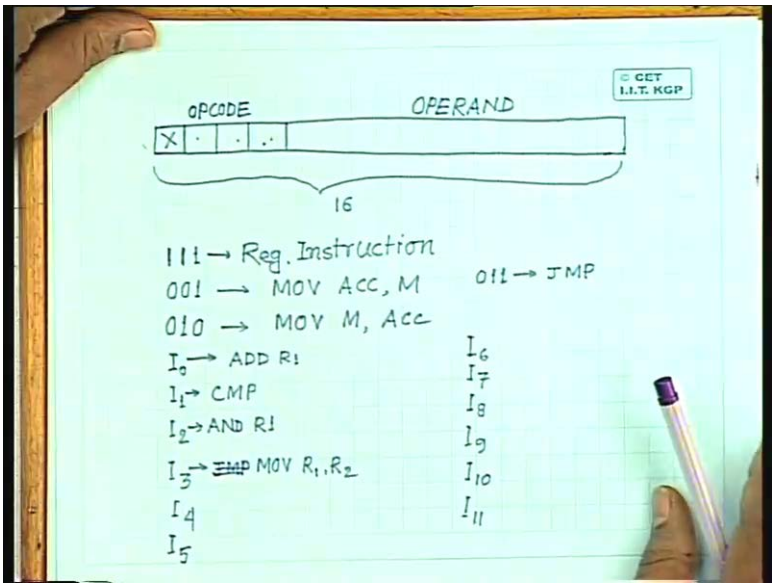
Digital Computer Organization
Prof. P.K. Biswas
Department of Electronic and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur
Lecture No. # 03
CPU Design-II

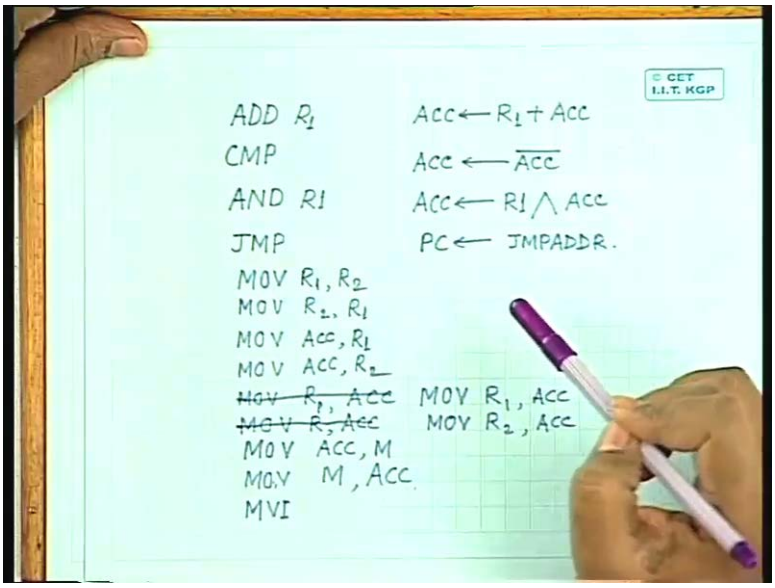
Now let us see that given this instruction, set of instructions and the instruction format, how we can design the hardware?

(Refer Slide Time: 00:01:09 min)



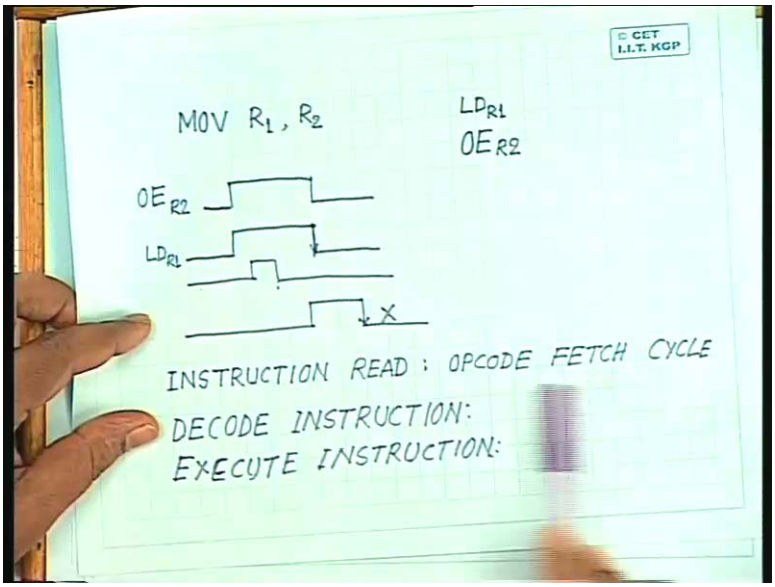
So here for designing the hardware, what you have to do is we have to study each of these instruction in detail that what these instructions are doing and in which sequence those operations will be done. However you will see that few of the operations for execution of any of these instructions are common.

(Refer Slide Time: 00:01:39 min)



For example whenever the CPU will execute any instruction, the first one as we have said is an opcode fetch cycle. That is the instruction has to be read from the main memory and it has to be put into the instruction register. After that it will be decoded, the signal will be given to the control and timing circuit.

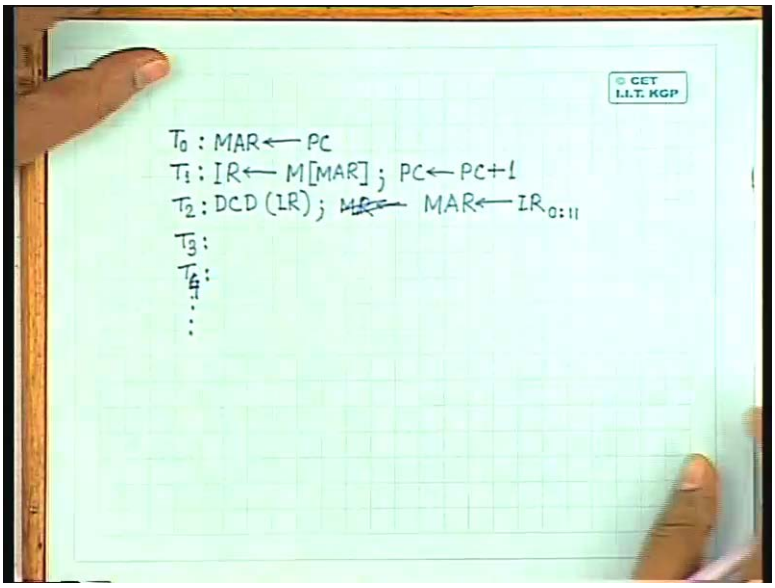
(Refer Slide Time: 00:01:54 min)



The control and timing circuit will generate the control signals, the required control signals in the required signals. So the first cycle that is opcode fetch cycle, it is common for each and every instruction. So let us see how this opcode fetch cycle will be performed.

For opcode fetch as we have said that this is similar to a memory read operation that means the memory address register has to be set with the address of the instruction that is going to be fetched and we know that the address of the instruction which will be executed resides in the program counter. So the first operation that has to be done is setting the data into the memory address register or address into the memory address register and this will come from the program counter. So the first operation will be, you have to load the memory address register with the content of the program counter.

(Refer Slide Time: 00:03:16 min)



So once the content of program counter goes to the memory address register then that particular address, that particular location in the memory has to be read and whatever you read from that address has to be put into the instruction register. So next operation that will be performed is instruction register gets the value from memory whose address comes from memory address register. At the same time what I can do is because once you read an instruction, the program counter has to be incremented so that it points to the next instruction in the memory. Here I assume that each and every instruction takes just one location in the memory. So the address of the next instruction which will be executed is the next location in the memory.

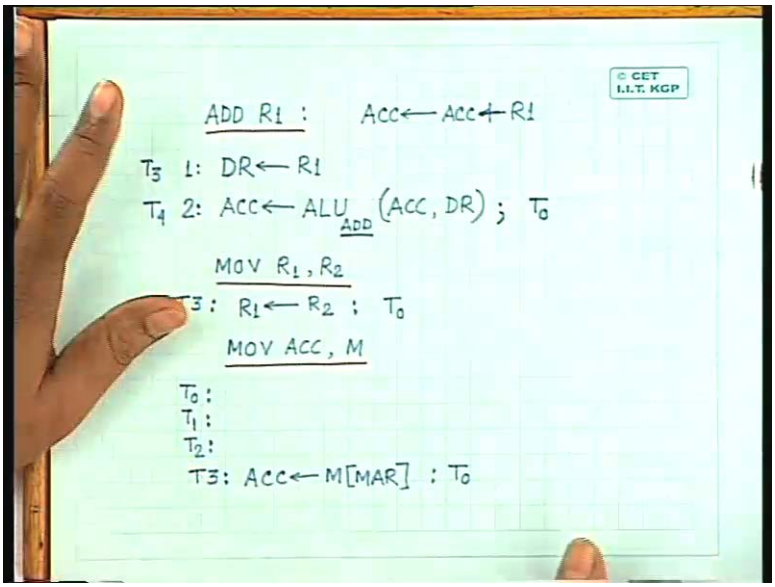
So I simply increment the program counter by one. This is a simplified situation where my assumption is every instruction occupies only one location in the memory but for a complicated situation where if I incorporate the provision that the instruction length can be varied. It need not necessarily be one location, it can be multiple locations. In that case I cannot simply increment the program counter here. Incrementation of the program counter has to be done after decoding the instruction because only when you decode the instruction you would know what is the length of the instruction and the program counter has to be incremented accordingly. But for our simple case because I am assuming that every instruction is loaded in only one location in memory so I can just increment the program counter at this point.

Now after that you have to decode the instruction. So decode the content of instruction register. So these are the operations which are to be performed always. Now simultaneously when I decode the instruction, I can do one more additional thing because till now I do not know what is the type of instruction that I am going to execute because decoding is not yet complete. It is possible that it is a memory reference instruction. So there is no harm that from the instruction register, the lower 12 bits which is supposed to be the operand address for the memory reference instructions. Simultaneously while decoding, I can send the content from lower 12 bits of the instruction register to memory address register. Even if this is not needed but this does not harm, so what I will do is I will shift the lower 12 bits of the instruction register that is IR_0 to IR_{11} to the memory address register during the same time when the instruction is decoded. These are the operations which are common for execution of any instruction and the operations are to be done in this sequence.

So what I can do is I can define some timing intervals that during a particular time, program counter content will go to be memory address register. During the next interval of time (Not Audible) (Refer Slide Time: 07:45) for instruction in that case my address of the memory that is to be read is already available in memory address register. So I don't have to spend any additional time to set the memory address register that is the advantage. Even if it is not needed, this does not harm. So I will define this timing intervals as T_0, T_1 and T_2. I am referring T your 8085 microprocessor. You know that these timing intervals are popularly known as machine states. I hope you are aware of the term. So these are different machine states T_0, T_1 and T_2. During T_0 the operation is specific, during T_1 the operation is specific, during T_2 the operation is also specific. T_3 onwards the operations will be dictated by the decoder output.

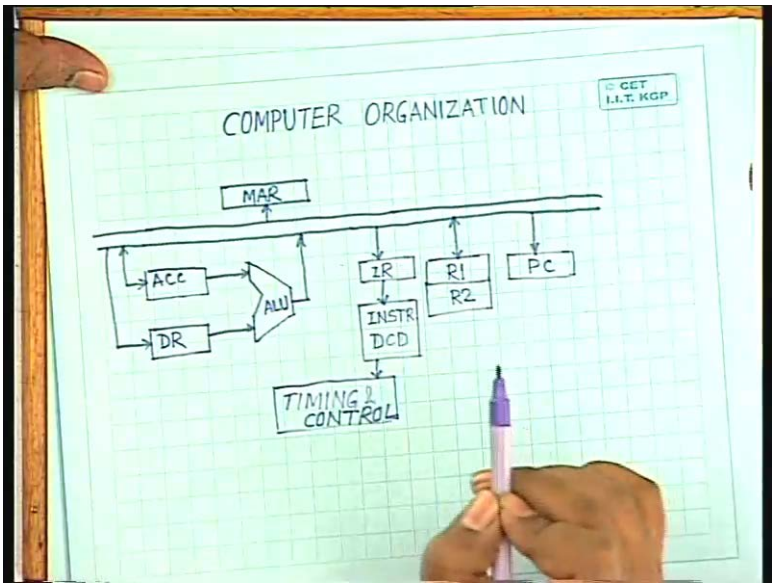
How many such states will be needed for execution of a particular instruction that depends upon what is the instruction that you are going to execute. So like this. So this to decide the operations during these states T_3 onwards, I have to analyze each and every instruction in details. So let us take few of the instructions. See what is to be done during T_3 T_4 like this, so this is T_4. So let us take few of the instructions. Let me consider the first instruction say ADD R_1. What we have to do for addition of R_1?

(Refer Slide Time: 00:10:08 min)



So the instruction that I am considering is the instruction ADD R_1 and we know that this is performing a function of adding the content of the accumulator with register R_1 and the result goes to accumulator where accumulator is the destination address. So this is the operation that will be performed by this instruction ADD R_1.

(Refer Slide Time: 00:10:54 min)



Now if you look at the architecture that we are considering. I have this ALU, the addition operation has to be performed by this ALU which needs two operands. For this instruction ADD R_1 one of the operands is in the accumulator, the other operand is in register R_1 but R_1 is not directly connected to accumulator. That means I cannot get the

data from R_1 to ADD with the content of accumulator and feed the results to accumulator which is not possible as far this architecture. So the ALU is getting data from accumulator and the data register DR. One of the operands is in the accumulator, the other operand we have to load into the data register. So that means from R_1 that data has to be transferred to date register before this addition operation can be performed. So that is the first step. So for execution of this ADD R_1 the first operation that we have to perform is load the data into data register from register R_1 that is a first operation. Then once you have the data into the data register, if I assume that output of accumulator is always connected to ALU, output of data register is always connected to ALU. That means I don't need any extra timing signal to set up this path. The accumulator to ALU, this path and data register to ALU this path, I don't need any extra timing signal to activate these two. I can assume that those are always connected but what is the important is whether ALU output can be activated or not. ALU output is to be activated only when you perform either some arithmetic operation or some logical operation where ALU is involved. So output of the ALU will go to the data path only when we need the output from the ALU.

In this case that is needed. So once I said the data into the data register from register R_1 the next operation that will be performed is accumulator will get the output of ALU and in this case the function of the ALU is ADD function. So I will put it this way, ALU_{ADD} then accumulator and data register. [Conversation between Student and Professor - Not audible ((00:13:58 min))] I will come to that. So these are the two operations which are to be performed while execution of this ADD R_1 instruction. These type of operations are called micro-operations. ADD R_1 is a complete operation, you break that operation into a number of micro-operations in sequence. So when all those micro-operations in the proper sequence are complete then only the execution of the instruction is complete. So what are the timing signals that will be needed?

For first operation that is loading the data from R_1 to DR, I need one time interval. [Conversation between Student and Professor - Not audible ((00:14:54 min))] Why do we need two symbols? [Conversation between Student and Professor - Not audible ((00:14:58 min))] but as we said that both of them can be done simultaneously. You refer to that figure here. Both of them can be done simultaneously because output is enabled simultaneously, I activate the load input of data register. So one time interval is sufficient. So for this I need one time interval. We will see that later.

So for this, it will take, let us assume that it will take one time interval. For this also it will take one time interval. So earlier we had three time intervals T_0, T_1 and T_2. Now during T_3, I can assume that this operation can be done. During T_4 this operation can be done. [Conversation between Student and Professor - Not audible ((00:16:00 min))] Yeah but ALU is a combinational circuit. ALU is a computational circuit, the output will be available after few gate delays. So if we assume that each of this time at interval is long enough to take care of that gate delay, this one time clock is sufficient. That is the final details as I said it may so happen that if my circuit is very high frequency circuit. I am operating the CPU at very high frequency where each of this time intervals will be very small.

So that a single time interval cannot take care of the gate delay. In that case we may have to extend it. Instead of only T_4, I can go for, I may have to go for T_4, T_5, T_6 and so on or also of them together to get an extended time interval. Load the data, output of ALU into accumulator at the end of that extended time interval. So those are all finer specification or finer details which has to be done at the time of implementation but logically this is okay. Based on my assumption that each of this time interval is long enough to accommodate the hectics. So during the time interval T_3, R_1 will be loaded into data register. During time interval T_4, output of ALU will go to the accumulator.

Now the question is ALU is a multi function chip, it performs various operations out of which I want only the ADD operation. So as you know that ALU has got some functional select inputs. By selecting those inputs properly I can define that which operation the ALU will perform. Now it is the timing and control circuit which once it gets the input from the instruction decoder, it knows that it is the addition operation that will be performed. So again the timing and control circuit can generate control signals which will be fed to the function select input of the ALU and those signals should be available during the time interval T_4.

Now once that is done, now the function of ALU become specific. Instead of a general purpose ALU with these select inputs available, function select inputs available, the ALU becomes simply an ADD circuit, a combinational circuit performing an addition operation. So output of that will be sum of accumulator and data register. I have to make a provision that output enable of ALU should be active so that this added result will be available onto the data path. Then I have to activate the load input of the accumulator so that from this data path, this result can be loaded back into the accumulator itself.

So for execution of this ADD instruction including the output phase, I need total 5 intervals T_0, T_1, T_2, T_3 and T_4. Out of which T_0, T_1 and T_2 these are common for all the instructions. T_3 and T_4, these two time intervals the micro-operations during these two time intervals depends upon the instruction. Now this is not sufficient, I have to care of one more thing. That is at the end of T_4, execution of this instruction is complete but I must set the machine in such a way that the machine is ready to fetch the next instruction and execute. As we have said that fetching the next instruction starts at time interval T_0. That means at the end of T_4 I must set the state to time interval T_0. So these are the operations to be performed for this ADD R_1 instruction.

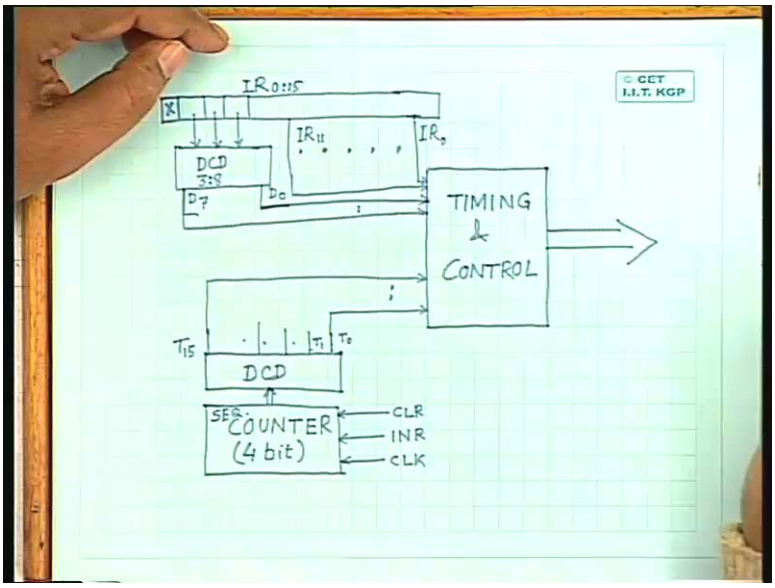
Let us take another instruction, so this was a register reference instruction. Similarly you find that this is the addition instruction which is register reference instruction. If I simply have a data transfer operation say move R_1, R_2. If this is the instruction which has to be executed as before, the first three time intervals during this the operations are specific, this has to be done. We have to specify that what will be done during T_3, what will be done during T_4. This being simply register transfer operation for transferring the data from one register to another register, just one time interval is sufficient. So I have to have time interval T_3 during which the micro-operation of transferring data from R_2 to R_1 will be done. At the end of this operation, I have to put the machine in times at T_0.

The hardware that I am going to design that should take care of all these things. Is that okay? Let us take a memory reference operation and in our instruction said, we have put only memory reference instruction which are data transfer operations like move accumulator, memory or move memory, accumulator. These are the only two memory reference instructions that we have put. Let us take any of them say move accumulator, memory. So these are the instructions that we are doing. So here again an operations during T_0, operations during T_1, operation during T_2, they are already defined. Here get you get the advantage.

You find that during time interval T_2 what we have done is along with decoding the instruction, we have also transferred the content of instruction register 0 to 11, this is 12 bits to memory address register. That means that the address of the memory which is to be read and transferred to accumulator that is already said. I don't need any extra time interval to perform this operation which otherwise would have been needed. So here what I can do is during time interval T_3, I can directly read the content of the memory and load that into accumulator. So I can straight away perform the operation of loading the content of memory with address in the memory address register to accumulator. [Conversation between Student and Professor - Not audible ((00:24:08 min))] Yeah exactly that is my assumption. As before after this operation is done what I have to do is, I have to set commission state to T_0 so that the machine is now ready for fetching the next instruction and executing it.

So now let us see that what are the hardware components that we need. However this can be expanded when you implement the entire system, what you have to do is you have to do this kind of analysis for each and every instruction so that you can design the complete instruction decoder, you can design the complete timing and control circuit. Let us see what will be the situation with these instructions only. So it is clear that I need two specific units. One for decoding the instruction, the other unit for generating the machine states or generating the time intervals T_0, T_1, T_2, T_3 and so on. Now what is this simplest way of generating the machine states? [Conversation between Student and Professor - Not audible ((00:25:33 min))] You simply use a counter, output of the counter you fit to a decoder. So for generating the machine states, the unit that we can use is something like this.

(Refer Slide Time: 00:25:49 min)



I use a counter. The counter output will be fed to a decoder. Then depending upon of the state of the counter, one of the decoder outputs will be active. So if I assume that this counter is a 4 bit counter, let us assume this. Now how many bits you need in the counter that depends upon what is the complexity of an instruction. So here when I assume that my counter is a 4 bit counter that means I can have 16 different machine states that is T_0 to T_{15} and none of the instructions, in this case can take more than 16 times states including the opcode fetch cycle.

If you have any instruction which takes more than 16 time states for execution, completion of the execution 4 bit counter will not be sufficient. I may have to go for 5 bits counter, 6 bit counters. So then how many bits you need in the counter that depends upon the complexity of the instructions that you have within the instruction set. That can be decided only after complete analysis of all the instructions in the instruction set. So when this counter output is fed to the decoder, the decoder outputs will generate different time states. So this is T_0, T_1 so like this I will have up to T_{15}. So with this my machine state generator is complete.

On the other side what I need to have is an instruction decoder. Instruction decoder gets input from the instruction register so I will put it this way. This is my instruction register IR which is having as we have said, it will have 16 bits $IR_{0\text{-}15}$. Out of this right now we are not making use of the most significant bit that is IR_{15}. I will extend this later. The next three bits I am saying that this contains the opcode of the instruction. So the simplest way is you have a decoder, here it will be a 3 to 8 decoder. So I have a decoder which is a 3 to 8 decoder because it takes 3 inputs from the instruction register and it generates 8 outputs D_0 to D_7. So now my instruction decoding circuit is done. The machine state generator circuit is done. What I have to do next? I have to combine these two to generate the control signals. That means the timing and control circuit, [Conversation between Student and Professor - Not audible ((00:29:58 min))] yeah for register reference

instructions, yeah that is true. [Conversation between Student and Professor - Not audible ((00:30:07 min))] input to the counter, yeah. Counter will have one input of clear, [Conversation between Student and Professor - Not audible ((00:30:20 min))] that is the set, counter will have one input called increment, counter will have another input called clock. So this clock is the master clock or assumption is when this increment input is one, with every clock pulse the counter will be incremented by one.

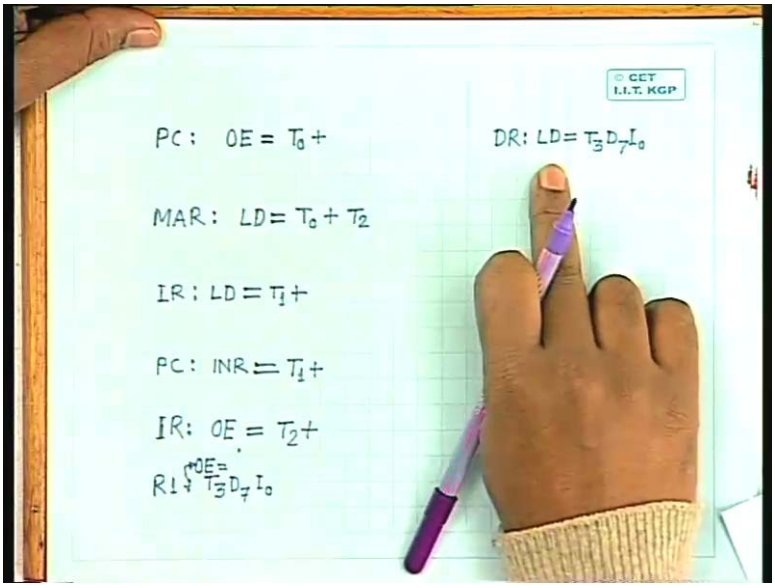
When the clear input is one, with this clock pulse the counter will be clear to say and that is what will enable us to set the machine state to zero after performing desired operation. This design means I have to set to that how these inputs will be generated so that requires some of that, when I need to increment the counter, when I need to clear the clear counter. So this now becomes the timing and control circuit. The timing and control circuit gets input from this instruction decoder. It gets from this, input from this machine state generator. It also makes use of these inputs IR_0 to IR_{11} for register reference instructions because these are the bits which uniquely identify a register reference instruction. For register reference instruction this D_7 will be active and this gives you all the timing and control outputs. So as a higher level schematic, I can say this thing will be timing and control unit along with this and this performs your instruction decoding unit. Timing and control gets this input so I can say that this is also a part of timing and control and this counter is normally known as a sequence counter. Is it okay?

Now the thing is what will be done initially? You all know that whenever you switch on the power of 8085 or you reset a 8085, the program counter of 8085 becomes 0 and you all know that the zero th location in the main memory must contain an instruction. If it does not contain an instruction then however sophisticated hardware you design, that microprocessor based system will never work. Why? Because whenever you are switching on the machine or whenever you are resetting the 8085, the first time state which is generated is time state T_0. That means initially the sequence counter will be set to state zero and during time state T_0, the operation is specific. This is hardware specific, this is not programmable. That is the program counter goes to the memory address register and whatever you get from the memory at that particular location that goes to instruction register, it its decoded and then finally other operations are decided. So in the zero th location in the main memory for an 8085 based system I put anything other than an instruction, the instruction decoder will not give me any value. It will give me some … (Refer Slide Time: 0:34:36.4) and timing and control circuit cannot recognize that, it cannot generate any proper timing control signal.

Effectively the system will fail. So that is the reason that it is always told that zero th location in 8085 based system should always contain an instruction. Typically what you put is a jump instruction and with that jump instruction you come to a bigger routine which is your main program and that location is different for different CPU's. It is not that for every CPU it has to be 0 0 0, that location is different. So with this, these two parts are complete. Now let us see how this block is to be realized. So for realization of this block, let us again come to yeah. So we know that during time interval T_0, your program counter content has to be loaded into memory address register, that is known.

So what are the control signals that are needed for this? I have to activate the output enable of program counter. I have to activate the load input of memory address register. So during time interval T_0, program counter output has to be enabled. Memory address register load input has to be enabled. So let me list out the control signals for the instructions that we are considering. One is program counter, one is memory address register.

(Refer Slide Time: 00:36:35 min)



So for program counter I have output enable control signal, for memory address register
So far what we have encountered is load control input. We have said that during time interval T_0, output enable of program counter must be active. During time interval T_0, load input of memory address register must be active. They may be active in other situations as well but during T_0 they must be active. So what I will put is I will set output enable of program counter is equal to T_0 OR, so I put an OR condition because when T_0 is true, output enable of program counter has to be true because it is an OR logic.

Similarly for load input of memory address register, I will also put as T_0 OR. That is whenever T_0, machine is in T_0 load input of memory address register has to be active others will come later on. So that is what is done during T_0. The next time interval is T_1. During T_1 what are the registers that are involved? one is instruction register otherwise the program counter. So I have the next register which is instruction register, I have program counter. The control signals that are encountered till now for instruction register, it is the load input. For program counter it is the increment input INR. So if you remember this block diagram, you find that for the sequence, sorry this is the program counter not the sequence counter.

So for program counter I have an increment input and the condition is during time interval T_1, the load input of the instruction register must be active. So I said T_1 OR whatever be the other conditions, during T_1 the machine state load input must be active.

Similarly for increment of the program counter during time state T_1, the increment input must be active. Here following the same logic as in case of sequence counter, I assume that clock input is also going to the program counter. So whenever increment control input of the program counter is active with every clock pulse, the program counter content will be incremented by one.

Now what is done during time interval T_2? During time interval T_2, you find that the instruction is to be decoded. However this decoder is simply a logic circuit. So I need not specify any control input for this logic circuit because it is the combinational logic circuit. When it has input after few gate delays, it will give me the output. The thing that I have to ensure is before time interval T_3 starts, the decoder output must be available. However during time interval T_2 again I am loading the content of instruction register 0 to 11, the lower 12 bits of the instruction register into memory address register. So for this the output enable of instruction register, one output of the instruction register is always connected to instruction decoder. So for that part I don't need any activating signal but the other path from the instruction register which is going over to the common data path, there I must have some enabling operation, some enabling circuit.

I will say that the output enable of the instruction register, here I have an instruction register, the output enable of this connects instruction register to common data path, output of the instruction register to common data path. for inputs we don't have any problem, input can go simultaneously to all the destinations but the one for which the load input will be active that will only load that input, others will not loaded but for output we must have specific selection otherwise there will be data clash. So for instruction register, the output enable control will decide the output of instruction register going to the common data path and this must be active during time interval T_2. So I will put it as T_2 OR, so whenever T_2 is true output enable of instruction register is true. During time interval T_2, the content of instruction register goes to memory address register. That means the load input of memory address register also must be active during time interval T_2.

So when you go to load input of memory address register, here this should be T_0 OR T_2. So with this all the common operations are complete. That is the operation that you have to do during T_0, operation we have to do during T_1, the operation we have to do during T_2. Now T_3 onwards, the decoder output will come into picture. So for that let us consider this ADD R_1 instruction. We have said that the decoder output for this ADD R_1… for ADD R_1 we have said that the D_7 output of the decoder will be high because we have said that 1 1 1 in the opcode tells you that it is a register instruction.

So when the opcode field is 1 1 1, in that case in this decoder that D_7 output will be active. So what will be my logic? During time interval T_3, if I find that D_7 is high, instruction decoder output D_7 is high and at the same time it is ADD operation if I_0, the zero th bit in the operand field of the instruction register is high then it is ADD R_1. So my logic is if it is time interval T_3 and D_7 is high and I_0 is high. Then what I have do to? What are the micro-operations? I have to transfer the data from R_1 to data register. So two more registers come into picture, one is register R_1 and other one is the register, data

register that is DR. So during time interval T_3, if D_7 is high and I or R zero or I_0 is high then output enable of R_1, so this is output enable of R_1 must be high.

Similarly this output of R_1 goes to the data register. So I have to activate load input of the data register. So load input of the data register must be active following the same condition that if the machine is in state T_3 and instruction decoder output D_7 is high and the instruction register bit I_0 is high. This is one of the conditions. Whenever this is true, output enable of register R_1 must be active. Whenever this is true, the load input of register, DR data register must be active. There may be other conditions as well so I will put as OR condition. So this OR some other condition this will be active. This OR some other condition this will be active. So with this I will complete my discussion today. We will continue from this point onwards in the next class.
Thank you.