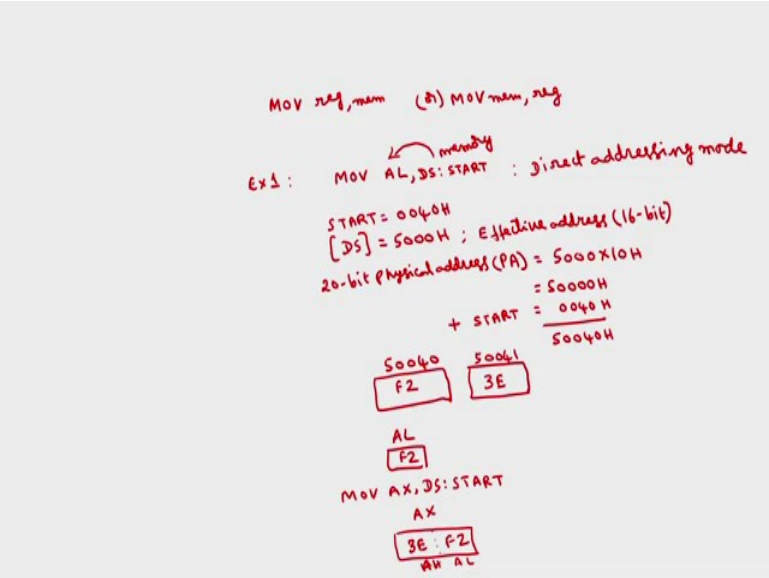
See here there are some 4 possibilities register to register, register to memory or memory to register. The fourth combination is not allowed memory to memory transfer is not allowed. Data transfer from memory to memory, memory 2 to memory 1 is not allowed. So, for that you have to transfer the memory contents into some register then from register to again another memory. So, direct transfer of data from one memory to another memory is not allowed using this instruction. There are some string instructions where we can transfer a string of the data from one set of the locations to other set of the location that I will discuss later.

So, here always the first register or memory will be destination and this will be source. So, if I take some examples so, we have 3 possibilities here as I have told register to register, memory to register or register to memory. So, if I take example, all the examples. MOV CL, BL this is register to register. So, before the execution of this instruction if CL contents is 35 H and BL contents is 99 H. Simply the contents of BL will be transfer to CL means CL contents becomes after this 99 H, but BL contains remains same 99.

So, here this actually this particular instruction belongs to register addressing mode here because both source and destination are registers. So, later I am going to summarize this addressing modes, this is called register addressing mode, here both are registers. This is 8 bit registers we can have 16 bit registers also MOV DX, AX. So, simply the contents of AX will be transferred to the DX, the contents of AX remains same. So, these are the examples of register addressing mode. We cannot transfer 8 bit so I mean between this source and destination one is 8 bit register, another is 16 bit register this is not possible the size of both the registers must be same.

So, in memory to register to register data transfer the sizes of both the registers must be same. Coming for the second possibility this is register to register, the second possibility is we can transfer from memory to register or register to memory, I will discuss both.

(Refer Slide Time: 21:11)



MOV register to comma memory or you can have MOV memory to register. Now, how to represent the memory? Registers we have different combinations we have 8 bit registers such as AL, AH, BL, BH, CL, CH like AX, BX, DX we have different registers. What about the memory? Memory we can have 1 mega byte or the memory total memory.

Now, from which memory location you want to transfer the data to the register. So, we can represent this memory location in different ways. So, depending upon this one so, we have different addressing modes. So, the memory can be directly we can represent in the instruction itself using data segment. So, the examples of this different this memory to register transfer or register to memory transfers will be if I write something like MOV AL , DS some START: DS or DS :START. So, in all this memory in that all MOV instructions by default the segment register will be DS.

So, by default the segment register will be DS here default segment register will be DS and none of the flags will be effected here, we write here that. So, for this MOV instruction none of the flags will be effected and segment register is DS. These are 2 important points for MOV instruction, see here what happens is START is a offset this we have to define by using assembler directives. So, initially we can assign this some value to START is defined as is equal to some 0040 H, if the contents of DS.

So, we know that DS is a 16 bit register which holds the upper 16 bit starting address of the data segment register, which is called effective address. If this is a say for example, 5000 H. Then the physical address, this is called effective address effective address is always 16 bit, then 20 bit physical address will be I will call as PA will be we know that we already discussed in the earlier classes 5000 x 10 H is the just DS segment, this will be equal to 50000 H.

So, if you have the start value of 0040 H the meaning of this one will be the address from where the memory location from where the data will be transferred to the AL, this is basically this transfer the data from memory location to AL, this is memory. Which memory location, what is the physical address of that memory location? To compute this, so, what is the data segment contents is 5000. So, the physical address will be 50000 plus you have to add START, the meaning of this one is data segment physical address plus START value so, plus if you start if I add START value which is equal to 0040. So, the total address becomes 50040 H.

Now, this instruction if the contents of 5000 H is 50040 H is this 8 bit some F2 H then after the execution of this instruction for this particular data. So, after the execution of this instruction AL will be loaded with F2 H. For example, if I write here instead of writing AL if I write MOV AX ,DS slash START, if I assume the same values of DS and START. Then what happens is here because the destination register is a 16 bit register, if the destination register is 8 bit register it will just simply read a byte from this location.

If the destination register is 16 bit register it will read a word is nothing, but 2 bytes. So, from which location, so whatever the first location that we are going to get that will be transferred to AL out of AX. So, we have 2 parts this is AL and this is AH. So, this contents F2 will be transferred to AL, the contents of this 50041 H let this is something like 3E this will be transferred to the AH. Depends upon destination register is 8 bit or 16 bit it will read a byte or word from the memory locations.

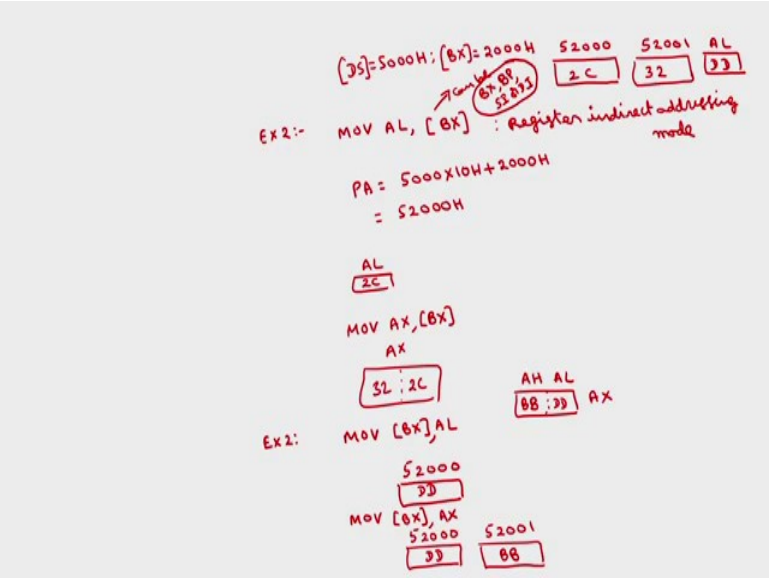
So, the starting location will be this if it is a byte only this contents will be transferred to that particular destination 8 bit register. So, if this destination register is 16 bit register it will take one byte from here, it will transfer to the lower order register of that 16 bit register and this address plus one, contents will be transferred to the higher order 8 bit register of that AX 16 bit register, so this is about this address.

So, this actually this belongs to a direct addressing mode; here in case of direct addressing mode the address of the memory location will be specified in the instruction itself, the register which contains this address is specified in the instruction itself, this is one way to represent the memory.

So, in any case it can be from memory to register or register to memory. So, how to I mean represent the memory location? Depends upon the way in which that we are going to represent the memory location, access the memory location, compute the memory location we have different addressing modes. This is direct addressing mode where the address of the memory location can be computed by taking the DS contents then multiplies 10 H, then add that start value which can be defined by using a assembler directive in the program.

So, that will be physical address 20 bit address so, from this we have to transfer a byte or word depending upon there is the destination register is 8 bit or 16 bit. So, this is about direct addressing mode, there are several other ways to I mean represent the memory.

(Refer Slide Time: 29:07)



The second way to represent the memory is, this can be from either register to memory or memory to register. AL comma, now we will specify the contents of some BX register. Now, what will be the operation? So, in order to compute the physical 20 bit physical address here so, what is required is.

So, if I assume that before the execution of this instruction if the contents of data segment register is say some 5000 H and the contents of BX is some 2000 H, then the 20 bit physical address will be computed, physical address is given by now 5000 x 10 H plus the contents of BX which is 2000. So, this will comes to 52000 H this is the 20 bit physical address from where the contents will be transferred to AL.

If I assume that the contents of 52000 H here will be some 2C, then after the execution of this instruction the contents of AL will be 2C. Suppose if I write MOV AX ,BX contents of BX, if the contents of the next address is 52001 is some value 32 say is when after this the contents of AX becomes 2C will be the lower contents and 32 will be the contents of AH this is similar to the previous example. See here the address is not specified data segment by default it is there, but here you are specified some base register that is why this particular addressing mode called register indirect addressing mode.

So, in this we are going to specify the register which contains some address. So, from that address we are going to offset of the address , offset of the memory location from that physical address we are going to transfer the data indirect, where as in the previous case directly we have specified data segment in instruction itself. So, this is from memory to register, you can perform in a similar manner register to memory also.

In the same register indirect addressing mode if I write something like MOV BX, AL if I assume the same values of the registers and data segment register on the other registers. Then the operation that will takes place here will be if I assume that AL contents is here from DD H.

So, if I write this the physical address is 5200. So, after this instruction what happens is 52000 will be lowered with the contents of AL which is before the execution of this one I am assuming that DD. So, this can be either from the memory to register or register to memory, if it is 16 bit MOV BX, AX, if AL is this one if AH is also I am writing here same thing AH and AL this is AX if contents is something like AL of you assume that same DD if I assume that this is BB.

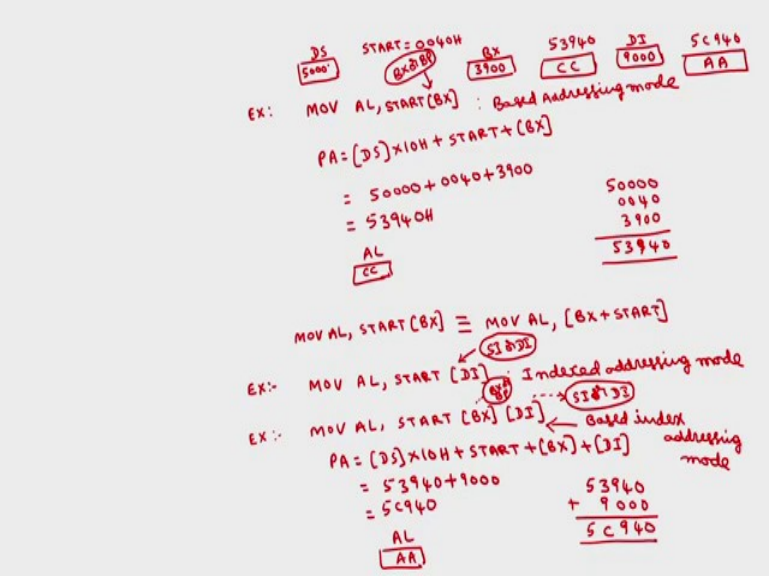
Then after this what happens is 52000 H will be having AL contents which is DD same as the previous one, then this place one address this place one which is 52001 will have AH contents BB. So, these 2 are examples of register indirect addressing mode in one case the transfer is from memory to register, in other case it is from register to memory.

So, here one thing that you have to remember is the register that we are going to specify here. So, this register can be this can be one of this 4, it can be BX, BP, SI or DI.

So, in order to BX that is BX is a general purpose register which can be used to store the data like AX in addition to that the general purpose of BX; BX also will be used as a base register in address calculations of this register indirect addressing mode. So, the possible registers here will be BX, BP which is base pointer I have discussed in this architecture, source index or destination index.

Then coming for the another example this MOV instruction is a very useful instruction in 8086 we can write several programs using MOV instruction. So, this a very strong instruction. So, we have lot of combination here. So, if I use another MOV instruction. So, this depends upon the how we are going to specify the memory address.

(Refer Slide Time: 35:10)



If I write something like MOV AL , START [BX] we can write. Then how to compute the physical address of this memory location? So, by default always we have to use data segment only, data segment if it is 5000 and if the START value is from 0040 H this has to be defined by the user by using assembler directive and if the contents of BX is from 3900 say.

So, we know that all these registers are 16 bit registers they contains only effective address, then the physical address will be computed using DS x 10 plus START plus BX.

So, this will come to here so, this will be 50000 plus 0040 plus 3900. So, which will be 50000, 0040, 3900, this will be 40935 so, this is the physical address of the memory from where the data will be transferred. Now, if the content of this location is 53940 is CC, then after the execution of this one AL contents becomes CC. So, this type of addressing mode is called based addressing mode.

So, in this we are going to write some offset then base register this register can be here either BX or BP only. In case of based addressing mode, the offset will be present in the register say that it is BX or BP and this START this also can be written as an assembler directive this instruction is same as MOV AL , START [BX] is also same as this is equivalent to MOV AL comma you can include this inside this BX plus START. So, this 2 are one the other same while writing the assembler directives you have to write in this manner.

So, similarly you can have this destination as AX then this location plus another location contents will be transferred to AX. So, if it is from register to memory again the contents of this will be transferred to this location, if it is AL, if it is AX contents of AL will be transfer to this AH will be transferred to this plus another memory location. So, here we can have source can be either register in memory destination can be either register memory so, this is called based addressing mode. So, here another addressing mode where index addressing mode same thing, but here instead of base register we will use index register.

So, this can be here either SI or DI source index or destination index this is called this type of addressing mode is called indexed addressing mode. Now, also same thing now the physical will be computed in a similar manner I am not going to give the examples same thing. So, DS into this plus start plus here instead of BX it will be contents of DI, you can compute this physical address from that address the data will be transfered to AL, if the destination register 8 bit, if the destination registration is 16 bit. So, this physical address contents will be transferred to AL this plus one will be transferred to AH.

So, the only difference is here instead of using base register we will use index register whereas, in the previous addressing mode register addressing mode this register indirect

addressing mode this can be all the 4 either BX BP, but here there is you know offset where as in this applications we have offset start here also START and START.

So, another example is you can use both also which is called based index addressing mode. So, if I write say MOV AL here also you can write this as DI plus START. So, this can be either BX or BP, this can be SI or DI. Now, here what is the physical address, how to compute the physical address? If I take this values start values is this DS is this BX is this DI is say for example, 9000 say, then what is the physical address from where the data address will be transferred to the AL.

So, all the all this MOV instructions we are going to I mean discuss about the different ways to access the memory location which memory location how to compute the memory location physical address of the memory location.

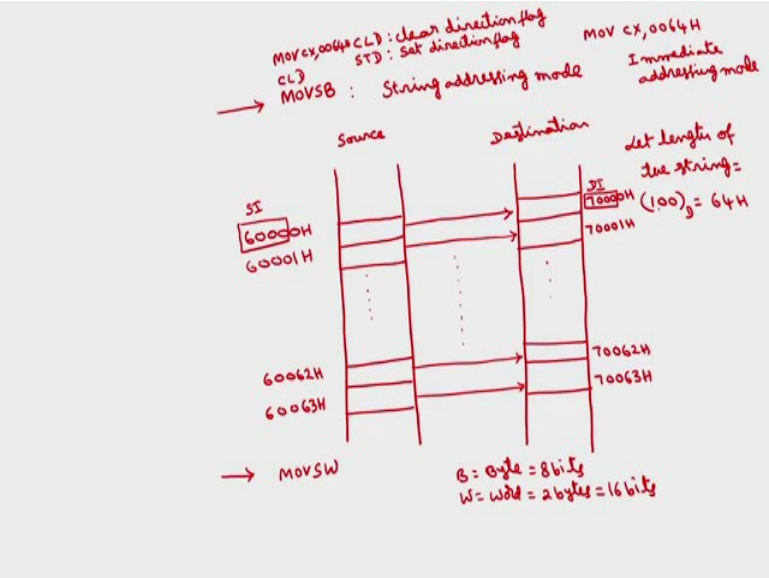
So, this is clear that the physical address will be now here similar to this DS x 10 H plus START plus BX plus DI. So, up to this we already computed which is 53940 H; 53940 H plus only extra additional term is a DI, DI content is 9000. So, this will becomes now 53940 plus 9000 this will be 049, 9 plus 3 is 12 in hexadecimal 12 is represented by C.

So, suppose if the before execution of this instruction contains of 5C940 is AA. So, after the execution of this, this is 5C940 after the execution of this instruction the contents of AL becomes the contents of this memory location which is AA. This is about based index addressing mode this stands for this stands for based index addressing mode.

So, here the different way to represent the memory location to transfer the data from either register to memory or memory to register. So, in any case how to compute that memory? The memory address there are different ways, we can computer using only the direct the data segment that is given here. Then we can add some offset or we can computer from using base registers or index registers only base registers and displacement index register plus displacement, base register plus index register plus displacement this is to compute the physical address of the memory.

Now, register will be specified in the instruction register can be 8 bit or 16 bit ok. So, these are all about the MOV instructions and we can have MOV string operations also. So, next MOV instruction is move string byte.

(Refer Slide Time: 44:32)



There is no memory there is no address, there is no register, there is no memory, so directly MOV string byte; this of course, I have explained while discussing about the directional flag ok. So, this MOV string byte by default it will assume that the number of bytes to be transferred from source to destination it is a string operation basically this belongs to string addressing mode. If want to transfer an array of data from some set of locations to another set of locations. So, this is a source, this is destination. So, I want to start from this location, I want to transfer some number of data to this destination locations.

So, I want to transfer this contents to this contents, this to here, this to here and this to here. Let us assume some locations say this is some 60000 H and this will be 60001 H. First to decide how many bytes you want to transfer, what is the length of the array, length of the string or length of the array. Let length of the array is string is equal to 100 decimal so, this will be 64 hexadecimal. So, I want to transfer 100 bytes starting with this address to this destination say destination this address is 70000 H, this is 70001H.

So, in the last address will be total I want to transfer 64 H locations in decimal this is 100 hexadecimal if you want to convert the decimal to hexadecimal it will be 64 H. So, starting with 0 the ending address will be 60063 H because 0 is starting point ending will be 1 less than this. So, the above address will be 60062 H and this will be 70062 H and this will be 70063 H. So, I want to transfer 100 bytes starting with 60000 of this source

and destination starting address is 70000 and I want to transfer the data from top to bottom. So, I want to decide whether this is from top to bottom or bottom to top.

So, total I want to transfer 64 (Refer Time: 48:27) KiloBytes. So, if I write simply MOV string bit. So, before this we have to specify something what is that is before this instruction you have to by default this 64 H has to be to be taken into only CX register only always string operations. So, CX in addition to this general purpose operations of CX; CX also will be used as a counter register in string operations. So, you have to I mean load this 64 on to the CX, 0064 H.

So, because CX is 16 bit register where as this is 8 bit data the remaining to I have added 00, means the maximum number of the bytes that can be transferred will be FFFF H. This is the maximum capacity of the bytes that can be transferred from here to here that is 64 kilo bytes of the data can be transferred from one set of the locations to another set of locations using the string instructions. So, by default the number of bytes to be transferred has to be taken into CX register, then we have to either set or reset the direction flag.

So, you have to write either CLD clear direction flag this is the instruction that I am going to discuss later, but this is a simple instruction this simply clear direction flag CL stands for clear direction flag, STD stands for set direction flag. So, if the direction flag is set then this will be from top to bottom if direction flag is reset this will be from bottom to top. So, this will be bottom to top because this is bottom to top 6000 is bottom value. So, top value is 630. So, you have to clear the direction flag. So, simply you have to write here before this CLD and to load this 0064 H this also I have not discuss this is immediate addressing mode.

So, we can use directly MOV CX ,0064 H, this will comes to the immediate addressing mode, you can take either 16 bit or 8 bit data directly you can transfer to the either 8 bit or 16 bit register. Here because this destination is 16 bit you have to take 16 bit immediate data if the destination is 8 bit you have to take only 8 bit data this is another MOV instruction that have missed there. So, this will be immediate addressing mode. So, you can write here simply this operation is nothing, but MOV CX ,0064.

So, these 3 instructions are enough then automatically the contents of this will be transferred to here, this will be transferred to here, this to this, so on up to 60063 H to

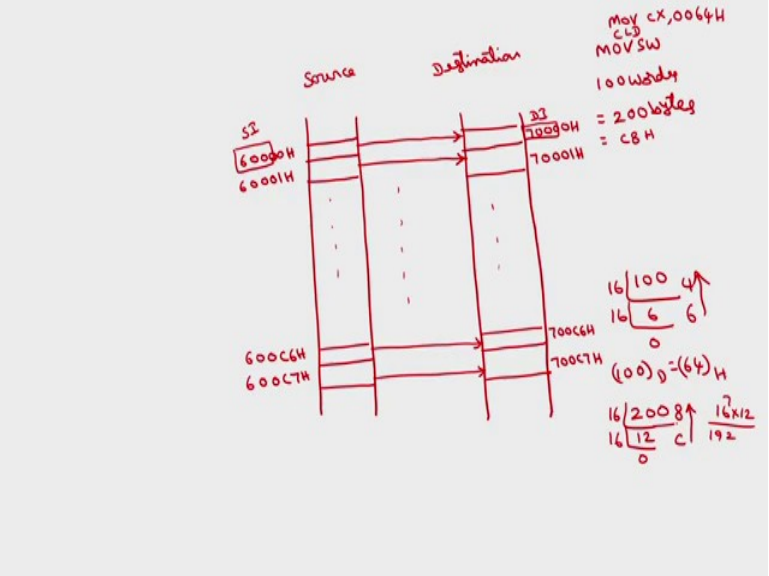
70063 H. So, by simply writing the 3 instructions a string of data will be transferred this is what is called MOV string byte, similarly you can transfer even MOV string word also. There is another instruction which is called as so, this belongs to the string addressing mode. So, this is one instruction another instruction is we can write MOV string word B stands for byte which is equal to 8 bits, W stands for word which is 16 bit or 2 bytes.

So, everything is same. Here also we have to set this CX to some value the number of bytes to be transferred number of words to be transferred now here in this case number of words to be transferred if I wants to transfer same 100 decimal words or 64 hexadecimal words. So, then we have to load into CX register then you have to clear the direction flag if you want to transfer from so, the source index and destination index.

So, here in this case this address will be pointed by this upper 16 bit starting of this will be pointed by source index and this upper 16 bit destination address will pointed by destination index registers. So, after each byte the contents of the source index and destination will be automatically incremented if the direction flag is cleared if direction flag is set they will be automatically decremented.

So, it is how this 100 will be transferred. Here also same clear direction flag, then we have to write MOV string word. So, instead of this MOV string byte if I write MOV string word, then what happens is this will take 2-2 bytes. So, these 2 bytes will be transferred to these 2 bytes. So, totally this will take 200 bytes will be transferred from source to destination.

(Refer Slide Time: 53:59)



So, this will be, this is source and destination, same example, but here the number of bytes will be transferred is 100 words, because I have written MOV string word and I have loaded CX with 0064 H and I have cleared the direction flag. So, 100 words has to be transferred.

So, 100 words is 100 decimal, 100 words is equal to 200 bytes and hexadecimal what is this 200 bytes 100 is equivalent to 64 how to convert this decimal number to hexadecimal number we have to divide with 16 , this is 64 H, 100 decimal is equal to 64 hexadecimal. Now, what is 200 decimal, 200 16 12s will be? 192.

So, 16 12s 192 reminder is 8, 16 0s reminder is 12 which is called as C. So, this is equal to C8 so, this is equal to C8 H. So, if I start here. So, what is the address that have taken here given before 60000 and 70000? So, 60000 H, 60001 H, then the last address will be C8 means 00 to C7, 600C7 because starting is 0 and this will be 600C6. So, totally 200 bytes are 100 words will be transferred.

So, this will be 70000 H, 70001 H. So, one up to this is 700C6 H, 700C7H. This is how you can transfer whether bytes or words, here also this source and destination will be pointed by this. This is how you can perform MOV string byte or MOV sting word. So, these are some I mean data transfer instructions. So, we have some other instructions like PUSH and POP, XLAT. So, the remaining data transfer instructions we will discuss in the next class. Thank you.