

Computer Organization
Part – I
Prof. S. Raman
Department of Computer Science & Engineering
Indian Institute of Technology
Lecture – 13
Problem Exercise

In the previous lecture, we considered one example that was a problem related to n-operand instructions, where n was 0, 1 or 2; here it is in the two operand instruction format. Then we considered 4 bits for opcode, 4 bits for source data – that is one of the two data – and 4 bits for the destination data. Then while discussing these 4 bits, we said 2^4 or 16 combinations are possible; that is, 16 opcodes are possible. With this opcode field, it must be identified whether we are referring to no operand or single operand or double operand instructions. So we said with 4, only 16 are possible. Actually these are the 16 possible codes. Now this is one way of coding. Of course, it is very economical in the sense that all the 4 bits in the field are used; so starting from all 0s to all 1s it goes.

There is another way of creating codes with n bites; let us say it is 4 bits itself. For instance, how about this? Suppose we have one code; then let us say this is another code; this is a third code; then we have this code and then we have this code. Now with 4 bits, I can also create these five codes; of course it is nothing but a subset of this. Now the point you have to notice here is that if there is a 1 bit, then there is only 1 bit in the entire field. Both these are possible; now I say a code must contain four possible combinations. This is also one form of coding, but then when you do it this way, being a subset, the entire 4 bits position as in this particular case are not used. So you can say that when you make use of all the 4 bits and then fill in 1s and 0s appropriately and exhaust all the possible combinations, then we get the entire thing: 2^4 ; otherwise, with less than 2^4 also, you can form a code. Actually both these are in use.

Now in a scheme such as this when you have all the 4 bits, this could need a decoder, because just by looking at the code and by looking at the position of 1s and 0s, you cannot make out what it is. You have to study all the 4 bits in the combination; so a decoder is needed in this particular case, whereas in this particular case, it is not necessary. You can just see whether there is a 1; if there is a 1, looking at where exactly the 1 is, you can make out. You do not need to study all the four aspects. So both these combinations are possible; for instance, a scheme such as this could be called a linear select scheme. It is called linear mainly because you have only one at a time; one of the different positions in a linear manner. So we can form different types of codes; both of these are actually in use. What is the advantage of having this? The disadvantage was seen: that you need a decoder later on to understand which code it is. Now what is the advantage of this? The advantage of having this is that you can include as many codes as this particular scheme permits. That is, 16 codes are possible. Anything is going to be less, any other type of coding. This is one point we have to note mainly because we have to consider the second problem, which will come to this.

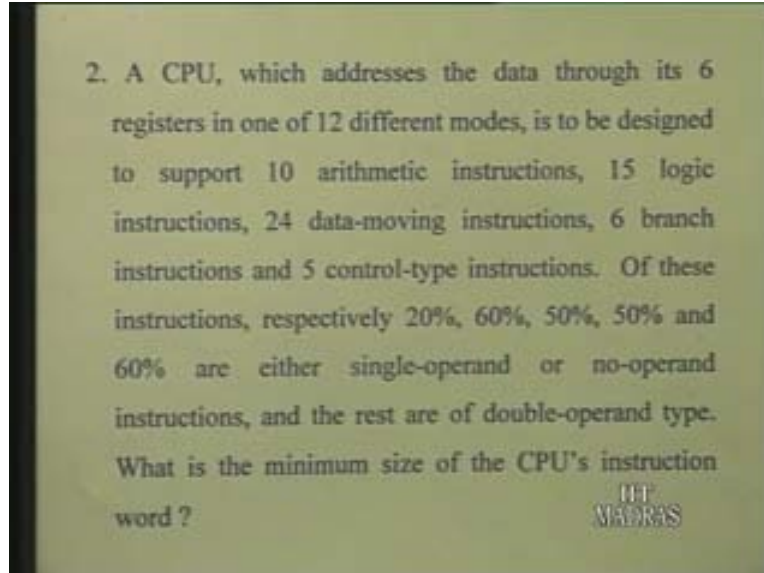
Now let us take a look at the chart and then see what the second problem is about. The second problem says a CPU addresses the data through its six registers in one of twelve different modes – you remember I was talking about the instruction format in which we not only give the opcode but also mode with reference to the register. So we will have to formulate that. Then this particular CPU is to be designed to support different types of instructions. The numbers are given – you have 10 arithmetic instructions, 15 logic instructions, 24 data-moving instructions, 6 branch instructions, and 5 control-type instructions. In other words, the study of any instructions we said will consist of studying each class of instructions.

Now add and subtract for instance will form arithmetic instructions, then or, and, NAND nor, or – these things will form the logic instruction; then data-moving instructions move from one register to another register; any transfer from memory locations to register and so on so forth will form the data moving instructions. The six branch instructions, normally the programs is executed instruction by instruction, but then suppose you stop and when you do not consider the next instruction but then you take an instruction away, then you have to branch away and stop the normal sequencing operations. So branch instructions are needed for that. Then, there are 5 control types of instructions; there are different types of instructions here – for instance, if in a particular board a particular bit or a set of bits must be set; they will be instructions, so this all these types of instructions will form an instruction set. These instructions, respectively it says, for arithmetic instructions 20% and so on and so forth.

All these are single operand or no operand instructions, that is, the single operand and no operand instructions have been grouped here and the remaining, the rest of them, are double operand type. So again we are back to the n operand case, there is a no operand, 0 operand and 1 operand and then 2 operand case. Now it says, what is the minimum size of the CPU instructions word? We will note this point, minimum size, and then I will talk about it again later.

Let us write down and then note down the various points. What did it say? May be you can make use of this op-code. We have to find out the sizes in this particular case, and we have to work out a few more things. This address has data through its six registers, so the source data will be referred to through two different fields: one half of the field will be referring to the register, and another half of the field will be referring to the mode. So mode of the source data is one thing – and similarly this also. This is of course for double operand instruction. So what it says is that the CPU addresses the data through its six registers, which is the number of registers you have; in this particular one, it is 6, then 12 different modes; the number of modes is 12. Now what is the question? Let us take a look at the problem again – it says, what is the minimum size of the CPU's instruction word?

(Refer Slide Time 06:46 min)



Now working out the minimum size, the way you can do it is only by saying to accommodate reference to any of the six registers, for instance here or for here. If you want the particular instructions word size to be minimum, the way to go about is only by decoding. In other words we have to code, given that this will be a binary word, we will be using the binary thing, which means for referring to any one of six registers you need six unique codes like this, and obviously you need at least 3 bits, because 2^3 is 8, and so you can accommodate certainly the 6 that you need.

This implies that you need a 3-bit code. So now you have this register part; this particular register part of the field will be 3 bits in size. The same argument holds true with reference to modes. You have to accommodate as many as 12 different modes. This implies that the minimum you need is 4 bits, because 2^4 is 16; in the other way, 6 can be accommodated. Similarly here, 2^4 are 16 and so 12 can be certainly accommodated; so you need 4 bits here.

Now the way we arrive at that is mainly because we have to look at the chart; that is a minimum size. For instance, you need not go in for this now; referring to six registers can also have the register subfield; it can also have 6 bits. Similarly for the modes you can also have 12 bits, but then it becomes unwieldy; you are not fully utilizing all the things. Now the question is how to determine the opcode size? That is the main problem, for which you have to work out the details. What are the details? Let us go to the chart and look at the problem again; well, we have these various class arithmetic instructions, logic and so on and so forth. Each of these and how many percent of these are single operand and no operand is given. So let us prepare a table and then work out the details of that.

(Refer Slide Time 12:50 min)



So let us first take the instruction type and the numbers of them; it has been given single operand or no operand; so single operand or no operand is clubbed together; then we have the double operand instruction; so we have to work out. Now first, we take the arithmetic type of instructions; actually, it does not matter what it is, arithmetic, logic, data-moving, branch control. In arithmetic type, it is given that the CPU must support 10 instructions; and then in logic type the CPU must support 15 instructions; the next is data moving instructions. In data moving instructions we have 24; then in branch instructions we have 6. Then we also take a look at the chart: we have 6 branch moving instructions; then 5 control type instructions; 10 of arithmetic instructions; 15 logic instructions; 24 data moving instructions; 6 branch instructions; and in control type, there are 5. Now it is further given that 20% of this will be single operand.

First let us write down 60% of this; when it says 50% of this or 60% of this, these are single or no operand. So what is 20% percent of this? It will be 2; 20% is one-fifth, so it is 2; then 60% of this will be 9; 50% is half, that is, 12; again this also 50%, that is, half of 8 is 4; 60% of 5 will be 3. So how do you determine the double operand? Now why should we work out these double operand instructions? Because we have to see a common format, is it not? A common format is needed and the common format will be for double operand case; that is, the highest number operands. So if you work out for double operand for which we already have some partial information, that is source and destination, this will give us some information. Now let us see about opcode. Except for the opcode part, we already got rest of the information for the double operand instructions.

(Refer Slide Time 18:22 min)

	TYPE		So/No	D.O
A	10	20%	2	8
L	15	60%	9	6
D	24	50%	12	12
B	6	50%	3	3
C	5	60%	3	2
				<hr/>
				31
				<hr/>

HT
MADRAS

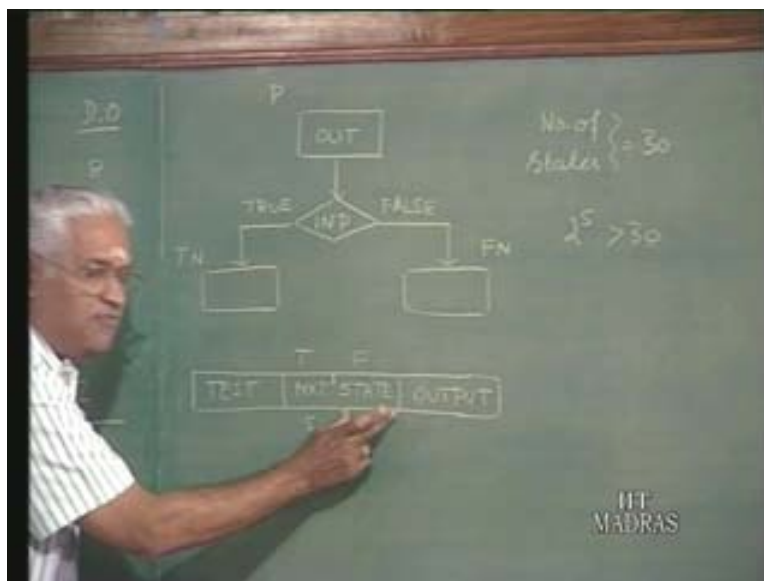
So if we knew how many double operand instructions must be uniquely identified, then we have solved the problem because this is what we have been looking for. Once you have the common format for double operand, later on it can be verified and then seen whether single operand and no operand cases can be accommodated with the same format. Now it is given as single no operand case and double operand; so these are the only three combinations. So 10 minus 2 or 8 of these will be double operand instructions. Similarly, 15 minus 9 or 6 of these will be double operand, 24 minus 12, that is, 12; 6 minus 3, so 3 of these, 5 minus 3 or 2 of these. So the total number of double operand instructions will be 2 plus 3, which is 5, plus 12, which is 17, plus 6 or 23, plus 8 or 31. So there are 31 instructions, double operand instructions, which must be uniquely identified. Now that means we need 31 codes for double operand instructions alone.

What does it mean? It means again going for the minimum, what is that? Let us take a look at the chart again, remind ourselves that the minimum size has been asked of us. So in the same way, the opcode also will have to be in this kind of coding. The minimum would be 2^5 because 2^5 are 32, which accommodates the 31 that we need. That means we need at least a 5-bit opcode field. Now let us make sure that the rest of them can be accommodated minus 31 go for double operand instructions. So one code is left; that one code will identify single or no operand cases. You can just total them up and then check whether it can be accommodated because of single operand is one in which we need at least one data field, operand field, for referring to the single operand, which means this particular field can be used as extension of the opcode and this one has already 7 bits. So with the remaining one code with 7 bits, 2^7 will be possible. So it has a maximum of 2^7 and remembers there is only one code left; so of the 2^7 minus 1 will be used for maximum, for single operand, because we need to have at least one for referring to the no operand of that. Because there is only one code that is left, that one code only says whether it is single operand or no operand.

Then we have 2^7 ; you can just work out and make sure that this is 20, but remember that with 2^7 codes, the entire thing cannot be used for single operand. The minimum is $2^7 - 1$ is single operand and the remaining thing in that particular one will indicate no operand, because that is only one minimum. Then no operand is referred to, we will have another 2^7 . Anyway, actually the problem itself does not say it; we do not have information about how many single operands and how many no operands are there. As you can see, when you add this up, it is going to work out for less than 2^7 and that is for the combinations of single and no operand. So this is how you work it out. Now what is it? It says 5 plus 4 is equal to 9; plus 3 is equal to 12; plus 4 is equal to 16; plus 3 is equal to 19 bits – this is the size of the instruction word.

Take a look at the chart – so minimum is 19 bits; that particular thing is the instruction word. Now let us just quickly review the two things – the first problem we saw how, given the size of the instruction word, how many instructions can be accommodated, how many different types of instructions. In this second problem, it is the other way – given the number of instructions in the type of instructions, what will be the size of it? So, only variations of these will be there generally. Now let us go on with the next problem, that is, problem number three – the next chart please. This calls for information; let us see the problem. You have to recall whatever you have done early in the lecture; what is it? Just read the problem; the format of the micro-word of a 30-state micro-program controller. So it is back to the controller design and micro-programmed controller; so it is micro-programmed implementation.

(Refer Slide Time 29:32 min)

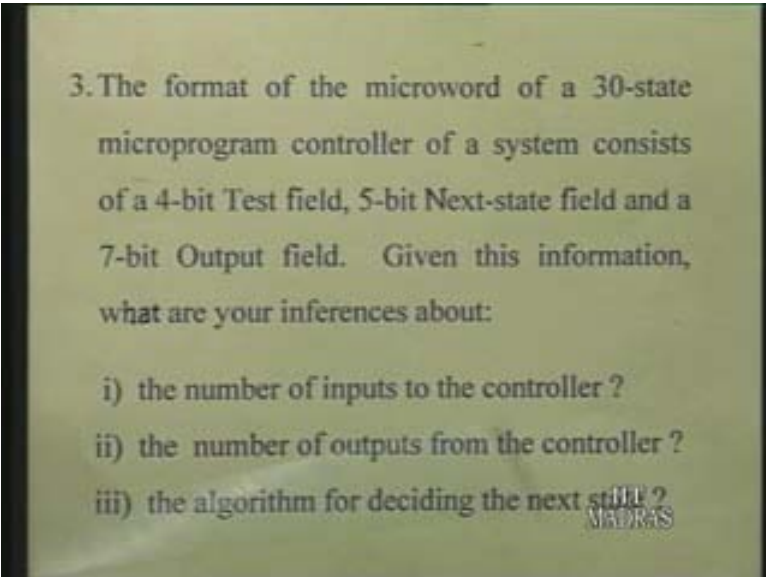


Now this particular format consists of a 4-bit test field; you remember the format we generally have test field, next test field and output field. That is what you have in any micro-word for a micro-program control. So the size of it is given as 4 bits, 5 bits and 7 bits, respectively. Now given this information, what are your inferences about the number of inputs to the controller; the number of outputs from the controller; and the algorithm for deciding the next state? Let us quickly recall some of what we had seen earlier.

Let us just take an ASM chart. While discussing the ASM chart, what did we say? That from any given state, that is, the present state, some output may be generated and then, in that particular state – I will just call state P – it checks for an input, and depending on the condition of the particular input being true or false, two other states will be there. The system goes into either of these two next states. I will call it true next state and this the false next state. Now in any state, more than one input can also be checked correspondingly, and if you have more than one input we can introduce some extra dummy states and finally bring it to this. For the one present state, there are only two possible next states. It is a very general thing that can be done that way. Let us not worry about that. Now what is given here? We are talking about the format of a word; that is what this problem states. The format of any micro-word essentially is, you have this test field and then the next state field; next address or next state or whatever it is; and then you have the output field. Now let us read the problem again. In this particular one the controller is for controlling a machine, which has a maximum of 30 states. So the number of states will be 30, which means, as per this particular diagram we do not know whether for a given particular state instantly what micro-word for a particular state is.

For instance, this may possibly be a micro-word for state P. And if it is for the state P, then it says the inputs to be tested are nothing but this. The test field will have information about the inputs to be tested, and then, depending on the particular input being true or false, the two next states – true or false next states – will have to be identified. Really speaking there could be two parts – one thing, which gives the true next address and the other thing, which gives the false next address. So there will be two subfields of this, and then the output that must be generated in the particular present state P. According to this, we have the number of states as 30; then how many bits do we need? Assuming as in the previous one, we go in the economic coding; it is economical from the point of view of the number bits usage. So you need at least 5, because 2^5 is 32 and that will accommodate 30, which means the next one must be 5 and this part also must be 5. By this for a given state we have to identify both the false and the true next states.

(Refer Slide Time 23:57 min)



3. The format of the microword of a 30-state microprogram controller of a system consists of a 4-bit Test field, 5-bit Next-state field and a 7-bit Output field. Given this information, what are your inferences about:

- the number of inputs to the controller ?
- the number of outputs from the controller ?
- the algorithm for deciding the next state ?

MADRAS

So you need a 5-bit subfield for, say, false next state, and another 5 bits for true next state. Now what has been given in the problem? Take a look. It says the test field is 4 bits; its next field is only 5 bits and then you have a 7-bit output field. So now we said we need at least two 5s but what has been given in next field is only 5 bits; test field is 4 bits; and then the output field is 7 bits. We have no choice; we do not have the 10 that we require; we have only 5. Let us see the chart – which one indicates the number of inputs to the controller? The test field indicates that, because in any micro-word you have to give information of for every state; the inputs to be tested must be given in the test field.

Now it is given that the micro-word is 4 bits. Why? What do you infer from that? Four bits means the number of tests; so basically that is what we have to do. The number of tests and number of inputs is the same; so test will indicate how many numbers of inputs can be there. When you have 4, what will be your inference? Go back to your coding. We have both linear select and we have decoded type of arrangement. So suppose we use this code the number of inputs will be only 4; if you use this code it will be as many as 16. So we cannot say what it is; all we can say is that the number of inputs can be anything from 4 to 2^4 ; that is the only inference we can draw. Let us finish off the other one – what is the next one? The number of outputs from the controller is similar to this number of outputs – you would be tempted to say the same thing.

What is the deferral between this and that? Here also, 7 to 2^7 . We will accept the answer 7 to 2^7 . It can be even less than 7; does not matter we use maximum 7. Now suppose we say the number of outputs can be as many as 2^7 , what is the implication of that? Think a little. Suppose the number of inputs should be 2^4 . Let us go back to the code when we use this; this is when you have 2^4 . What happens is at any time only one code is possible. Similarly if you have the output 2^4 or 7 outputs, at any time only one output can be generated – is it a practical thing to do? Recall when we came across this micro-program controller and controller in general in generating the data path control signals for the data path and in the data path in different parts. We set the data path by enabling and disabling the different parts in the architecture, which means simultaneously different signals have to be generated. So this is really meaningless because if you have 2^7 , then only one output can be generated at a time.

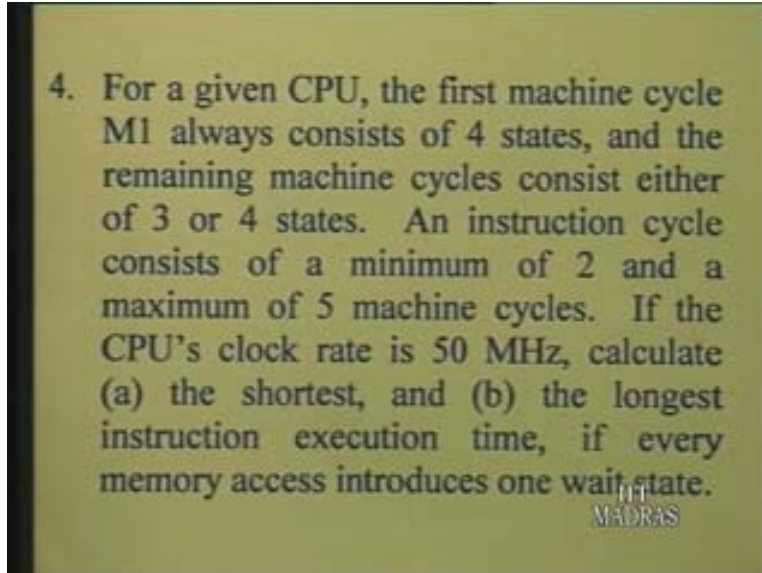
So it is not really good, but anyway, maybe there is something which is possible. But in the practical cases, which you usually come across, it may not be very correct answer. Well, theoretically 7 is 2^7 , but practically speaking, it will not be 2^7 . Instead, what can happen is that in the 7-bit output, some part can be coded and some part maybe separate. For instance, you may have, say, 7; now this one has 2-bit code; this will be a 3-bit code; these two will be separate, which means independently I will say this or this or this 3 bit, denoted by C, or the 2 bit, denoted by D, can be generated. This means actually simultaneously at least four signals can be generated; out of it these 2 bits together form one thing. So this is neither linear select nor decoder, but it is in-between these two. Now it is the decode scheme within this C and D, and it is linear select or combinations of that also – A B C D. So you can generate them. Now let us come back to this third part of the question: it says what is the algorithm for deciding the next state? I almost hinted what will be the algorithm. Now the algorithm for the next state must indicate to you when the test fails, that is, the input is false. Then you have to indicate one 5 bit and with 2, a 5-bit code for the next state.

If it is true to another 5-bit code, it is the true next state. But it so happens that the problem state is the only 5 bit that you have. So what is the way? We cannot indicate both; so, for a given state P, let us say if it is P, it says test, any one of these inputs; and then in the state B, generate any one of the outputs as indicated. And it says you can give only one 5-bit next state. Which one? That means basically either you can give this true or this false or you can give this false but not the true; only one of these codes is possible. So obviously that is the only way you can go about it. What is the problem actually? It says what is the inference about the algorithm for deciding the next step? The algorithm must be somewhat like this. We can write the algorithm. If the test is true, then next state will be, let us say, the present state plus one, which you do not have to indicate anywhere. Otherwise, the next state is as given by the 5-bit field. That is, whatever is given in the 5-bit next field? So if this is the algorithm then it is enough if we give false in the particular one.

The false next state is being given; the true is not being given. Remember it can be the other way – you can also put if the test is false – both are possible. So if the test is true the next state for the controller will be the present state plus one, which can be designed. Is it anything new? No. What you do in program counter is always incremental by one; use the program counter to fetch an instruction and then the next instruction is got from the next location, for which the program counter content is incremented automatically. Only when it is not so, whenever a branch or jump condition comes, some other address will have to override.

So the same logic can be applied here – if the test is true this has to be designed; in the next will it automatically be the present state P plus one. So as I have formed here, this code for T_n will be T plus one, that is, the next one, whereas this false next state will be indicated to the 5-bit field. I hope you have noted enough points to see how exactly the format of a micro-word is arrived at, and also important points such as assuming the next field as the present one plus one. That is an assumed next field; this one will be the given next field, which is given in the format itself. So this you may say is assumed next field: assumed to be the present one plus one, assumed next state is address, whereas this particular one is given next state in the word itself, micro-word. And another important thing is that when you can go for linear select and when you cannot really go for it – it will be meaningless to generate only one output.

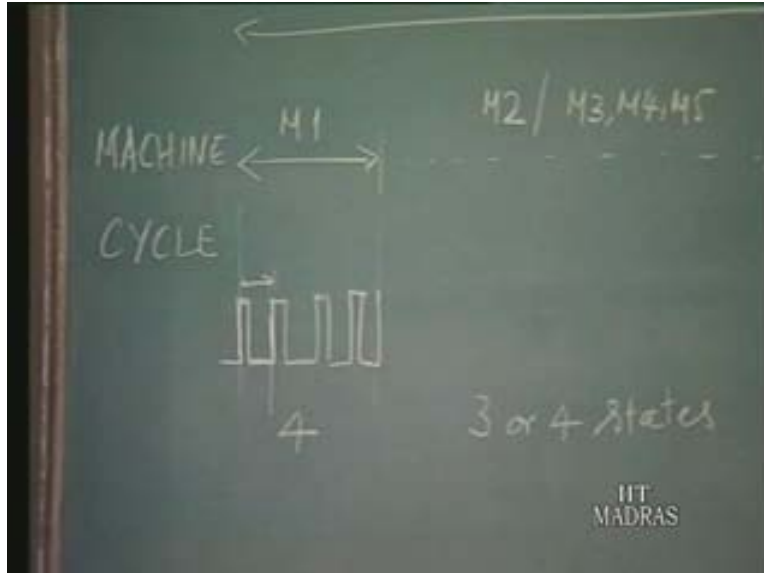
(Refer Slide Time 41:48 min)



Let us take another problem – this is to do with the instruction cycle. Let us see what it is – the chart says for a given CPU, the first machine cycle M1 always consists of four states. You have to recall here that an instruction cycle consists of machine cycle, which consists of states; so here we have the details of that. The first machine cycle consists of four states – that is always so; and the remaining machine cycles consist either of three or four states. So it may be three or four states; it depends on the instruction. The instruction cycle consists of a minimum of two and maximum of five machine cycles. So this particular thing, a variable minimum of two, would mean M1 and M2, let us say, and maximum of five. So M1 is always there, then M2 and then beyond that M3, M4 and M5 are there.

Now it says the CPU has a clock rate of 50 MHz – we talk about clock rate here because this clock rate is the one which is going to define the state duration. Then calculate the shortest and longest instruction execution times. How do you calculate? We have to look for an instruction, which has minimum machine cycles and minimum states specifically and the one with maximum machine cycles and maximum number of states. Now there is additional information here – what is that? If every memory access introduces one wait state, what is the memory access? How do you identify that? What is the machine cycle? A machine cycle is always entered into whenever there is a bus activity. So a memory access would mean actually accessing for some information over the bus. Now let us work out some details for you to note down from the data given in the problem. First of all, an instruction cycle will consist of machine cycles. So this is a machine cycle; given that M1 is always there, you may have the rest of the bits for instruction. You have only M2 minimum; we said it says that instruction cycle consists of minimum of two machine cycles, M2, then optionally you also have M3, M4 and M5. Another thing that is given is that about the first machine cycle there is some information. It says that there are four states. In the first machine cycle there are four states; essentially it means four clock periods. Do not worry about the non-uniformity of it.

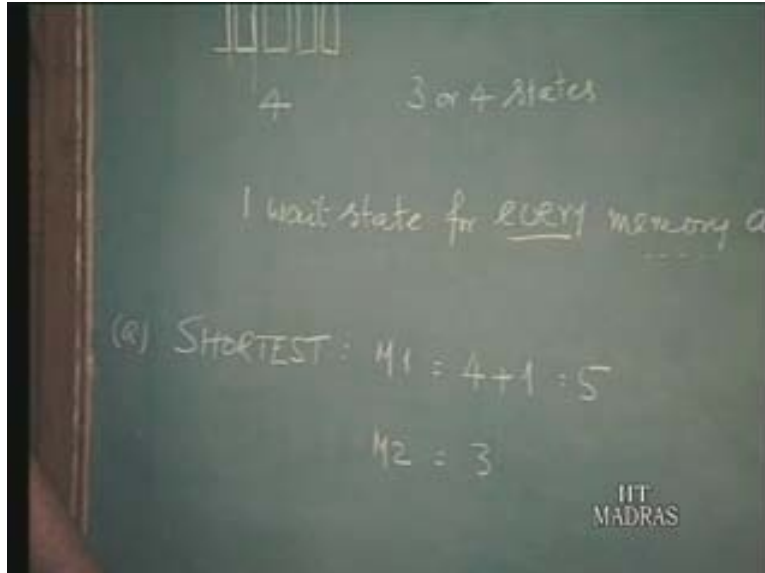
(Refer Slide Time 47:25 min)



There are four states – now this is fixed, because it is given that M1 always consists of four states, and the remaining machine cycles either consist of three or four states. Each one of them, each of the remaining machine cycles, that is, M2 to M5, will consist either of three or four and an instruction cycle consists of minimum of M1 and M2 and the maximum of M1 up to M5. Then of course, the CPU clock rate is given; we will make use of it later. That is, the clock frequency that is given is F clock or 50 MHz, from which we know the state duration. So from this point to this point, if you put it as a duration t_{cl} , one over 50 MHz will work out to 20 nanoseconds. We will check that later. There is one more point we saw. The last line in the chart says every memory access introduces one wait state.

What is M1? Let us see: a machine, say, M1, is fetching the instruction; there is some memory access. This is for fetching the instruction; there must be a memory access. Similarly, M2, M3, M4 and M5 may be memory access. Now what it says is with every memory access which corresponds to a machine cycle there is one wait state, which means there is an extra state that is introduced mainly because the memory is slow compared to the CPU. So now let us work out. Let us note that there is one wait state for memory access for every memory access, which means essentially we have to consider this information whenever there is a machine cycle associated, which is usually associated with the memory access.

(Refer Slide Time 50:01 min)



Now what is the shortest execution time? First we calculate number of states and then, later on, multiply that with the state duration so that will give us the execution time. So for the shortest instruction, how will you calculate? The shortest would obviously mean only two machine cycles, M1 and M2. During M1 we have four states; then obviously it is for fetching the instruction, which means a memory access. So there must be one extra state. During M1, you should have five states. Now we are talking about the shortest, that is, the first part of the question. Then we have M2. M2 is another machine and it says it has either three or four states.

So the minimum will be three, and there can be a memory access; we do not know. There is nothing wrong in assuming there may not be memory access because you follow the instruction here and then for executing the instruction, you take three; they may not be memory access. So we can as well have it as three states only. So this is for the shortest path, that is, we have eight states in the minimum. Then the longest one, the longest instructions execution will be M1 as before – there will be five states. Then we have M2 to M5, that is, four states because in the longest we have to include all the four states. So there are four states and again since it is the longest, we will consider that each one of those four machine cycles has four states and because it is longest again, we will assume that everything has memory access; there is nothing wrong in assuming. So there are five states, that is, M2 to M5 – we have four machine cycles into the number of states.

The longest is four and each has one extra state. So we have 4 plus 1 is equal to 5; 5 into 4 is equal to 20. So we have 25 states. For the first question, the shortest instruction execution time will be 8 into 20 nanoseconds, which is 160 nanoseconds, and the longest one it is 25 into 20 nanoseconds, which is 500 nanoseconds. Because there is a state duration, we are calculating the number of states for a given instruction. You find that 8 is the minimum and 25 is the maximum, and every state needs 20 nanoseconds. So this is how we arrive at it.