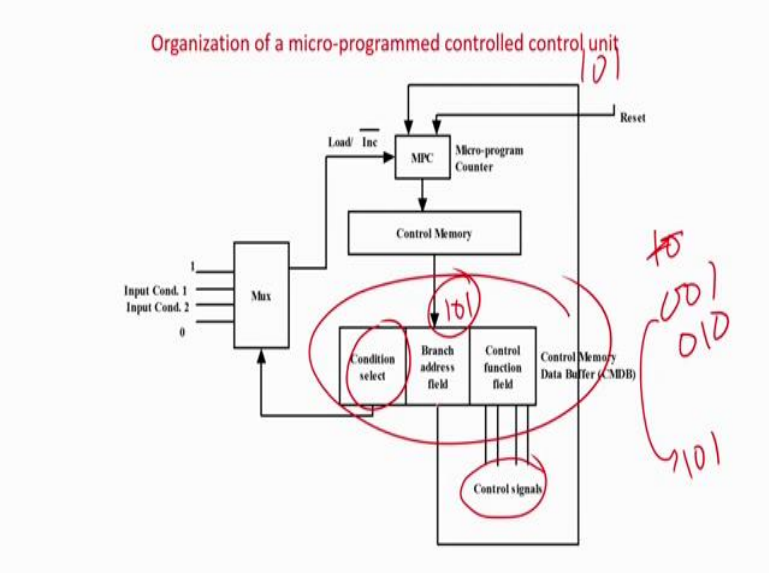
So, we will take some elaborate examples, which will actually clear out the whole theory for you.

(Refer Slide Time: 30:27)



This is the figure which will actually clear what I was saying, you have to take it in slightly elaborate manner, and let us look at it what happens. We basically have a memory. So, we have actually 3 fields. So, this is the control function field. So, this is very very important the control signals are like program counter in, program counter out all these things will be there basically that is the main part of it. There are 2 other parts basically one is called the condition select and one in the branch address field. Branch address fields means, it says that from this instruction if some conditions are true or something then the next address may be say 101.

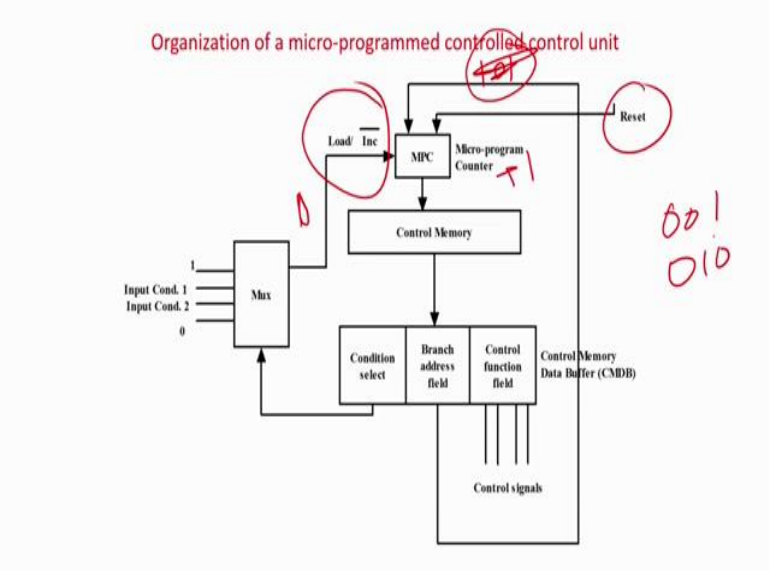
So, this is going to say that I am not going maybe at this present in the instruction called say 001. So, I will go to 101, if some condition is true. That is what this is saying and where I have to go, I have to go to 101. So, here we are telling which are the signals, here we are telling which is the instruction location you have to go, and we are saying that what is basically the condition it has to be met.

Now, how can I tell that I have to go to this branch? This is very simple 101 means you have to update the value of micro-program counter micro-program PC to 101. Otherwise it will become from 001 to it will 010 it will increment, but you are telling no I don't want to go to 010 rather I want to go to 101, which is basically 5.

So, if you look at it. So, this part of the control signals will directly go to the ALU the PC in and all these ports, but basically the branch address field is going to be basically one input to the MPC. So, if you allow or if the jump condition is true basically the MPC will directly take the value of 101, that is what is going to happen. So, this jump address is basically you are feeding as one input to the micro-program counter.

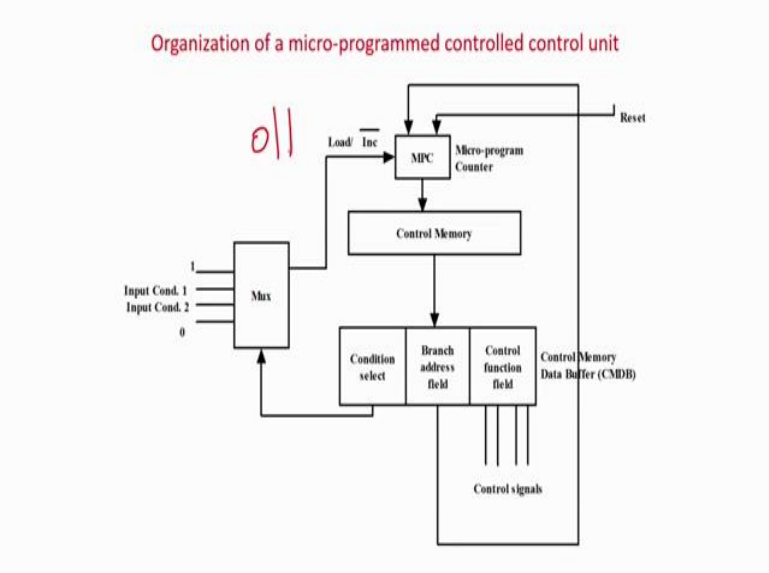
So, now, let us look at how the micro-program is designed. One is the reset field obviously if you want to start from some 0 location you can reset it, and this is what is the micro-program and the jump address. Basically, this input corresponds to the jump address which will load the micro-program counter to the jump address.

(Refer Slide Time: 32:32)



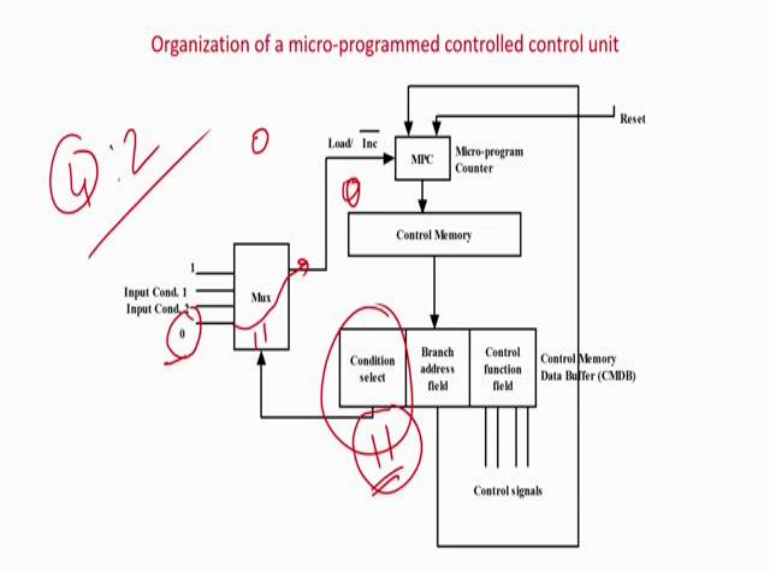
But, here you see this is very interesting there is a control to MPC. So, what is the control to MPC one is load and increment bar; that means, if this line is 1. So, what is this going to do it is going to load the value which is available at this port so; that means, if this is 1 you are going to take the value of 101 in the micro-program control and you are going to jump to 101, but if this line is 0 then actually what is going to happen it is just going to make it as $+1$. So, basically it was 001. So, if this line is 0 you will go to 010, and you will be incrementing. So, the logic is very interesting. So, how the implementation is very interesting. So, you see the branch address field is actually feeding to the micro-program counter. And if this control is 1 then you are going to jump and if this control line is going to be 0 then, you are just going to increment basically now, who tells that this line will be 0 and 1 then my job will be done.

(Refer Slide Time: 33:29)



How to decide that whether this line will be 011, that is going to be decided by this condition select field. So, for that they are actually be having a multiplexer-based implementation. You can have any other implementation they have a micro programmed based implementation.

(Refer Slide Time: 33:50)



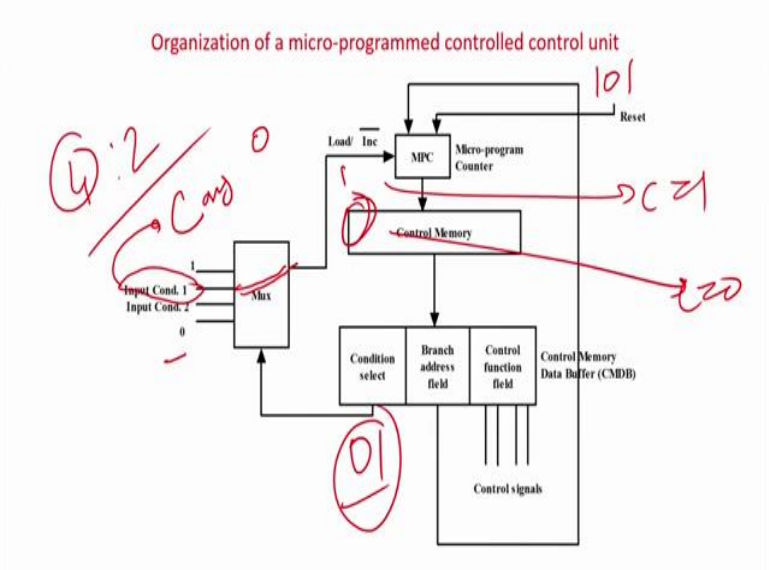
So, they actually say that say for example, I have only 4 condition codes. So, you have a 4:1 multiplexer as simple as that. If you have many conditions to be checked that is going to increase. So, I say that there is actually only 4 condition which has to check. And here I say that if I put 11 sorry if I put the value of code 11. So, if I put 11 in the multiplexer then of

course, this line is going to be connected to this. This I am making a default 0; that means, if the condition select is 11 then this is 0. So, this part is connected this I am putting a default 0 over here, then this line is going to be 0 and I am going to just increment the micro-program counter; that means, what; that means, very interesting that if I put the selection code 11, then the last part may be the last bit of the multiplexer is connected to the MPC control and this is hardcoded to 0.

So, if I give the condition as 11 you are always giving to get 0 and the micro-program will be just increment; that means, if I give the condition code as 11, then whatever in the branch address I need not bother always there will be an increment in the micro-program. So, it will go from 2, 3, 4, 5 like that there will be no jumps. But say for example, I want to check if the I want to go to 101 location if there is a control flag which is 1, say this is a input 1 that is the carry flag say input 1 is connected to the carry flag.

Now, I will give the condition here as 01. So, if it has 01 the second bit will be actually connected from here to here. The second the second bit is going to connect from the input control to the line over here, now I am going the control 1. So, the now actually this line is going to be 0 and 1 will totally depend on the condition that which is connected to this.

(Refer Slide Time: 35:02)

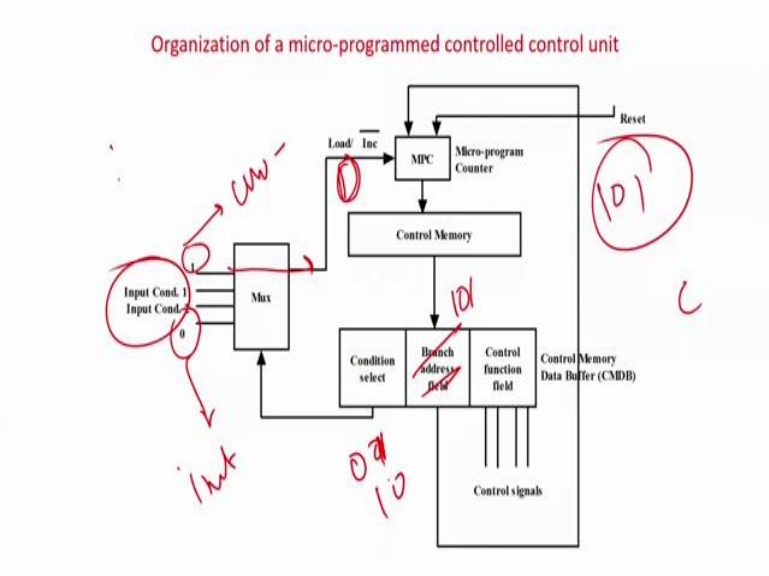


So, here I have a connected carry 0. So, if I put the 01 say then this line will be 1 if and only if carry bit equal to 1, and this line is going to be a 0 if the carry bit equal to 0 so that means what, if I put selection condition as 01 and if the carry bit is a 1 so you are going to load 101. I put a

01 and if the carry bit is a 0 you are just going to have a 0 over here and it means, it will go to just I increment mode it will actually execute the instruction number 2.

So, that is how actually it happens and of course, if you want to go for jump unconditional then, somehow irrespective of this basically if I want to have basically a jump unconditional that means, I just want a 1 over here and whatever address I put it should go over there; that means, here the condition should not matter anything or whatever condition I put over here should not matter anything.

(Refer Slide Time: 36:12)



So, I will put the condition as 00 which is the default condition in which it's a jump unconditional. So, if I put 00 the first bit of the multiplexer will be connected to the output, and here I have put a default 1. So, in that case this line is always going to 1 and you are by default jump to 101, which is specified over here. So, this is how actually it is implemented.

So, nicely looking at it you will find the interesting stuff over here. So, you are going to see that first bit is a 1. So, if I put 00 this 1 is going to be 1 and whatever is the value in address field will come over here, and you are going to jump to that instruction. Last bit I am always explicitly keeping as 0; that means, if I put over here over condition 11 then, this going to be selected this thing is always going to be 0 and you are just going to increment.

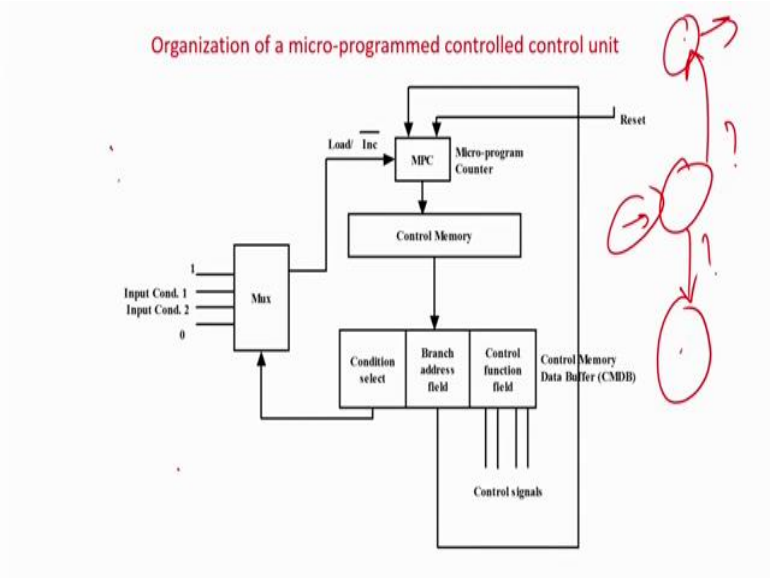
So that means, if I put a 11 this branch address basically has no meaning, and if I put a 11 then no condition has if I put 11 over here, the branch address has no many because it is just going

to be incremented but, if I say that if I put 00 over here. So, in that case it is a jump is very very important here actually you are going to take 101 or wherever you want to jump that becomes very important. Because, this is going to be connected over here, but only thing is that no condition bits are required you directly jump to 101. So, this 1 corresponds to basically unconditional jump, this 1 corresponds to just increment. So, in in case of increment this has no value.

But for the 2 other conditions like 00 sorry 01 and 10 some conditions will be checked. So, how do I get input to the conditions this may be this I connect to carry input, and this I connect to see maybe parity input. So, and then, if I put 01 then you will jump to the instruction only if there is a carry, and if I put 10 as the control condition select then you will tell that if and only if parity bit is set, then I will jump to instruction 101.

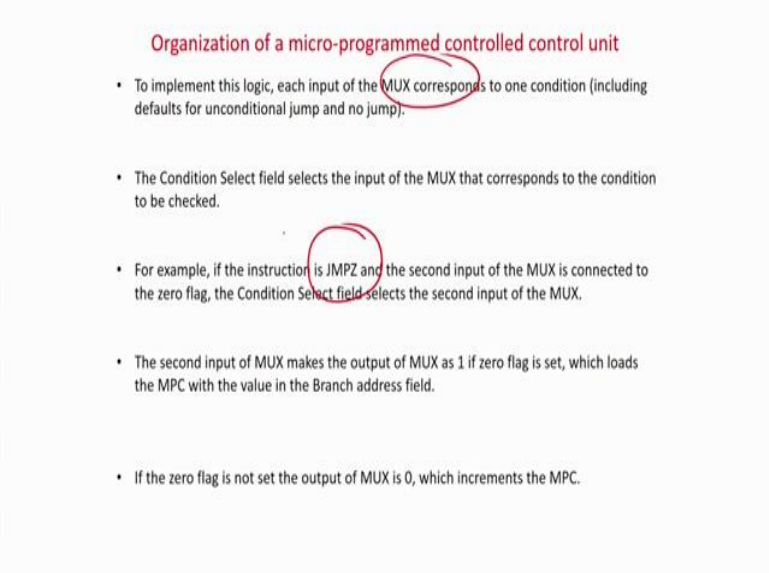
So, therefore, we have seen how this circuit basically implements the sequencing using a memory, because we have to remember that un unlike if I said we do not have some flexibility like this, where some condition checks are here and you go from this state to this state based on some condition, and control from going to here or going to here is done by state encoding that is not possible.

(Refer Slide Time: 38:23)



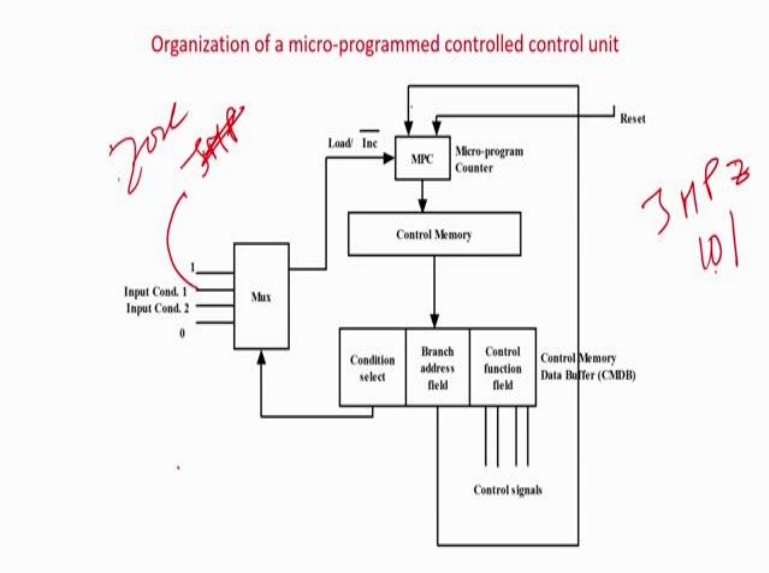
So, we are actually incrementing it in this nice interesting digital design fashion. So, this is basically the organization of a micro-programmed control unit.

(Refer Slide Time: 38:45)



So, basically whatever I have told you is actually written in the slide. So, to implement this logic we actually this use a multiplexer. So, the multiplexer corresponds to jump unconditional, jump conditional and just increment basically the, that is what is the case. So, if it is a jump on 0. So, some 0 flag has to be checked.

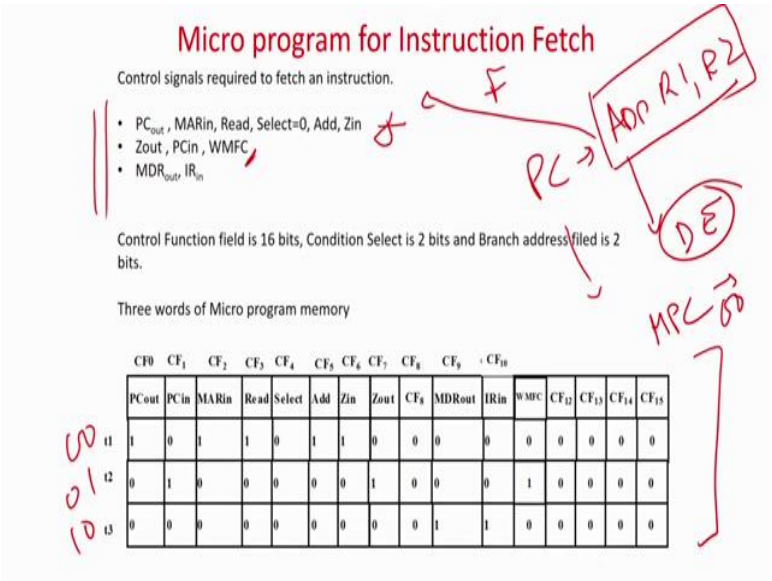
(Refer Slide Time: 39:09)



So; that means, if you have to ah. So, if you if you have some instruction like jump on 0 to say 101. So, this 1 may be connect you have to connect it to the 0 flag. So, it is a 0. So, this 1 will be a 1 it will be loading the instruction 1. So, therefore, means it has to be set. So, this is how

it is done, if the 0 flag is not set multiplexer is 0. So, it just increments the MPC whatever I have told you. So, whatever I told you is explained in the thesis I explaining the slide you can just go through this.

(Refer Slide Time: 39:32)



Now, example so much theory we have told and we have explained all these terms in case of a of a very abstract English, That if this happens to jump over there this is a micro-program control, and memory is divided into some control signals this part has to be for conditions, this part has to be jump address. And so, forth. So many things we have to discuss. Now, we will take an example and we are going to do it.

Say for as I already told you say $ADD\ R1$. So, this is the instruction which is to be executed that is the macro instruction. So, you already know that instruction is to be first fetched and then decoded and execute it. So, as I already told you. So, if so many times we have dealt with this, this is the micro-program, or control 3 micro instructions which corresponds to fetch. So, this one actually takes the value of program counter out to the memory address. You read the memory address you basically $select\ 0, Add$ and Z_{in} means that is for incrementing the value of PC, then the incremented value of PC by the way we are all assuming a single bus architecture, then the output of program counter will go to Z sorry the Z will feed the program counter.

Z is having the value of $PC + constant$ then you wait for some about of time, and then basically you dump the value of memory data register the instruction register.

So, the instruction $ADD\ R1\ R2$ comes to instruction register. So, this is many many times we have discussed it and for almost all instructions these are the 3 micro instructions required to fetch it. So, let us see how it so, whenever you have to execute this instruction automatically same signals for the program counter is fetching it. So, immediately what it will do? the program counter will MPC will load basically will be loaded with a micro-program memory. So, let assume that is a micro-program memory say it will go to 00, 01 and 10.

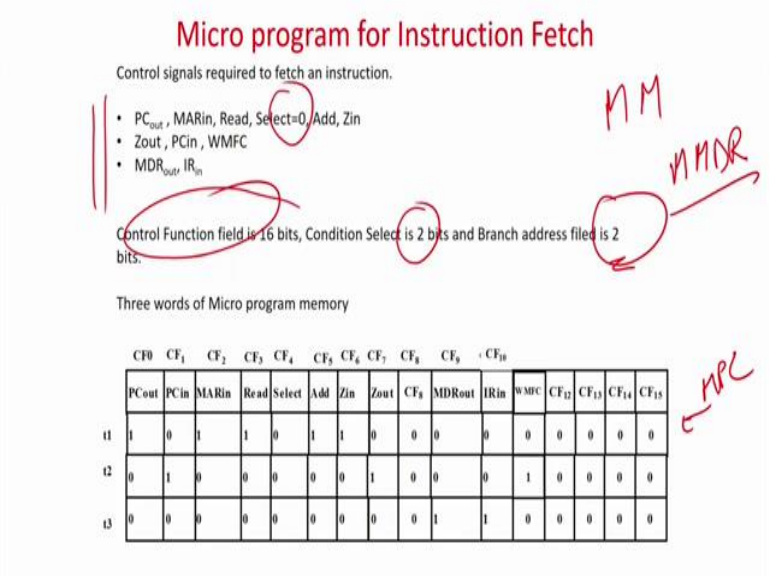
So, it will immediately make $MPC = 00$, which corresponds to the starting micro instruction for fetch that is this 1. So, whenever any instruction has to be executed it will make the MPC 00; that means, here we are assuming that the micro-program which corresponds to fetch of any instruction basically exists in this micro instruction memory micro instruction memory 00.

So, that is default. So, after this is fetching is done then we will see what happens basically. So, we are assuming that the control function is 16 bits; that means, we assume that there are 16 signals. In fact, we are not something may be vacant because, as I told you address widths are generally 2 bits, 4-bit, 8-bit, 16 bit, 30, 32 bit and so forth, but the control instructions or the control bits may not be 2 to the power it may be because, you might have 10 of control bits like $PC_{in}, PC_{out}, MAR_{in}$. So, that may not be in the in the terms of powers of 2 because, that correspond to some basic hardwired points in the circuit.

So, here they are taking 16, but the control signals are basically how much 11, 12 basically well and they have kept 2 bits for condition select, and 2 bit is for the address because there are 3 basically in this case that are basically 3 memory words in this MPC sorry in the micro-program memory. So, address field can be 2 bits control there can be 2 bits; that means, they are assuming that control will can depends only your 2 bits, that is 2:4 multiplexer like that ok.

So, now let us see, what and we all know that in the first cycle PC_{out} should be 1, MAR should be in, $read$ should be 1, $select$ should be 0, ADD should be 1, and Z_{in} should be 1, that means these are the different points you have to set it. So, it is very easy here unlike a FSM. For FSM you have to design a circuit and synthesize it. Here, you just write the first bit corresponds to PC, second bits corresponds to PC_{in}, the first corresponds to PC_{out}, this corresponds to PC_{in}, this corresponds to MAR_{in}, read that means what?

(Refer Slide Time: 43:16)



This memory location will be read from the micro-program memory it will go to the micro-program memory data register, and this one this point actually will be connected to the program counter in, this memory data bus micro-program memory data bus will be connected to the PC input, this will be connected to the MAR input, this will be connected to the read port of the main memory and so, forth.

So, just this will be coming out of the main micro-program memory data bus that is data is in the buffer register. And these points of the address bus is directly connected to the corresponding positions in the hardware block. And so it is very simple that the memory data is coming from the memory and it is directly connected to the different points and the job is that is done.

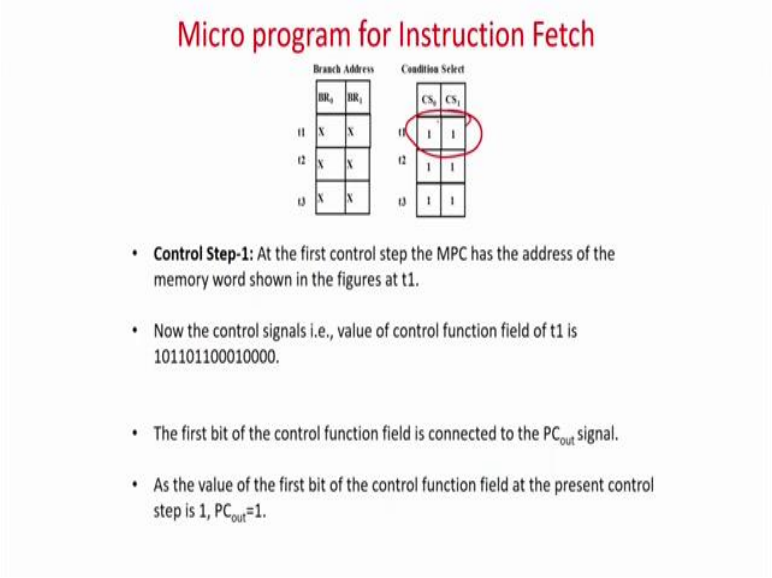
So, in the first thing this is cycle you know $PC_{in} = 1$. So, I will store a 1 over here PC_{out} is 0 all other signals which are not menti1d over here are basically 0 MAR_{in}. So, I will put a 1 over here, $read$ I will put a 1 over here $select$ 0 over here already told. So, I am putting it Add is to be 1, Z_{in} to be 1 and of course, all other signals here are basically 0, do that I do not require them. So, basically put it as a default to be 0. So, see $Z_{out} = 0$. This correspond to some other 0, MDR_{out} is 0, IR_{in} is 0, $WMFC$ is 0 and all other values you have to put it 0.

So, only whatever are mentioned over here you put it 1 and 0 as explicitly 2 explicitly it is telling that select has to be 0 because, program counter has to be incremented by a constant.

That we explicitly put as select as 0 and whatever mentioned over you put 1 and all others has to be reset to 0 that is done.

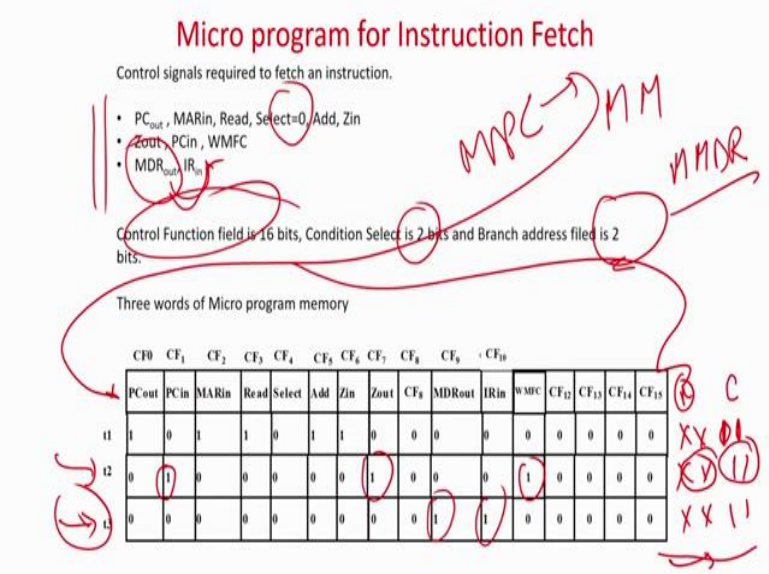
So, whenever MPC will point to here. So, these values will be brought from the mainly micro-program memory data bus, and it will go to PC_{out} 1 PC this is go to PC_{out} 1 PC_{in} PC_{out} is going to be 1, sorry PC_{in} is will be 0 are not do anything with it MAR will be 1. So, all the connection should be done the values will be open done and your job is done.

(Refer Slide Time: 45:17)



Next because, in this case there is no question here I will show you. This is actually corresponding to the branch and control select. So, in this case first I am putting at 11. So, if you look at what was 11, 11 is basically corresponding to this bit; that means, basically it is going to give a 0 over here. So, it is just incremented. So, if you look at it I am putting this one and branch address is 00; that means, basically I can write it over here for your ease. So, here I am doing it as an xx and control bit is 11.

(Refer Slide Time: 45:44)



So, control again I will write it that is useful for us ok. So, when I am set that is this control is basically 11; that means, I am not going to look at this next address is next address, I am not going to look at in next address anywhere because, the control is 11 means whatever be the control means input I will just increment the value. So, I put 11 here and your job is done. So, automatically MPC will become 2, it will be just incremented because 11 as I showed here corresponds to increment and not load it to any variable next location.

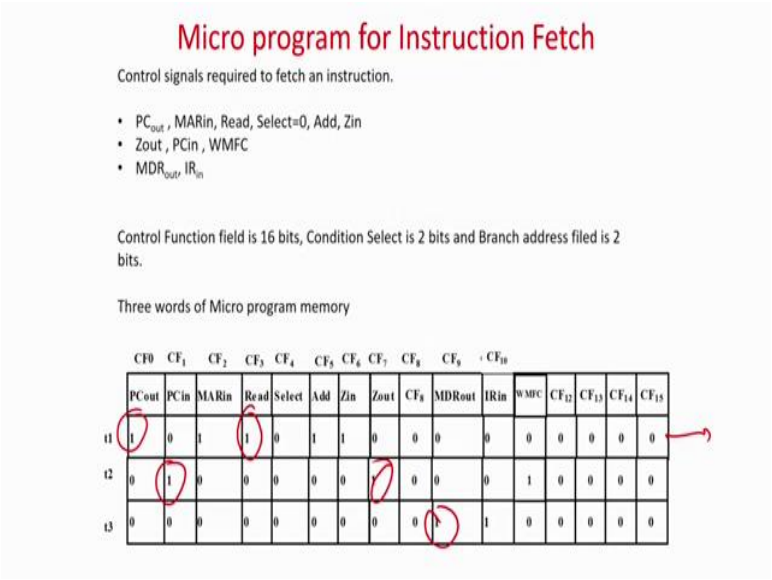
Next one next to see Z_{out} PC_{in} and WMFC, so in this case PC_{in} will be a 1 and basically Z_{out} will be a 1, and WMFC is the 1. WMFC 1 means, basically it's a signal which is generated and; that means, you are waiting for another signal from the memory, which has to come, if both of them comes then only you can go to the next one. So, there is a slight hardwired block over here, which I am not describing WMFC means, it has generated a signal which is called wait for the memory, memory said that I am ready you can read my data. When both the signals will come then automatically it will trigger and you are going to the basically the next instruction.

So, this part you can take as a black box, that unless this WMFC actually ensures unless another same similar acknowledgement comes from the memory that the data is ready, and it it has you can it has been put to the memory data register you can take the value basically it will not going to go through the and counter number 3. So, there is a hardware block for that, you need not put too much mind about it just assume that it happens.

Now what, now here also condition check is 11; that means, here also there is no question of any kind of branch you just wait, and in this case basically it is xx. So, no nothing to whatever I even if I put 00, 11 or 10 whatever here, but it is not going to jump to that place because, conditions are 11 that is default increment. Now, we will come to memory location 3. So, memory location 3 says that MDR_{out} because, already memory has been fetched and you are going to dump it to the IR. So, in this case you are going to have $MDR_{out} = 1$ and $IR_{in} = 1$, and then it is basically here after that this is actually next program next part will be $HALT$ or END and then your micro-program count the next I have not shown over here.

So, when it will come over here, it will correspond to END and then MPC will not increase. It will actually micro-program PC after 3 basically stops. We can have we can also put some condition for stop over here in the next space. It will not get incremented to 4 rather it will actually give floating. Now, what happens basically? The instruction register now the value has come to the instruction register. It will decode what is the instruction, if it is an ADD instruction then MPC will correspondingly point to some other portion of the main micro-program memory, which corresponds to such a code which corresponds to ADD. It is a $LOAD$ instruction IR will decode it, and basically it will point to some other memory location some other memory micro-program memory which looks like this, and what is going to happen that will have the corresponding signals which will correspond to what which will correspond to basically the micro instruction sequence for $LOAD$ instruction and that way basically it will happen.

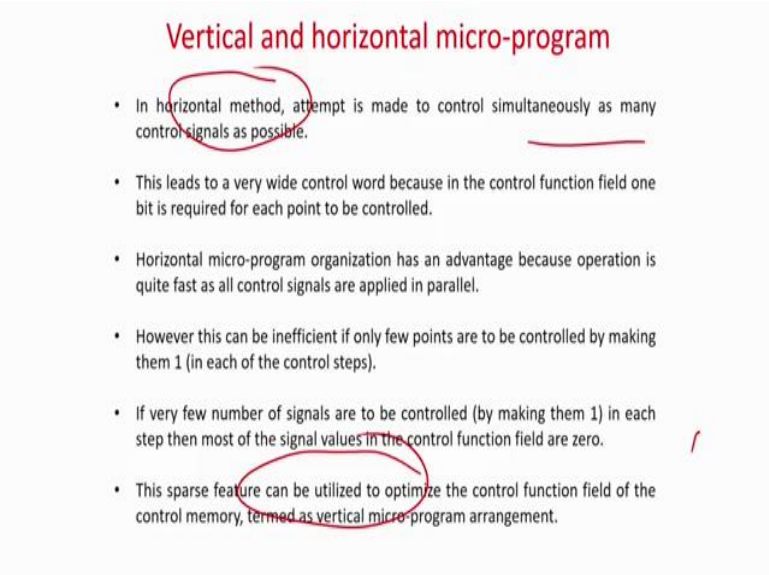
(Refer Slide Time: 48:55)



So, as we were looking at basically this micro-program controls. So, one thing you might have observed that whenever actually this control signals are coming from each of these memory words in the micro-program control memory, you can see that simultaneously you are controlling each port or each point we have to control in one go or in parallel, that is if you want to make $PC_{out} = 1$ you are setting it as 1, in the same time instant you want to make $read = 1$ you can make it as a 1 and all we are doing in a very parallel fashion.

But, if you look this matrix is basically a very sparse matrix like in the second time you only require 2 signals to be 1. And in the third you just require 2 signals to be 1.

(Refer Slide Time: 49:36)



So, what happens basically the there is a lot of wastage of memory, when you are talking about the micro-program control memory where you are fitting in the signals required to be make 0 and 1 at each time instance of the micro-program.

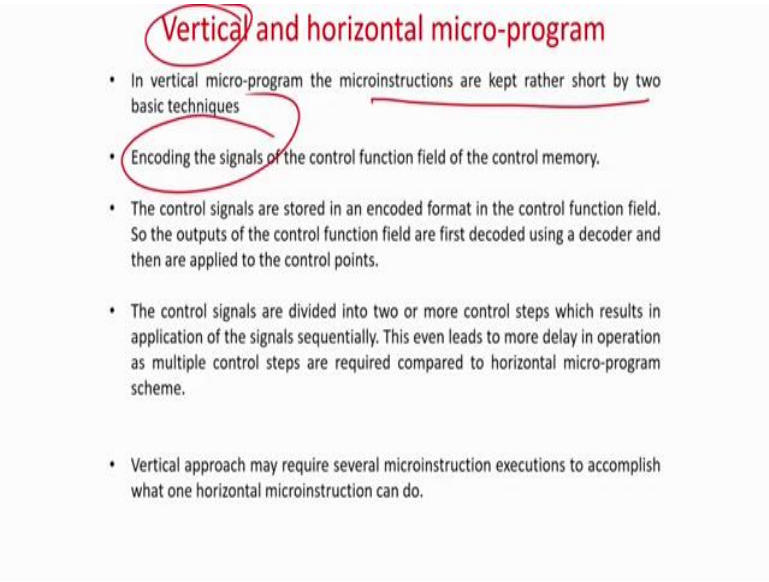
So, this actually method is called a horizontal method, and in which case you simultaneously control as many signals and possible. How it is possible? Basically, you have a memory in the in the micro-program control memory, and you put 0s and 1s as required and each bit of the memory word corresponds to one port. So, you parallely send all the signals to all the ports to be controlled and it is very fast. But, the problem is that as many ports are there that may that much long your word has to be. So, if there are some 32 points to be controlled the word has to be 32 and so forth.

But on the contrary if you will see you see lot of 0s so; that means, actually this has still sparse in content. So, if you have such sparse content, then you are wasting lot of memory. So, how can we actually implement in a better way. So but otherwise have advantages like in one go you can control all the control signals required to be given at an instance of time. So, it is parallel, it is fast there is no processing involved like an FSM. You need to find out at each state what is the control signal required.

So, you have a circuit which generate the signals from that state encoding and so, forth. So, that is slightly complicated in terms of circuit, but here it is very simple whatever want to give written in the memory just apply it, but there is lot of wastage because of sparse.

So, therefore, this sparse can be utilized to optimize basically; that means, somehow, you can if you can somehow throw away these sparse values like, where ever it is 0 you don't require them only you keep the 1s in some way so that the whole memory can be optimized. So, that actually is called a vertical micro-program.

(Refer Slide Time: 51:21)



So, in vertical micro-program the micro instructions are kept rather short by 2 basic techniques, and we will see about the techniques basically. So, what are the 2 techniques, one is a as we will see they are the 2 broad techniques. One is a hybrid technique and one is absolute vertical technique.