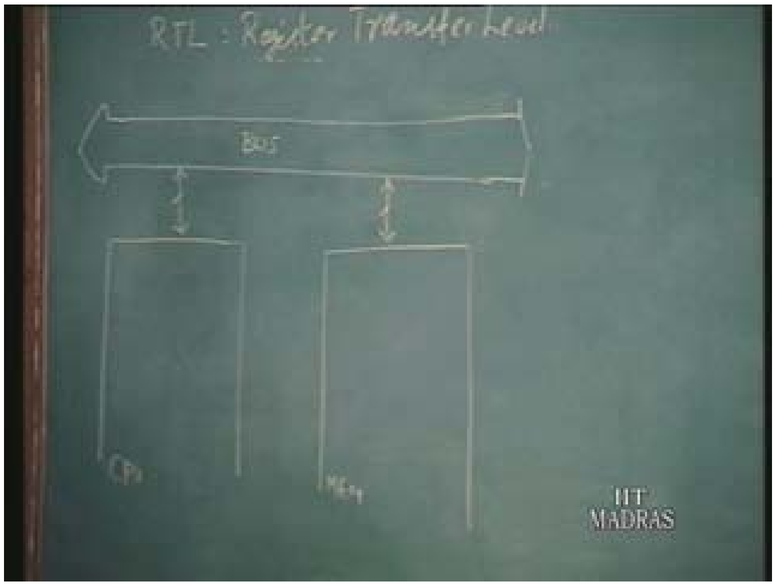
Computer Organization
Part – I
Prof. S. Raman
Department of Computer Science & Engineering
Indian Institute of Technology
Lecture – 5
Processor as a state Machine

In the previous lecture, you would have got some idea about what the so-called basic activity is or what we are calling RTL, that is, register transfer level activity. With reference to instruction fetch, I was giving you some idea about how the CPU and memory interact, and from that we saw what exactly we mean by the register transfer level. Now may be I will try to review that from the hardware point of view. I will just tell you conceptually what it was or we will just take a look at the same thing from a slightly different perspective. Now here is the bus, which connects the CPU and memory and then we have this CPU and memory sitting on the bus.

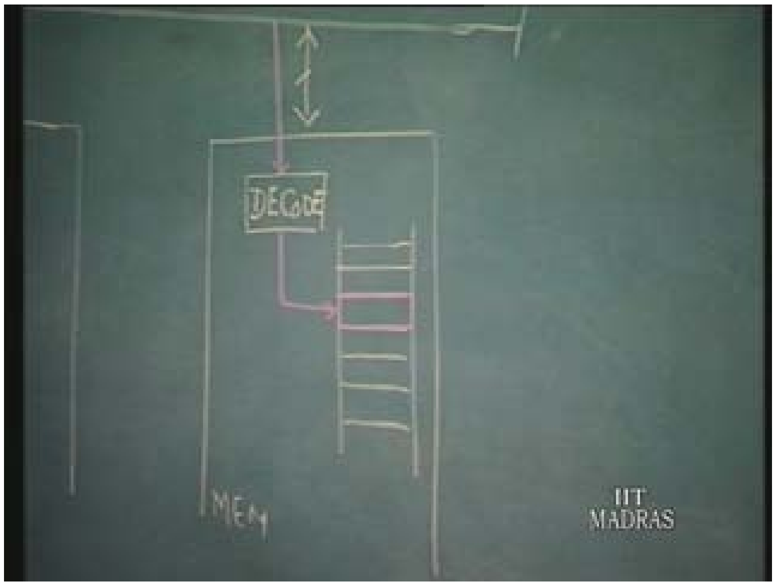
(Refer Slide Time 00:02:39)



Actually we have to go into some details of the CPU and memory here. That's the CPU, let us say, and then the memory; Let us not really bother about how many lines and in which direction, etc. We have a set of lines – now I will go into the details of these. What did we say in the first state shall we also work out what's actually happening. At the state level, we said that the state is defined by the clock period. Each one of these clock periods forms a state. If you recall, we were talking about four states in the instruction fetch, constituting the machine cycle. That is what we were seeing yesterday; we will work it out again. That entire thing is the machine cycle and we took the example of instruction fetch. It can be any other thing also, and the next machine cycle continues after that.

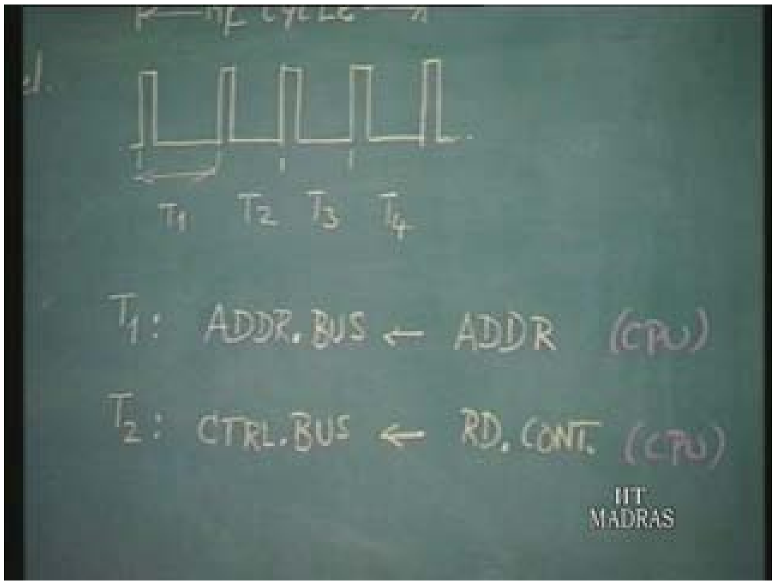
We said the CPU places the address; now I am going to give the details of this set of lines. The CPU places the address. We said earlier that the CPU will have some registers, different types of registers. Now I will just call this particular one, from where the address comes, as the address register. So the CPU places the address register on the bus, some set of bits code. What about the organization here? This consists of different locations; so this address will be received by some part of the circuit, which will decode that particular address, which will help select one of these. Let us say, for the address that is placed here, this is the specific location that has been identified.

(Refer Slide Time 00:05:17)



So the contents of that location in our case will become the data. What exactly is the data here? Because we said it is instruction fetch, this data is nothing but the instruction. This particular one is an instruction; CPU is trying to fetch that from the memory. So it places the address – we call this particular one T_1, the next period as T_2, then the next as T_3, and the next as T_4. Now let us write down: during T_1, we said the CPU places the address on the bus and also we saw subsequently a read control is placed on the bus and memory will place data on the bus. Obviously, when we say bus, these are different parts of the bus; now we may just say during T_1 the CPU places the address on the bus, so I will just call that part of the bus as address part of the bus. I am just qualifying this. That part of the bus on which the CPU places the address is actually the activity of CPU. CPU has placed address on the address part of the bus and we also saw in the previous lecture that during T_2, it is generating a control signal. We just call it a read control signal; I will say read, just to indicate that it is from the controller. I will use it like this (RD CONT) and then the read control signal is placed on the control. I will use some other control part of the bus; this is also done by the CPU.

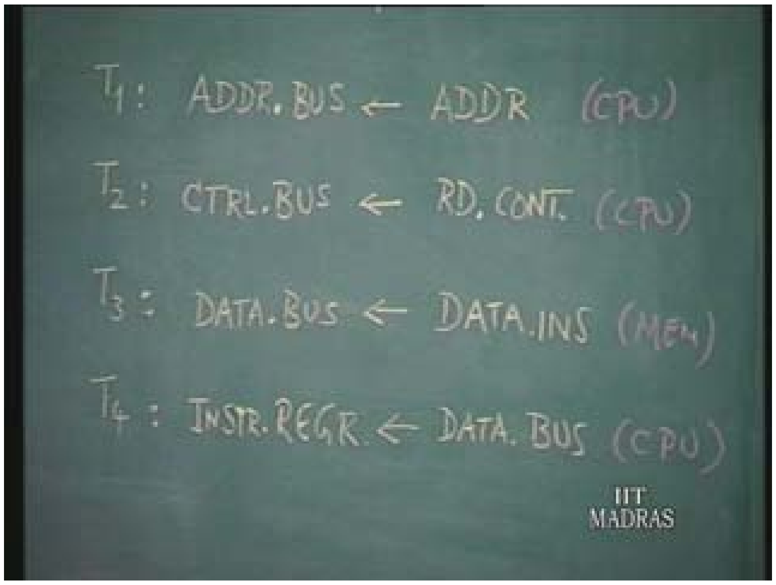
(Refer Slide Time 00:07:34)



Actually the CPU indicates that it wants to read something from the memory. Being an instruction fetch, we know that it wants to fetch the instruction into itself; that is, from the memory into itself. Let us say that there is a controller with some control part of the CPU, and the CPU, in this specific case, places this signal read. The CPU has placed the address; CPU has indicated that it wants read to be done. Now receiving this signal must trigger the memory to respond and the moment the address is placed, you can just take that address has been decoded, that is this is the part this address has been decoded. And in the entire memory organization, one specific memory location has been identified because of the unique address. Then when the read signal comes, I will just assume that this read signal goes to some control part of the memory.

It is some control part of the memory; that is, the read signal goes to some control part of the memory, which enables this to be placed on the bus; that is how it must be. This particular one, the contents of this, must go on to the bus. What exactly is it? It is actually in our case instruction, because this is a machine cycle referring to instruction. But just in general we will call this the data; during T_3 the memory responds by placing the data. What exactly is this data? In our case, it happens to be data whatever are its contents. In our case it happens to be instructions – if you want I will just identify that as instruction – and this is the content of memory. This has gone on to the bus as we said in the previous class. Actually the part of the bus is the data part of the bus. So now you can see that although different things come on the bus, we are getting a clear picture that the different things can be identified by placing them on different parts of the bus. Then, if you remember, we said during T_4 the data that is available on the bus will be read by the CPU into a register, and this being instruction, we also said that this goes into instruction register.

(Refer Slide Time 00:11:25)



T_1: ADDR.BUS ← ADDR (CPU)

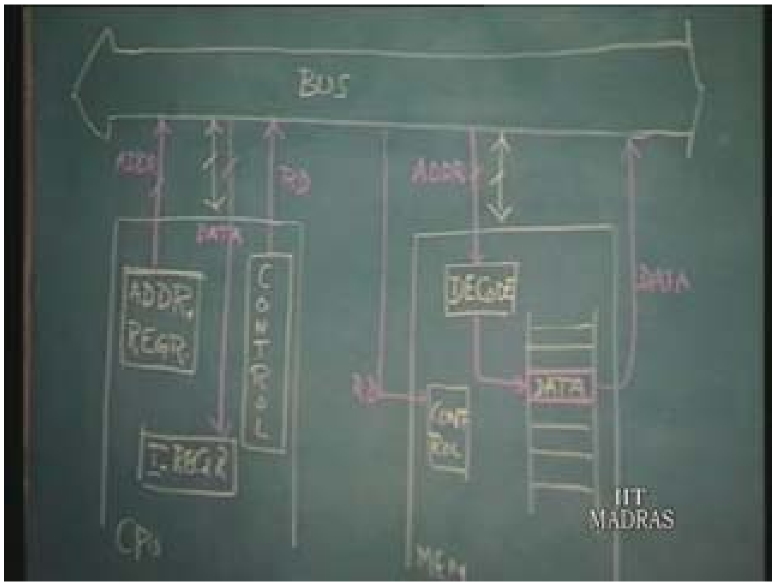
T_2: CTRL.BUS ← RD.CONT. (CPU)

T_3: DATA.BUS ← DATA.INS (MEM)

T_4: INSTR.REGR ← DATA.BUS (CPU)

Let us identify this as some register, that is, the instruction register. So the data that is coming over the bus is routed to this particular one – this in fact is the data.

(Refer Slide Time 00:11:47)

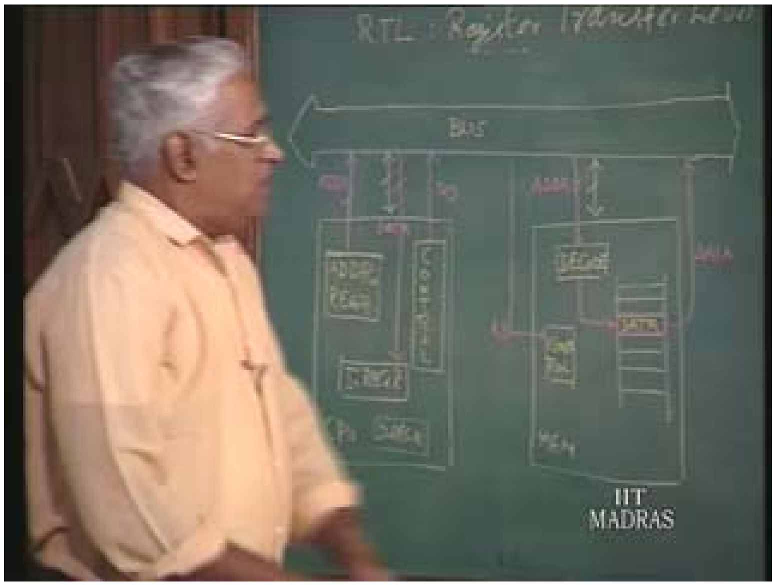


Is it clear what we mean by register transfer level? So from this register we are putting on a set of signal lines; one transfer takes place in T_1. Then from this part of the CPU, the control circuit, a 1-bit signal is the read signal that goes to the bus; that is, another transfer that takes place during T_2. Then in T_3, we are assuming there is one assumption that is made right here.

We will talk about it a little later; we are assuming that the memory is ready to respond because it knows from where to get the information, that is, the data in this case being instruction, and it also knows what it should do. Now this being read, obviously this means the CPU wants to read from the memory. So this is placed in the next state, T_3, on to the bus, then again it is assumed that CPU knows by the period T_4 starts that the data is available, and then the data is transferred again I register.

Now on the other hand, if it were not an instruction fetch, but it is say an operand fetch or a data fetch – suppose it were an operand fetch, which normally also we call data. That's why I told right in a few lectures before when we use the term data, be careful; it is a very general term. Operand will also be referred to as data the contents of any memory location can also be referred to as data. Now instead of instruction fetch, if an operand fetch comes, obviously the contents of that location will not be routed to instruction register, but they will go to another register called D register or some data register. So this is how the architecture of the CPU is also evolving – we will come back to this a little later.

(Refer Slide Time 00:14:30)



Now these are the basic activities, and most of these take place during one clock. Suppose the clock period is 1 MHz, what we are saying is within 1 microsecond, this transfer takes place. If it were 10 MHz, then within 100 nanoseconds, this is taking place and so on and so forth. Let us try to learn a few things here: one thing that we notice is the bus. We started saying that there is a set of signal lines for communication or interaction between two things on the bus, specifically in our case, CPU and the memory. We are seeing specifically that there are different things that come on the bus; so in one case address comes, in another case a control signal comes, in another case the contents of a location, which we generally call as data, comes. So obviously the bus consists of different parts and specifically we can even generalize from these that there is an address part of the bus; there is a control part of the bus; and there is a data part of the bus.

Now we have some idea about the bus also; that is a bus essentially will consist of address, then data, and also the control part. You remember I was talking about an 8-bit CPU or a 16-bit CPU and the memory which is organized. An 8-bit CPU means it is capable of dealing with an 8-bit data; in that case, it is meaningful to have the data part of the bus, which is 8-bit wide. It is meaningful, but it is not strictly necessary. For instance, if it were a 16-bit CPU and the data part of the bus happens to be 8 bits, it only means that in two steps, the 8-bit data will have to come from the memory and the CPU will start processing after it gets those 2 bytes. So you would find for instance that if you take an 8-bit microprocessor and that means essentially it is an 8-bit data and the address may be a 16-bit address and control signals depending on the complexities of the processor. So for supporting a processor, which handles 8-bit data, 16-bit address here also, we would expect an 8-bit part of the data bus, 16-bit part of the address bus, and the control signals needed for the processor–memory interaction. We will see more about bus later.

Now let us come back to this state activity, that is, the lowest level activity. I said we had made some assumption here – what is that? We have assumed that as soon as the CPU demands that it wants to read something from the memory, given this address, the memory is ready with the data. CPU is addressing and then CPU indicates that it wants a read to be performed. We are assuming that as soon as the read comes, the data will be made available by the memory and then will be placed on the bus. Where is the guarantee? The processor may be too fast or the memory may be too slow. For instance, if this period corresponds to, let us say, 100 nanoseconds, and if the memory is not that fast, it takes more time. Suppose from the time it gets the address and the control signal in this case read signal, suppose the memory takes 200 nanoseconds, that particular thing is called a memory access time. If it were 200 nanoseconds, the time taken by memory to access the data and put it on the bus is the memory access time.

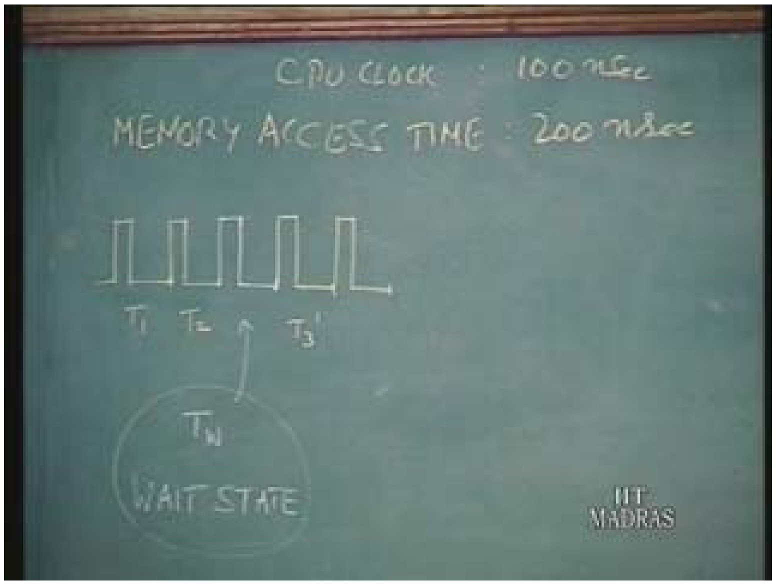
We said the CPU clock, that is, the CPU state clock period that is corresponding to the state duration, suppose each is 100 nanoseconds, what is it from the time the address comes? The memory has two 200 nanoseconds, but from the time the read comes that is during T_2, the memory has only 100 nanoseconds. You can assume that access time means after the memory gets all the information it needs to prepare the data, which means we are talking about after T_2.We find that after T_2, we have only just 100 nanoseconds, that is, during T_2, it generates the read signal and then we have only 100 nanoseconds and that is not sufficient for the memory, which is slow memory. What will it do? We have not bothered about that; we have immediately assumed and then said as soon as CPU wants memory will respond.

Now in this particular case, it is not really so. The memory needs more time than the CPU, that is, here we have a faster CPU and a slower memory. So this situation can very much arise. So obviously, in the next state, T_3, it cannot be ready – that is what it means. So we have to introduce what is known as an extra state, called a wait state; in other words, we will have T_1; then we will have T_2; and then we have to see that the memory will have to indicate to the processor that it is not fast enough, in which case, it is the CPU which has to take the decision. But it must have enough input – we do not know what is here; whether it is a fast memory or a slow memory.

Now, assuming that the memory can indicate that it is not fast enough, this particular signal will be coming from the control part. This particular one again will go to the control part of this. I will write it here. So this particular signal basically indicates that the memory is trying to tell the CPU that it is not yet ready to take on the task assigned. This particular thing is called a ready input. So the memory generates a ready signal – what it does is it will make 0, which means it is not ready. When it makes 1, it means it is ready.

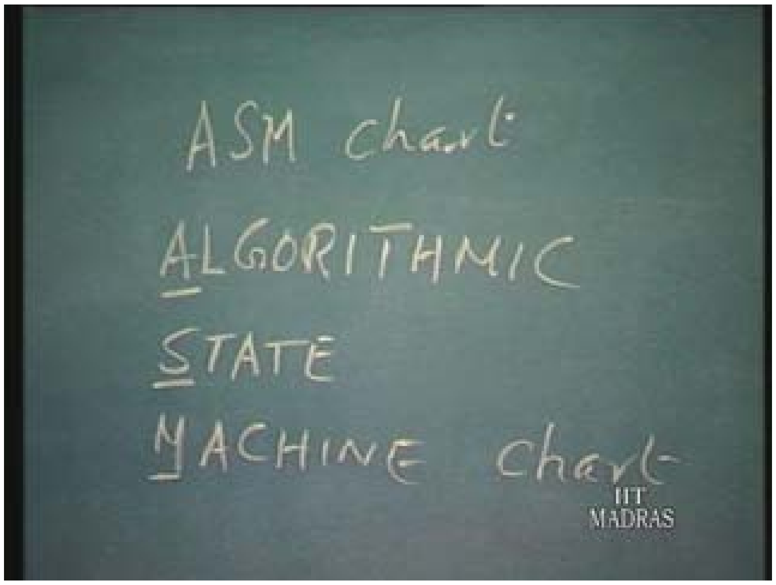
As soon as the memory gets address and reads, that is, after T_2, the memory will generate the not ready signal, which will be sensed, and on sensing that particular one, the CPU will introduce one more state. If it introduces it is enough; so we will just call this particular one as T_w or wait state. Basically may be it is not ready. In this case, one more state is enough, and then we can continue. Then essentially what we have here is the T_3 part, but it is not the same, as T_3 is extended. So we will just call it $T_{3'}$. It is different from this because a wait state has come in actually. Now where exactly will this particular information be included in our design?

(Refer Slide Time 00:24:30)



Let us go back to that chart I started with. In the last lecture I was talking about computer or CPU as a state machine. That is how we started looking at what the state is and then I was showing a state box. Now let us try to draw that chart for this right now. Incidentally, I think I did not mention that particular chart is not very much different from a flowchart, because it basically tells the flow of activities. But it is slightly different from flowchart in the sense that in the flowchart generally, we do not introduce the signals with some names. Here we will introduce the output signals and the input signals with names. So this actually is called a state machine chart, because that particular chart will be describing the flow of activities or algorithms. This one is called an algorithmic state machine chart, or an ASM chart.

(Refer Slide Time 00:26:23)


ASM chart

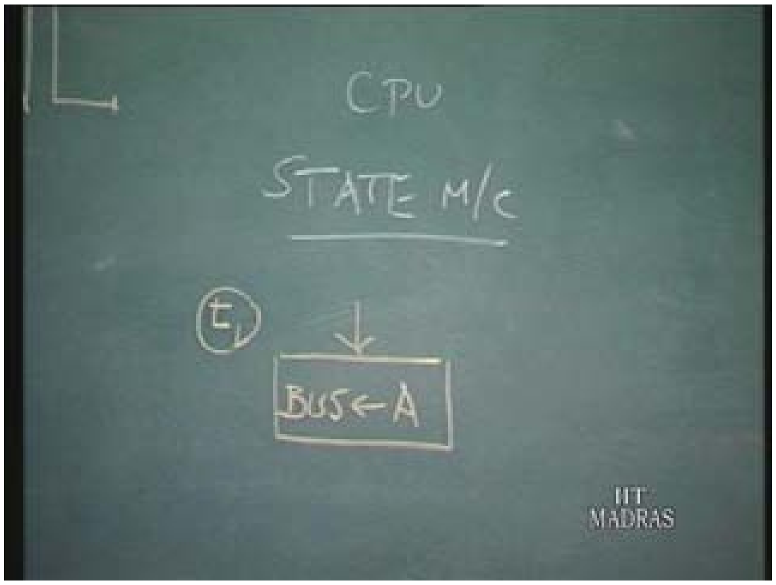
ALGORITHMIC

STATE

MACHINE chart

So let us try to develop this ASM chart, which I introduced in the previous lecture. We have to name this state – I will just name this state T_1 – it corresponds to our first state, T_1. In state T_1, the address must be placed on the bus. So I will just say the address is placed on the address part of the bus.

(Refer Slide Time 00:27:09)

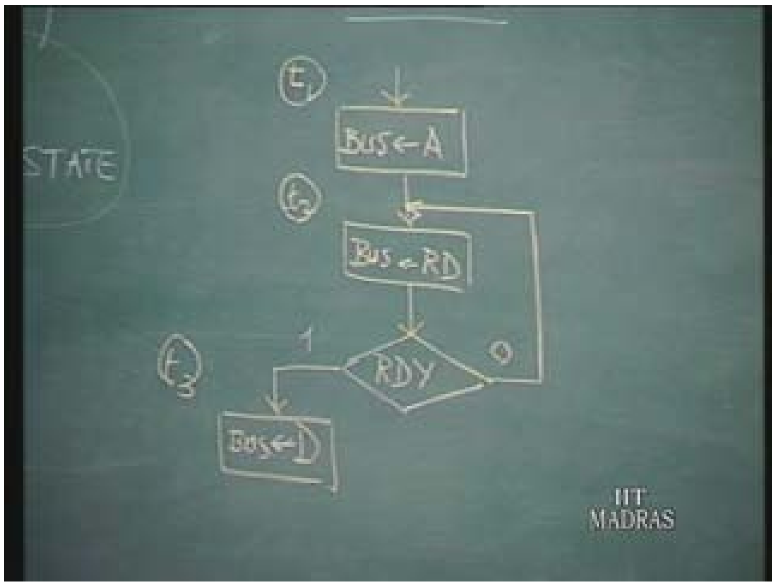


This address is placed on the bus; that's what is taking place in T_1. Next it goes to the next state, which we know is the state T_2. I will just name this next state as T_2, in which case read control signal goes on to the bus. So this in fact is called the state box; in the state box we indicate the signal that is output.

Why do we call this as output? Essentially you can see here the controller in the CPU must generate an output so that that output will make this address register place it on the bus. So we achieve the purpose; actually when we say output what we are talking about is the controller's output. So the ASM chart, which we are drawing, is for one instruction fetch. Essentially what you see here is the end result. At the end of this state T_1, the address will be placed on the bus and that is caused by the controller's output enabling the address register to go on to the bus. So address going to the bus is the end result so the controller must generate the signal, which causes it. We will come back to that a little later if necessary.

Then the controller must generate the read signal and put it on the bus; that must take place in the next state, T_2. Now, instead of this we will assume this is the situation, in which case, we have to check whether the state of this input – this is the input which again goes to the controller – what is the state of this ready input? Ready input goes to the CPU; that means essentially, it goes to the controller of the CPU. What is the state of this ready input – is it ready or not? Is memory ready or not? So if the memory is not ready, that is, 0, then just stay in the same state, meaning the read control signal, which may have been generated, extends it further that is it is not yet ready. On the other hand, if ready is 1, meaning the memory indicates that it is ready, then we can go to the next state, T_3, in which this memory data is placed on the bus. So I will just put it as D; this is in fact for causing this.

(Refer Slide Time 00:31:21)



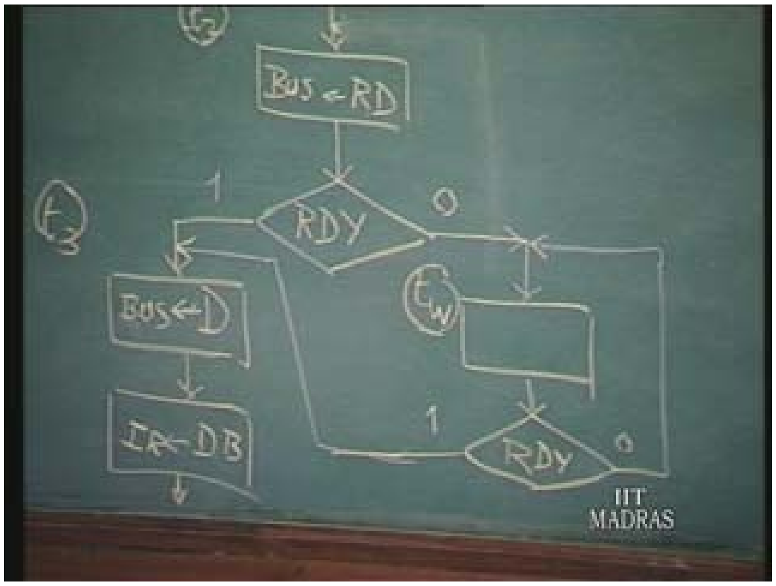
The data goes to the bus; actually the controller is not concerned with that; we only see to it that the controller keeps waiting. It will not go to that next one. So this is purely a memory activity. CPU is only waiting for that and then, once the processor goes to state T_3, the next is T_4, which means, after this the data that is available on the bus goes to instruction register.

So the data which is available on the bus goes to the instruction register and so on; it proceeds. This in fact is the ASM chart, which describes the activities, which in some way I have indicated here. This is for instruction fetch. Now you can see here that the CPU controller must generate this signal; as I already mentioned, the controller must generate the signal so that the address may be placed on the bus. In T_2, the controller must generate the signal so that the read control signal will go on the bus. And then after T_2, we are seeing that the controller is looking for the input; in fact that input is from the memory. So who knows that the CPU memory needs more time or it can respond at the same speed as CPU? The designer obviously – he chooses; he knows that the CPU clock is 100 nanoseconds and memory needs 200 nanoseconds.

Whenever the CPU demands, memory is going to respond after a delay of 200 nanoseconds; so the designer knows. Instead, if he had chosen a memory with 50 nanoseconds excess strength, there would be absolutely no problem. Even during part of T_2 itself, the data would have been available. So the designer chooses the CPU and the memory. And so it is the designer who will have to see that the controller, which is associated with the memory, generates this ready signal or not ready signal appropriately. There is some logic: that is, when address and read come, ready or not ready must be generated. That is the logic which will be included in this.

On sensing the ready signal, the CPU controller decides whether it can go to the next stage or whether it should continue in the same state. If you want, this particular thing can be slightly rewritten – this part of it – if, for instance, this particular thing should not be generated, it can be slightly modified and we can introduce the so-called wait state like this. When it is 0, then it goes into a state called wait state T_w, in which case, it does nothing. It just waits and then it keeps checking whether the ready is 0 or 1. If the ready is 0, it continues in the same wait state when it is 1, it comes to this point. We can either have it that way or this way.

(Refer Slide Time 00:36:18)



Both are permissible – it all depends on how exactly this read will be available to the CPU bus or whatever it is. I hope now you have got some idea about what a state machine is – state machine is one which goes through a sequence of states and in each state it generates an output, and whenever necessary, it also looks for an appropriate input, and then it decides what must be the next state. For the present state T_1, surely the next state is T_2, whereas for the present state T_2, the next state can be T_w or T_3; it depends on the state of the input. I have just done for one machine cycle, so it must be done for all the machine cycles. Depending on whether what comes over around the memory is instruction or part of an instruction or operand during data fetch or operand fetch, then appropriately that particular thing will be routed to different parts of the processor.

So you can see that when we talk about this, essentially we are talking about the sequence in which the various control signals for CPU must be generated, except in this particular case, we note that though we have marked here it is not the CPU control which generally does it. The memory responds. In fact from CPU point of view you can even remove that; that is what it means, because just like we do for this particular ASM machine, we have to go on doing for all the these machine cycles. A machine cycle is part of an instruction cycle; we are talking about instruction cycle consisting of different machine cycles, and we have just taken an example of one machine cycle. Now we have to do for all the instruction cycles, which mean we have to take for all the instructions supported by the CPU or all the instructions of the CPU, and then work out this low level activity, that is the registered transfer level activity.

When the sequence is ensured, we say that the controller goes through those steps and, depending on the nature of the input and a state; it will generate the appropriate output and a next state. So the starting point of the CPU design is the instruction set. The starting point of design of CPU is the instruction set, and, depending on the application, the set of instructions will vary. So start with the instruction, and for each instruction, work out what are the machine cycles, and then for each machine cycle, find out what are the RTL level activities, and that describes the behavior of the machine, in this case, the CPU.

That is why we say the ASM chart describes the algorithm and what exactly are you getting at here? You are getting to know the behavior of the controller of the CPU, which will ensure the various activities as given in that. So the ASM chart for the system – when we say CPU, we are talking about the ASM of the controller algorithmic state machine and then we develop the state by state behavior of the controller and that controller will generate the necessary signals for these various operations. Now that brings us back to two things: what did I say earlier? I said the CPU design essentially consists of data path and the data path control. So now I have been talking about this control. What is the data path? The data path essentially is the path over which the data should move. What is the data in this case? In this case the data that it handles is the address; the data that instruction register handles is actually instruction. So the data path and then for the bus point of view, the data comes over the data part of the bus.

If it is an instruction, it gets routed to instruction register; if it is data it must get routed to the appropriate data register. So now, to get some idea, we have to go into the details of these what are the various registers that we have and how they are internally organized. In other words we have to go into the architecture of the CPU to further understand what is going on. The further aspect of the design will depend on how exactly you organize the set of registers. So there will be some registers, which handle only registers; some registers which will handle only data; and some registers will handle instruction; and some other registers for some other purposes say included for controller and so on and so forth. Having got some idea about what is the control sequence, we have to go into the architecture of the CPU. Now before we go into that, I will mention one important point here. Let us not forget that the computer ultimately executes a program, which consists of instructions, and the instruction is executed during an instruction cycle, which consists of machine cycles, and then we went into the details of the machine cycle, that is, the states.

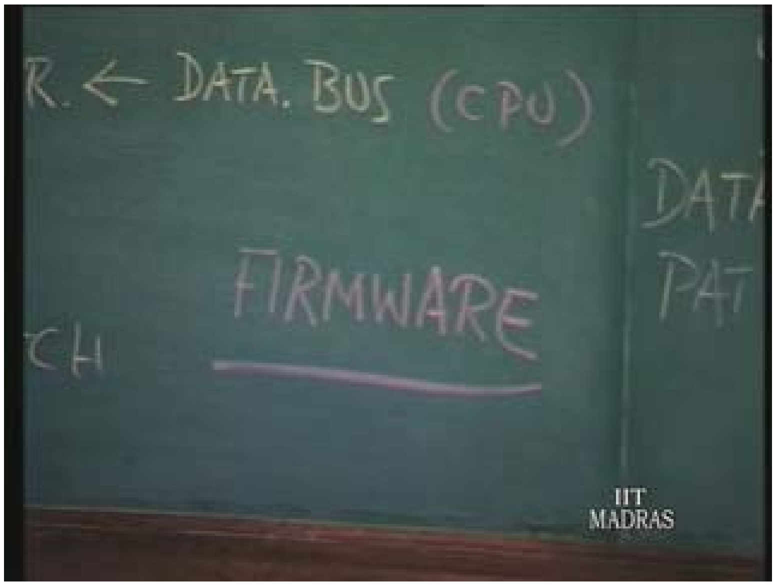
The program is executed one instruction after another instruction, which means from the CPU point of view from one address to the next address and so on and so forth – this is the normal sequence. So actually this particular address register though just called in general as address register, may come from one counter; for instance the counter initially may point to this and the next part of instruction or next instruction may come from the next one, which means this plus one plus one and so on and so forth. Specifically the address register which gives the instruction address is called a program counter; of course the particular sequence may be changed subsequently that is something. Similarly, there may be other instructions which make use of some other registers; so this is very common and then through the address register actually the source for this address register will be program counter or some other say memory address register and so on and so forth.

On this side we have just a path that is set up for the address; incidentally these various signals form part of this bunch; they are only blown up and then the details are worked out. Now another thing is again related to the program and the sequence of instruction, execution; that is, when it comes to the last point that is the state. It is a small step. One register transfer level activity is one that is going on. In other words, the program consists of sequence of instructions and ultimately when one instruction is taken, essentially we have a sequence of these steps. These are actually called microinstructions, because we are talking about an instruction consisting of a sequence of microinstructions. Now if the particular programmer can develop a program at this level, then he is called a micro programmer.

There is a difficulty here – if one can develop the micro program, what happens? The whole character or behavior of the CPU is altered. So really we are talking about a fine grain level programming, whereas usually the application program is written at a very high level. He will not bother about all these lower level transfers. But if the programmer has given the capability to write the program at that low level, the micro program is really altering the very basic character of the controller, which means he is altering the basic character of the CPU itself.

There are some micro programmable processors also, but remember that that is not true of any general purpose processor. This control unit can be designed making use of the hardware, which implements these micro steps or we can have a micro program, which indicates these and then we can have a controller based on that. If the controller can be designed using hardware circuits or you can have a micro program, which again in turn indicates these micro steps, this becomes a micro programmed controller. At that level from the general purpose user point of view, he will not be able to alter. We will talk about this micro program controller and the micro program itself in our subsequent lectures. But before that I would like to tell you one thing: that is, I said the controller can be rigged up or developed using hardware circuit and if we use micro program control, that particular thing will be called a firmware design. What is that?

(Refer Slide Time 00:48:59)



Basically either the sequence of control signals can be realized, making use of hardware circuitry; on the other hand, if you use micro programs, then by changing the micro program you can change the character and so as long as you don't change, it is firm enough. So a firmware design would be essentially meaning a micro programmed controller implementation. That is it for the present; in the next lecture we will take a look at the data path architecture, because having talked about control and how it comes about, it is time we take a look at the architecture or a typical data path; something we will assume and proceed.