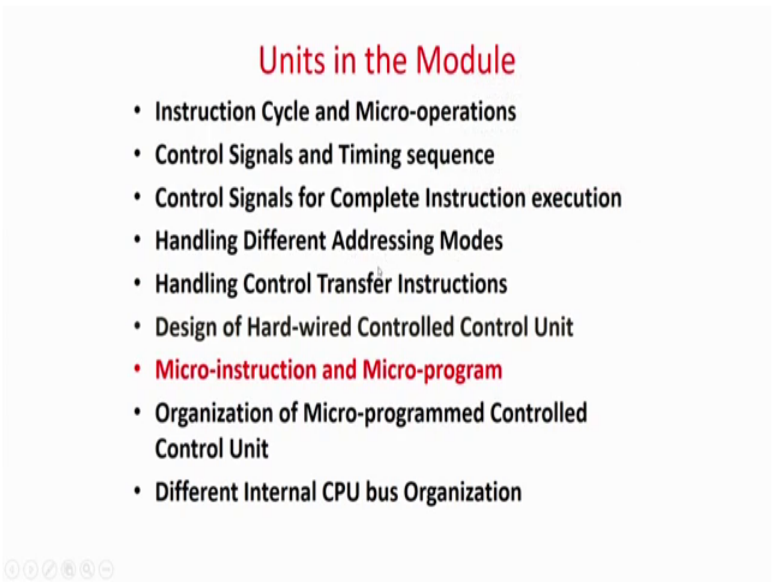
Computer Organization and Architecture: A Pedagogical Aspect
Prof. Jatindra Kr. Deka
Dr. Santosh Biswas
Dr. Arnab Sarkar
Department of Computer Science & Engineering
Indian Institute of Technology, Guwahati

Lecture - 21
Microinstructions and Microprograms

Welcome to the 7th unit of the module we are discussing, that is on the control circuitry of the computer. So, this is 7th unit in which we are going to study about micro instructions and micro-program.

(Refer Slide Time: 00:42)



Units in the Module

- Instruction Cycle and Micro-operations
- Control Signals and Timing sequence
- Control Signals for Complete Instruction execution
- Handling Different Addressing Modes
- Handling Control Transfer Instructions
- Design of Hard-wired Controlled Control Unit
- Micro-instruction and Micro-program
- Organization of Micro-programmed Controlled Control Unit
- Different Internal CPU bus Organization

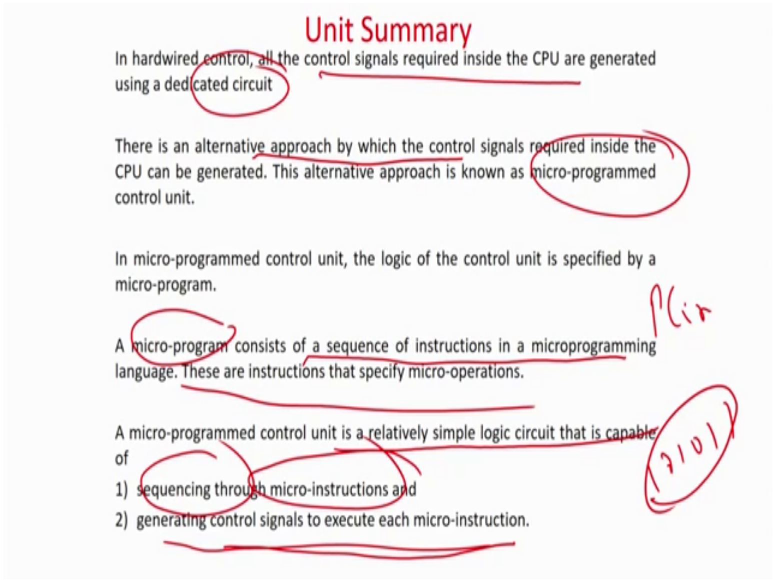
Basically, in the last unit we had seen that basically how to generate the control signals, if the hardwired for this is I exactly fabricated as a hard-coded non-modifiable circuit. Which you actually call as the hardwired control unit, that that is the sequence of micro instructions and the control signals to be generated corresponding to that can be generated using a hard-coded circuit, which is synthesized from it is finite state machine control.

So, that part we have seen that the circuit we generates out of this is a non-flexible circuit, but it is very fast, but it is non-flexible and it cannot be changed. In fact, for a given sequence of micro instructions and the control signals correspondingly we generate a final stage machine, and then we synthesis the circuit out of it.

In the next units basically, that is one micro-program control, but in these 2 units. So, basically, we are going to study how the same thing that is generation of the control signals can be d1 in a more flexible way and in a terms of a program. So, it is not it is very similar to what we understand by a normal computer program, which we have already looking throughout these lectures. But it instead of the macro instructions will be using micro instructions, and that control will actually called micro-program-based control.

So, this actually unit is focused on the generating of the control signals using a micro-program (Refer Time: 1:58) which is more flexible you can change it, but of course, it will be slower than a hardwired circuit.

(Refer Slide Time: 02:03)



Unit Summary

In hardwired control, all the control signals required inside the CPU are generated using a dedicated circuit

There is an alternative approach by which the control signals required inside the CPU can be generated. This alternative approach is known as micro-programmed control unit.

In micro-programmed control unit, the logic of the control unit is specified by a micro-program.

A micro-program consists of a sequence of instructions in a microprogramming language. These are instructions that specify micro-operations.

A micro-programmed control unit is a relatively simple logic circuit that is capable of

1) sequencing through micro-instructions and
2) generating control signals to execute each micro-instruction.

So, what is the unit summary what we are going to look into that. Basically, if the control signals are generated from a dedicated circuit we call it as a hardwired control. And alternative approach is basically which we can generate such signals which are basically programmed into some kind of a memory. So, it is we if it is a memory-based logic in which each of the memory cells or each of the memory word has the corresponding control signals to be generated, then that approach is actually called the micro-program control unit. Which is basically we are going to study in this unit.

So, basically a micro problem a consists of a sequence of instructions, and basically these instructions are nothing but which are the basically a micro-program is corresponding to basically sequence of micro operations that is very well known, that a macro instruction has some micro instructions which you have already discussed few lectures back. And corresponding each of the micro instructions or the micro operations basically we have some of the control signals, like for example, pc in should be equal to 1 etcetera. So, those bits we can actually make them 1 and whichever are not required we make them 0, and stored in one word of a micro-program control memory.

So, micro-program control memory is very similar to a normal memory, but we allocate it separately for the micro programming bits control. So, we call it as a micro-program memory. So, if we can put these control signals explicitly in some memory locations, then when you access that word of the memory such the corresponding control signals will become 1 and 0 as required.

So, that is basically the idea of a micro-program-based control. For each of the micro instructions and the control signals that has to be made 1 and 0 you ratherly write into 1 word of the memory. And then step by step you actually go from one location to another and so, forth. So, basically a micro-program control unit is a simple logical circuit. In fact, it is a memory with some peripheral circuits, which actually as will go through sequence of micro instructions. In fact, there is nothing but 0 1 0 1 1 these are some of the control bits, or which have to be need 0 and 1 based on the instruction micro instruction. They are written in a memory and whenever you access that memory those values will be given as directly output as control signals.

And therefore, you have to you have the go in sequence that is one job of that logic, and then basically sequence I will tell you what the sequencing and control signal generation. It has 2 parts basically that is this control signals has to be generated in sequence.

So, control signal generation of the control signal is very simple, that is just this values has to dump from the memory and it has to go to corresponding locations. So, in this case generating the signals is very straightforward, unlike in a hardwired based control. Where we have a finite state machine and the corresponding more or many machine logic has to be implemented here, it is very straightforward. The values are already in the memory and you dump it out.

Sequencing actually slightly tricky, which is somewhat very easy in finite state machine approach because, the area of the flow of states which can take care very easily, but in this case, sequencing is slightly tricky because, unless until unless otherwise specified you will go from step 1 to step 2 to step 3 that is the sequential memory locations. Micro-program memory locations like step 1 step 2 and step 3 and so, forth. But, whenever there is a jump instruction and you require also to check some flag conditions, which is simpler in finite state machine-based approach.

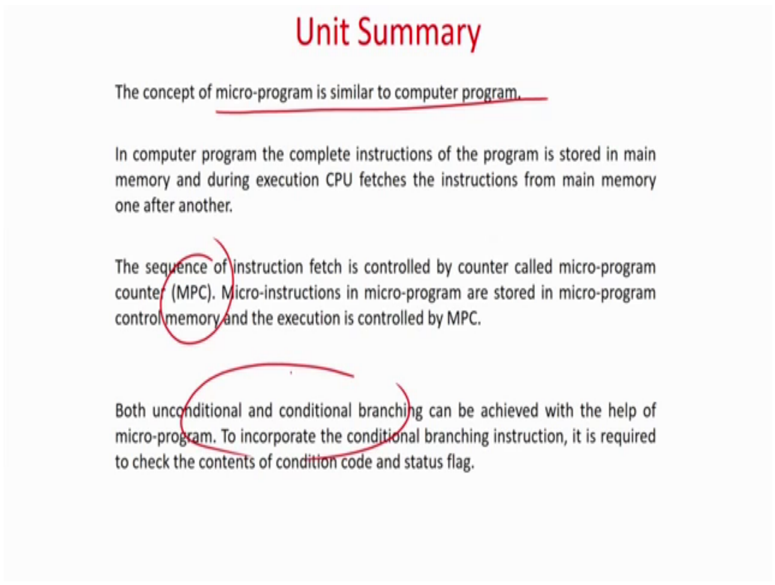
But here, we have to have separate arrangement. So, there just by looking at the control signals the that is the input output signals, then you have to also because the fags flag signals, then you decide that whether the next location is the location which has to generate the corresponding signals, or you have to jump to some other location, when the control signals has to be generated accordingly.

So, basically micro-program-based control is very simple, whatever signals you require you directly put them in the memory location. So, whenever you have to generate those control signals that memory word has to be fetched, but only sophistication is that you in normal case you will move through of step 1 step 2 step 3 like that in the memory control memory sequentially. But whenever you have to jump you have to go to some other corresponding memory location. So, accordingly the program counter we call it actually here micro-program control counter because, the architecture is very similar to that of a normal program execution.

So, whenever you have to jump to others. So, I more sophisticated program mechanisms as has to be used which will take the values from the input output signals as well as the flag signals and take decision accordingly. So, there is a slight difficulty compare to the hardwired control.

But, simplicity here in terms of if you compare to hardwired control is there generating signals is very simple. There is no circuit explicitly required values are stored in memory locations, which are directly faced than the values given that is what we are going to see in details.

(Refer Slide Time: 06:45)



Unit Summary

The concept of micro-program is similar to computer program.

In computer program the complete instructions of the program is stored in main memory and during execution CPU fetches the instructions from main memory one after another.

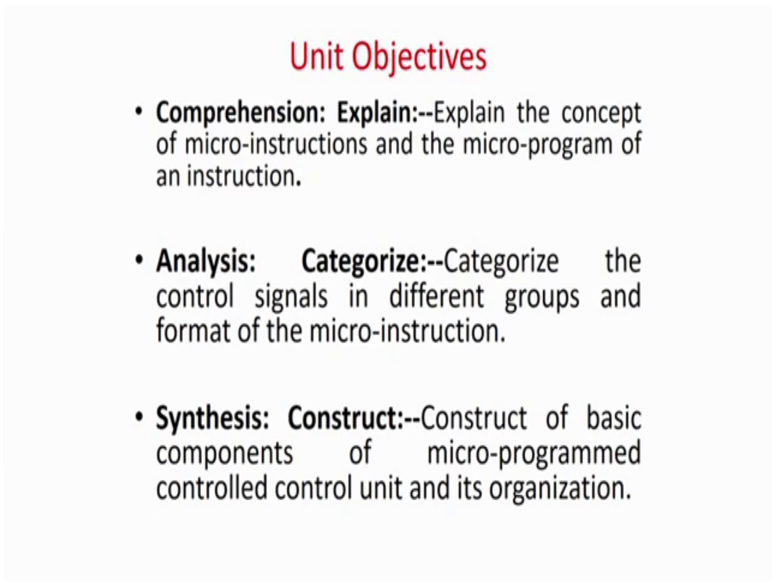
The sequence of instruction fetch is controlled by counter called micro-program counter (MPC). Micro-instructions in micro-program are stored in micro-program control memory and the execution is controlled by MPC.

Both unconditional and conditional branching can be achieved with the help of micro-program. To incorporate the conditional branching instruction, it is required to check the contents of condition code and status flag.

Basically, we can also say that micro-program is very simple to a computer program. In case of a computer program the whole macro instructions are stored in a memory, and you just actually fetch it one by another. And whenever you require a jump instruction, you jump that is actually controlled by the program counter. Here, also it is very simple very similar basically instead of a program counter we call it is a micro-program counter and also you have explicit memory which will actually STORE this control signals corresponding to each of the micro instruction, and there is actually called the micro-program micro-program memory.

So, there is slight architecture is more or less similar, but before the term because, the flow of instructions which happens in a normal macro program is very similar in a micro program, but we use the word micro-program to differentiate basically. Similarly, it has more conditional and unconditional branching. So, how these are very similar? So, that is what we are going to look into this look in this unit.

(Refer Slide Time: 07:37)



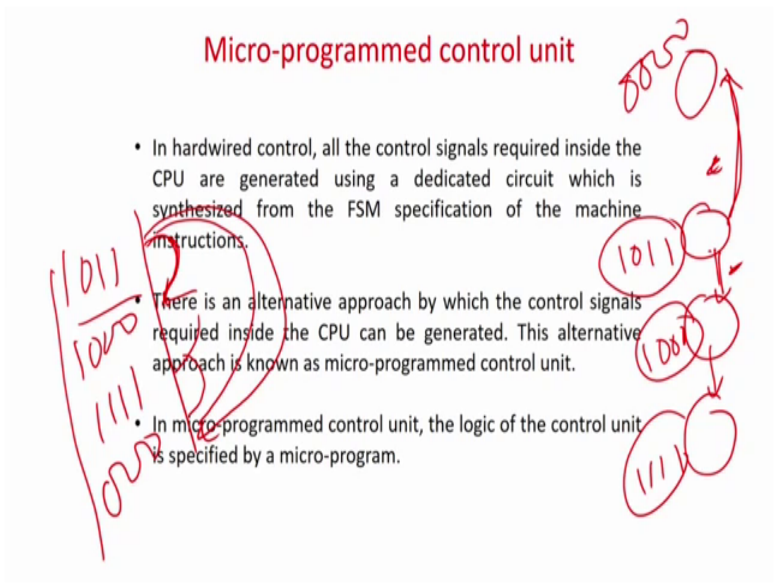
Unit Objectives

- **Comprehension: Explain:**--Explain the concept of micro-instructions and the micro-program of an instruction.
- **Analysis: Categorize:**--Categorize the control signals in different groups and format of the micro-instruction.
- **Synthesis: Construct:**--Construct of basic components of micro-programmed controlled control unit and its organization.

So, what are the basic objectives which you are going to feel free after doing this lecture. So, it is a first is a comprehensive objective in which case you will be able to explain the concept of micro instructions, and the micro-program of an instruction that is given an instruction. You will be able to explain that what are the micro instructions corresponding to that. In fact, we have already learned it in the fewer some classes back, but in this case also you will be able to translate it into a micro-program.

Then, next is an analysis objective you will be able to categorize the control signals in different groups, and the format of micro instructions. That is given some instructions macro instructions you will be able to generate the micro instructions and the micro programs out of it. Synthesis there is the synthesis objective construct construction of or you will be able to construct, basic comp1nts of a micro-program control unit and it is organization. That is, you will be able to synthesize whole micro-program control unit given a set of instructions. So, these are the basic objectives of this unit.

(Refer Slide Time: 08:34)


Micro-programmed control unit

- In hardwired control, all the control signals required inside the CPU are generated using a dedicated circuit which is synthesized from the FSM specification of the machine instructions.
- There is an alternative approach by which the control signals required inside the CPU can be generated. This alternative approach is known as micro-programmed control unit.
- In micro-programmed control unit, the logic of the control unit is specified by a micro-program.

Let us go into the unit in details. So, what is a micro-program control units?. So, how many of you have seen that, you know finite statement machine approach or the hardwired based approach basically, we have some states which are synthesized as a hardwired. So, you cannot change this and but it is quit fast that is the advantage, but problem is that for a given micro instruction sorry a macro instructions all sort of instructions this flow is very much fixed, that cannot be changed.

So, to have a more flexible approach we are actually go for something called a micro-program-based approach. So, let us assume that, each generate some signals called 1 0 1 1 may be the first signal is for program pc, in secondary for pc out, this may be for mdre. This is some control signal values may be this straight generates 1 0 1 1, next statement is 1 0 0 0. Next statement is all 1s some arbitrary values it is generating.

So, in this case you just move from state 1 to state 3 and may be on some other condition check you can very easily increment, it can go to some other state where we have all 0s. So, we can have some condition check. So, where or based on the inputs 1 the flat, but in case of micro control ah. So, in case of hardwired control, they are some it is a more over mini machine, there are hardwired circuits. Which will actually generate this values based on the state variables, or the state encoding process.

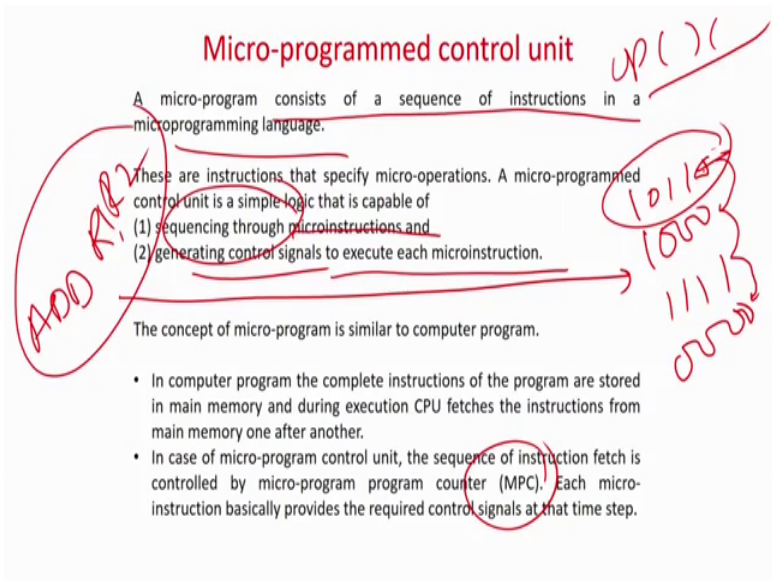
But, in case of micro-program we actually these values foR1 0 1 1 1 0 0 0 1 1 1 1 and all 0s is some memory location, that we are calling as the micro-program memory. Then we go from this step to this step this step to step where these steps to this step sequentially using a micro-program program counter.

So, if micro-program architecture is if you STORE these values in a micro-program memory, where it is very simple just to go for one memory location to another fetch the values, and automatically control signals are generated. But, what is difficult in a micro-program approach compare to ah hardwired approach is there. For example, going from here to here or here may be depending on some input values or a flag values simple because, it is a hardwired circuit which is implemented.

But, in this case it is a memory which actually goes here. So, either you go from here to here, or to go from this location to this location for that basically we should have some sophisticated arrangement. Which we will see which is not very straightforward, which we will have we have to have some more arrangement. So, that depending on some kind of input values you decide whether this is the next instruction, or this is the next instruction.

So, for micro-program control unit when you are implementing a branch kind of an instruction, or where conditions has to be check we have a slightly round around a arrangement. But, it is straightforward in finite state machine, but overall micro-program is very flexible because, you can easily change the code if you like based on the requirements. But, the sequencing is slightly tricky which is more difficulty in case of ah (Refer Time: 11:14) is a your micro-program jump instructions-based a based on some condition take it slightly sophisticated from (Refer Time: 11:22) finite state machine. So, basically that is what is a micro-program control unit.

(Refer Slide Time: 11:28)



Micro-programmed control unit

A micro-program consists of a sequence of instructions in a microprogramming language.

These are instructions that specify micro-operations. A micro-programmed control unit is a simple logic that is capable of
(1) sequencing through microinstructions and
(2) generating control signals to execute each microinstruction.

The concept of micro-program is similar to computer program.

- In computer program the complete instructions of the program are stored in main memory and during execution CPU fetches the instructions from main memory one after another.
- In case of micro-program control unit, the sequence of instruction fetch is controlled by micro-program program counter (MPC). Each micro-instruction basically provides the required control signals at that time step.

So, basically as I told you the micro program consist of a sequence of instructions in a micro programming language that is correspond to the micro instruction ah. Micro instruction there are some micro instructions and basically, you have to sequence through the micro instructions and generate control signals to execute. So, that is a 2 2 z1s that is say for example, like as I told you. So, these are this is signals which has to be generated. So, sequencing through the micro instruction is or generating the control signals means, you if you read this memory location. So, these values will be coming out from the memory micro-program memory. So, that is set the values accordingly.

So, generating a control signal for each micro instruction is simple just fetch the memory location. Sequencing means, I have generally sequence like this, but whenever there is jump instruction you have to follow or you have to move from one instruction to another based on which is non-sequential, based on some input variables then, somehow, I have to change the value of micro-program counter which will not be 1, but it will go to some other location.

So, in that case sequencing is slightly more sophisticated. Again, as I told you the micro-program concept is very similar to a normal program. So, the normal programs basically have some instruction which is phase decoded, it goes through an instruction register and it happens in that way, but same thing happens here, but yeah there like in that in a in a macro instruction, you have an opcode then you have operands and so, forth.

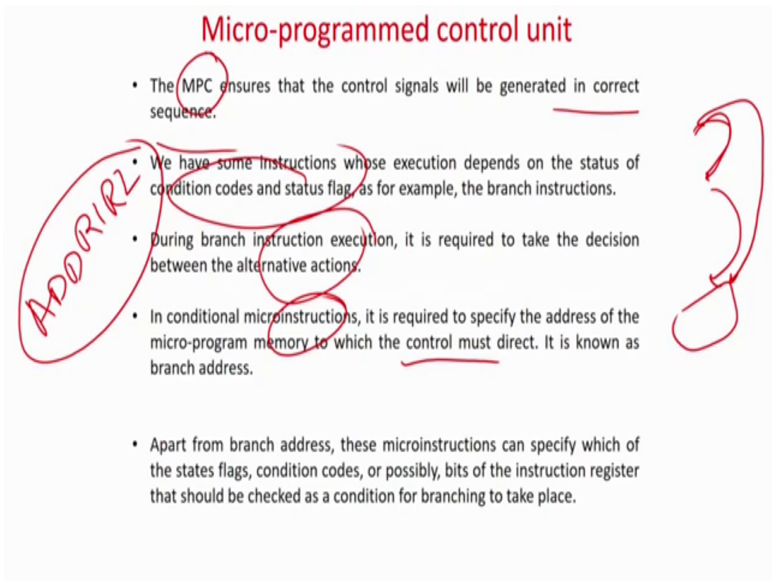
But here, there is no questions of any opcode and operand basically. So, what happens I will just tell you one example may be say, we have the instruction called ADD R1 R2. So, this is a macro instruction. So, in that case you will have the when it goes to the instruction register, you have the opcode and you have the operands.

But in case of a micro in case of micro is micro-program, basically you you have some bits and bytes so. In fact, these are also bits it into the in terms of bits. But here, we have opcode and operands and have separate meanings of different field. But, the in this case what happens generally the, this is one set of fields which will just correspond to the control signals directly. That is, you will directly feed, these 2 different control positions like program counter a ALU to be added etcetera.

Let me slight difference, but the control flow is very similar. You go for one institution to another, then to another if required you jump. But, basically also it is a program counter which will take care of this, but that concept remains same. But here, you did not decode likewise that it is corresponding to add or STORE or something like that directly control signals are generated in a micro-program.

But the few architecture is same. So, therefore, we call it there is a micro-program pc that is micro-program program counter basically, and also this memory we call it as a micro-program memory to slightly differentiate. Flow is same but, only the purpose is be different. So, in case of a normal macro program you have ADD, STORE, LOAD such that way instructions are executed you have an opcode and different operands. So, you have to use this opcode go to a address instruction decoder, and from there will be able to generate some kind of control signals. But, in case of micro-program the control signals are actually basically I am provided in a instructions itself because, micro instructions actually get translated into micro instructions, and they actually are encoded in terms of control signals.

(Refer Slide Time: 14:33)



Micro-programmed control unit

- The MPC ensures that the control signals will be generated in correct sequence.
- We have some instructions whose execution depends on the status of condition codes and status flag, as for example, the branch instructions.
- During branch instruction execution, it is required to take the decision between the alternative actions.
- In conditional microinstructions, it is required to specify the address of the micro-program memory to which the control must direct. It is known as branch address.
- Apart from branch address, these microinstructions can specify which of the states flags, condition codes, or possibly, bits of the instruction register that should be checked as a condition for branching to take place.

So, basically the micro-program control even ensures that this signal will be generated in correct sequence. So, sequence here is a bit tricky, because generating control signals is directly they are encoded. Directly they are actually put in the memory location. So, if you generate if you just take cell 1 memory 1 memory 2, memory 3 automatically control signals are generated as default because, they are already stored in the memory.

Sequencing is actually very important here because, many times we will depend on the condition codes and status flags. Based on somehow some storage that is some signal from the memory WMFC based on some case like some interrupt etcetera. Because of some case and also some status flag like 0 flag, carry flag so all, these things may take in to picture, and then you have to decide whether it will be the next phase, or it will go to some other instruction which is not consecutive.

So, you have to have also arrangement for something called a branch decision. So, in right normal programs we have micro instructions were actually which is conditional, and also, we have something which is unconditional. In conditional it is required to specify the address of the micro-program memory where the control must direct. So, means if it is some condition is not true we will go here, that is the next stage. But, if it is true you go from here to here, but then you have to actually tell what is the address. So, that part is quite obvious, but in this case, you have to tell the address of the micro-program memory of course, that is because that it is true because, you are executing in a micro-program memory

So, micro-program memory architecture and normal memory architecture there is not much different, there is almost the same thing. But, we are allocating some part of the micro-program memory when you have the micro programs, or the micro instructions corresponding to the macro instructions. We will take with example that will be more easy like for example, if you have some instructions called ADD R1 R2. So, first is first phase is called the fetcher instructions.

So, we have discuss many many time that for most of the instruction fetch phase is similar. So, whenever instruction has to be fetched this sequence of micro instructions is always similar. So, whenever instruction has to be fetched you can directly invoke that part of the memory, micro-program memory which has the micro instructions corresponding to fetch.

After that the add will be decoded, and then different types of activities has to take place. So, whenever ADD has been decoded, then it is a register to register operation, then the instruction or the micro instructions which is stored in the micro-program memory corresponding to ADD register to register that will be involved.

So, that series of micro instructions will execute, which will actually execute ADD which is from register to register operation, and then it was goes 4th. So, we will take some examples then it will be more clear for the time being restrict the fact that instruction fetch, decode and execute.

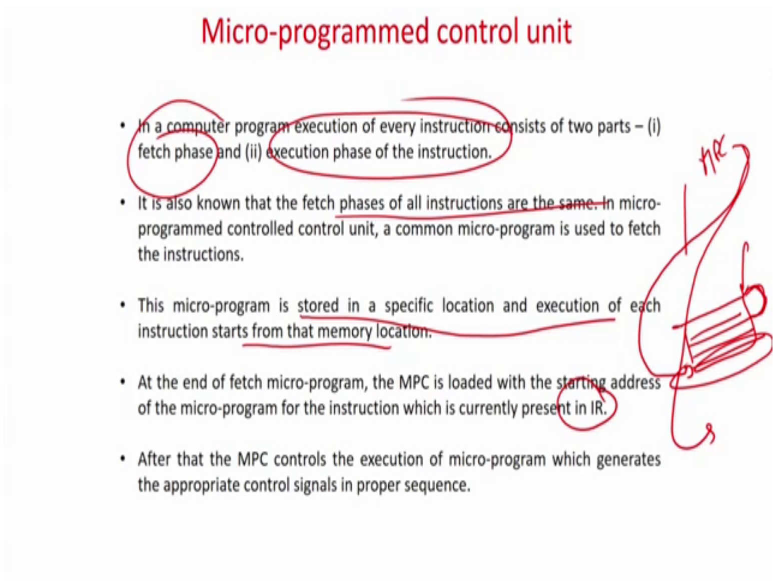
So, fetch everything is similar. So, whenever a new instruction has to be fetch ah. Fetched block of micro instruction which corresponds to it is memory phase, it will be invoked invoke means micro-program counter will start pointing to the first instruction microinstruction, which corresponding to corresponds to instruction fetch.

Then, when I instruction fetch is d1 then your decoding in the instruction register, and then accordingly it will tell the this is the ADD instruction which is from register R1 to R2 that is basically register to register instruction. So, corresponding microinstruction corresponding MPC will start point to the micro-program address or micro-program memory address. Where you have the micro-program corresponding to ADD R1 R2 then it will end again other steps will follow.

So, that is what is been actually is the java for micro-program control unit. So, apart from branch address these micro programs can specify which status flags conditions etcetera has to be taken for the condition check. So, that so; that means, you know very nutshell basically you just go micro-program counter increases one by one, but whenever there is a condition which depends on the input, depends on the flags, depends on some IO input. Which is which you get it from IO device there all these conditions has to be checked, and then the micro-program control sequence will tell the micro-program program counter that whether this is the next, or whether you have to go to some other memory location of the micro-program, and then you have to tell exactly which location to go.

So, that part has to be a part of the sequencing circuit logic for micro-program control ok.

(Refer Slide Time: 18:42)



Micro-programmed control unit

- In a computer program execution of every instruction consists of two parts – (i) fetch phase and (ii) execution phase of the instruction.
- It is also known that the fetch phases of all instructions are the same. In micro-programmed controlled control unit, a common micro-program is used to fetch the instructions.
- This micro-program is stored in a specific location and execution of each instruction starts from that memory location.
- At the end of fetch micro-program, the MPC is loaded with the starting address of the micro-program for the instruction which is currently present in IR.
- After that the MPC controls the execution of micro-program which generates the appropriate control signals in proper sequence.

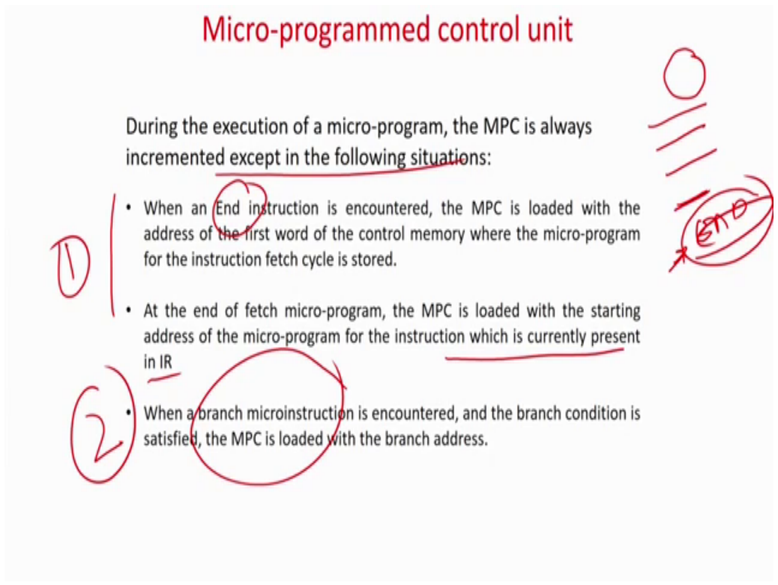
So, what I was saying you can just read in this slide. Basically, we have every instruction has 2 parts called fetch and execution phase. Execution phase means, here I am telling that decode do the operation and STORE. So, basically all for all instruction phase fetch phases is similar.

So, whenever a new instruction has to be executed first actually micro-program corresponding to fetch will be executed. So, the micro-program is stored in a specific location in the micro-program, and execution starts from the memory location maybe we have a this is a micro-program, maybe these part is reserved for the instruction micro instructions corresponding to fetch. So, whenever a new instruction has come your MPC will point to this. So, whenever this instruction is completed that is 3 or 4 micro instructions to for fetch is completed, then again, the MPC will start pointing to some xyz position. So, it will come here and it will end and it will now the floating.

Now, the instruction decoder basically that is the instruction decoder we will have the instruction like ADD R1 R2 stored or something, then it will decode. And find out what is the instruction it corresponds to maybe it is a LOAD instruction, then again accordingly the micro-program counter will start pointing to that location in the micro-program. Which is the first instruction micro instructions correspond to that particular instruction? It is ADD R1 and R2 so, this will correspond to the micro instruction starts where the micro instruction starts which corresponds to ADD register to register.

If the STORE. So, it may be pointing to some other location in the micro-program which corresponds to the start address of the micro-program which corresponds to LOAD or STORE. So, basically this is how it actually happens that is for any instruction. The first phase is similar that is the fetch, and then after the based on the instruction type which is decided by the instruction register, it will take the MPC value to corresponding position in the micro-program memory where the micro-program correspond to the instruction sits.

(Refer Slide Time: 20:42)



Micro-programmed control unit

During the execution of a micro-program, the MPC is always incremented except in the following situations:

- When an End instruction is encountered, the MPC is loaded with the address of the first word of the control memory where the micro-program for the instruction fetch cycle is stored.
- At the end of fetch micro-program, the MPC is loaded with the starting address of the micro-program for the instruction which is currently present in IR
- When a branch microinstruction is encountered, and the branch condition is satisfied, the MPC is loaded with the branch address.

So, now, if you look at this slide which actually tells in more formal manner with you I say that during the ignition of a micro-program control, the MPC always incremented. So, it go accepting the branch condition is that this is very obvious that, whenever there is a branch it will go to other place or 2 other cases basically it does not always increment, when an end instruction is encountered. So, what is the end instruction? Basically, say for example, as I told you fetch is the before it means, pf sequence of a before sequence that is to be executed. Whenever any instruction has to be taken from the memory and execute.

So, whenever instruction is their which has to be executed basically it pass might the micro-program corresponding to fetch is MPC points to that part and whenever the fetch instruction set is micro instructions are over or you go to an end of a micro-program then actually in then it does not go ahead; obviously, because maybe these 3 micro instructions are required for fetch. So, after that these part actually ends for that micro instruction

So, whenever it ends then it has to again wait whenever which is see the end instruction it does not go that is obvious basically if you take an instruction like ADD R1 and ADD R2. So, that is the macro instruction it will have many small small micro instructions sits program for there like fetch will comprise foR1 micro-program then R1 and R2 after decoded it is executed that is R1 and R2 had to be executed and it has to be written in R1

So, basically that part will comprised of the second micro instruction program. So, basically when the fetch is over it will go to the n instruction of the micro-program and of course, pc will be MPC will not increment. It will be blank, then basically that MPC will now, start pointing to the next instruction basically which corresponds to that macro program, right? For example, if ADD R1 and R2 it will start pointing.

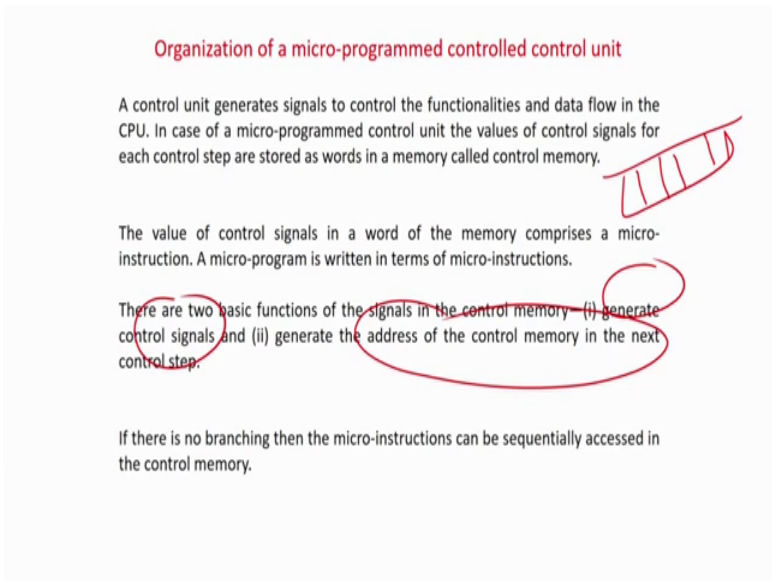
So, whenever it reaches the end of one micro instruct instruction program. So, it will again MPC will not be incremented, it will be waiting for the corresponding opcode that is the instruction register will tell basically, where it has to point is corresponds to the particular instruction. So, at that time is not incremented it waits basically when there is a jump instruction of course, the condition will tell you that where you have to jump.

So, at the end of the program the MPC that will be the starting Address of the micro-program for the instruction which is presently in the IR. So, basically these are the 2 cases this is case 1 and this is this is case 2. When we do not increment for all other cases basically we increment the micro-program program counter. One condition is branch that is obvious, and another condition is basically for a given instruction. There may for a given instruction is implemented by 2 or multiple micro programs.

So, whenever micro-program is over then, you have to wait for the other and that one will be told that one micro-program phase is over which is the next micro-program to be executed that will be told by the instruction register, like if we ADD R1 R2 the execution will start from a different part if this LOAD R1 R2, then it will be a different way. So, based on different instructions basically what are the execution part that is going to tell you which micro-program has to be executed for that.

So, in these 2 cases basically the micro-program, program counter has to be loaded one by the branch case and one by the instruction register.

(Refer Slide Time: 23:58)



Organization of a micro-programmed controlled control unit

A control unit generates signals to control the functionalities and data flow in the CPU. In case of a micro-programmed control unit the values of control signals for each control step are stored as words in a memory called control memory.

The value of control signals in a word of the memory comprises a micro-instruction. A micro-program is written in terms of micro-instructions.

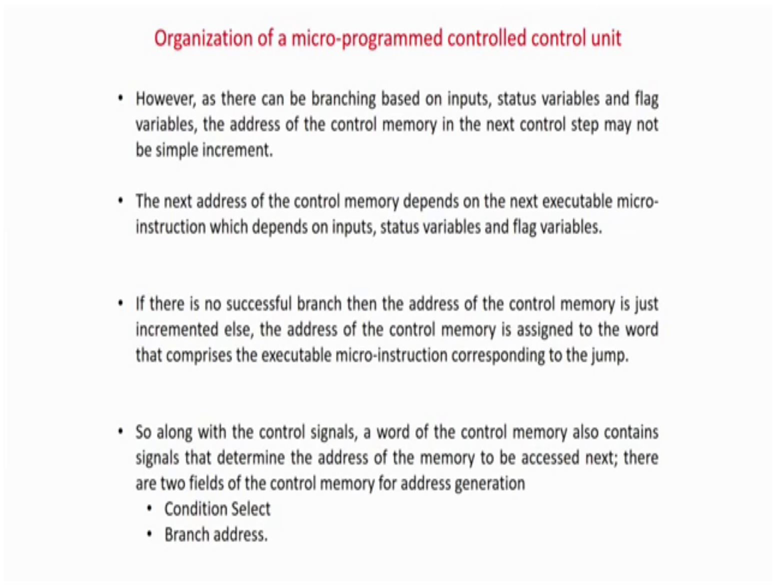
There are two basic functions of the signals in the control memory – (i) generate control signals and (ii) generate the address of the control memory in the next control step.

If there is no branching then the micro-instructions can be sequentially accessed in the control memory.

So, basically this is what in a nutshell we have seen micro-program for a given program with macro code micro. Macro code there is a sequence of micro programs like for example, if I take a macro instruction like ADD R1 R2. It will have corresponding defined micro programs which will execute it, and the micro programs actually execute exists in different parts of the micro-program memory.

So, basically what we our job is to do you have to actually generate this control signals, and you have to generate the Address of the control memory in the next, that we have to do the sequencing that is what are the jobs. So now, we will see how basically micro-program is organized which has to do these things one is the control signal generation and second is the how we can do the proper sequencing in this case again repeating control signal generation is very simple. Because, we have a memory and in all the memory actually memory bits you STORE the required signals which is to be made 0 and 1.

(Refer Slide Time: 24:54)

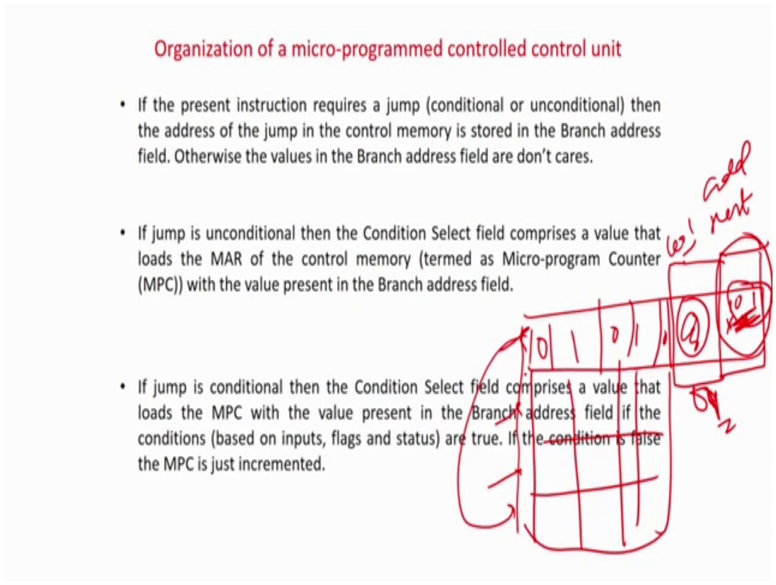


Organization of a micro-programmed controlled control unit

- However, as there can be branching based on inputs, status variables and flag variables, the address of the control memory in the next control step may not be simple increment.
- The next address of the control memory depends on the next executable micro-instruction which depends on inputs, status variables and flag variables.
- If there is no successful branch then the address of the control memory is just incremented else, the address of the control memory is assigned to the word that comprises the executable micro-instruction corresponding to the jump.
- So along with the control signals, a word of the control memory also contains signals that determine the address of the memory to be accessed next; there are two fields of the control memory for address generation
  - Condition Select
  - Branch address.

So, basically, whatever I was telling you can read in this in this slide. Basically, if there is branch then there are 2 cases in which you have you cannot go sequentially when one micro-program ends and in what case there is a branch instruction. So, in this case basically you have to think of the, which actually makes Address to go otherwise, MPC is just incremented.

(Refer Slide Time: 25:14)



Organization of a micro-programmed controlled control unit

- If the present instruction requires a jump (conditional or unconditional) then the address of the jump in the control memory is stored in the Branch address field. Otherwise the values in the Branch address field are don't cares.
- If jump is unconditional then the Condition Select field comprises a value that loads the MAR of the control memory (termed as Micro-program Counter (MPC)) with the value present in the Branch address field.
- If jump is conditional then the Condition Select field comprises a value that loads the MPC with the value present in the Branch address field if the conditions (based on inputs, flags and status) are true. If the condition is false the MPC is just incremented.

So, basically now, if you think basically this will be your program micro program. So, basically the micro-program is nothing but they are all some parts of a memory. So, I am feeling this value 0 1 1 0 something like this has been filled. Now, I have to reserve some part here and some part I will tell you why these things are to be reserved, because otherwise it will go in sequence. It will keep on going in sequence one after the other first this location this location, this location and so on [vocalized-noise.]

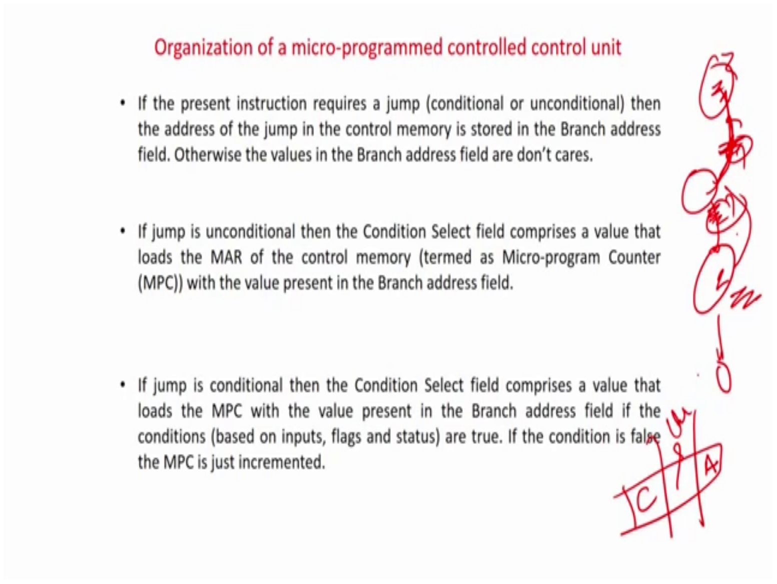
Now, somehow say that if I want to say that I want to go from this memory location to this memory location. That value that this this is the present state next that I have to go not this, but this some other memory location. So, another some part of the memory here will be telling that what is the next place that is the Address. That is the Address of the next location which the micro-program has to go, if some condition is satisfied. So, another part you have to keep for some conditional variable. That is based on some conditional variables the may be these are variables which has to be checked means, I am just filling in the basic gist in a few slides we will see the more concrete the how this is to be designed.

So, basically this one will go in this manner, but 2 and these are basically nothing but you control signals 0 1 1 0 1 1. Some part of the memory I will reserve at the next Address register, if if if I say that it will be xx or 0 0 or something then, it will say that I have to defined go to the next instruction. But, if I say that some values are there may be 1 0 1 1 some decode coding is there which will tell maybe 1 0 1; that means, it will tell that the next instruction is not this 1, but it has to go to 1 0 1. If some condition based on this memory location holds maybe if you say that 0 0. It tells that no condition has to be checked it will directly go to 1 that is default jump to memory location 1 1.

But, sometimes you may say that 0 1 and 1 0 1 and this 0 1 may correspond to saying that basically the carry flag should be 1. So, if the carry flag is 1 then you jump to 1 0 1 else you increment. So, the, this is the way micro-program memory will look like. You will have one part which is the most important part of it that is the control signal. Some part you have to reserve for the conditions which you have to change we can say 0 0 or x x; that means, no condition has to be checked, and you can also say xx; that means, no condition has to be check nothing you just the increment.

But, if I put some values over here it may mean that that condition if it holds then only this condition this is the newer location we have to jump, or you have if the, it does not hold you again go over here. So, basically apart from this control you should also have to STORE, where should be the next memory location and what should be the condition?

(Refer Slide Time: 27:58)



Organization of a micro-programmed controlled control unit

- If the present instruction requires a jump (conditional or unconditional) then the address of the jump in the control memory is stored in the Branch address field. Otherwise the values in the Branch address field are don't cares.
- If jump is unconditional then the Condition Select field comprises a value that loads the MAR of the control memory (termed as Micro-program Counter (MPC)) with the value present in the Branch address field.
- If jump is conditional then the Condition Select field comprises a value that loads the MPC with the value present in the Branch address field if the conditions (based on inputs, flags and status) are true. If the condition is false the MPC is just incremented.

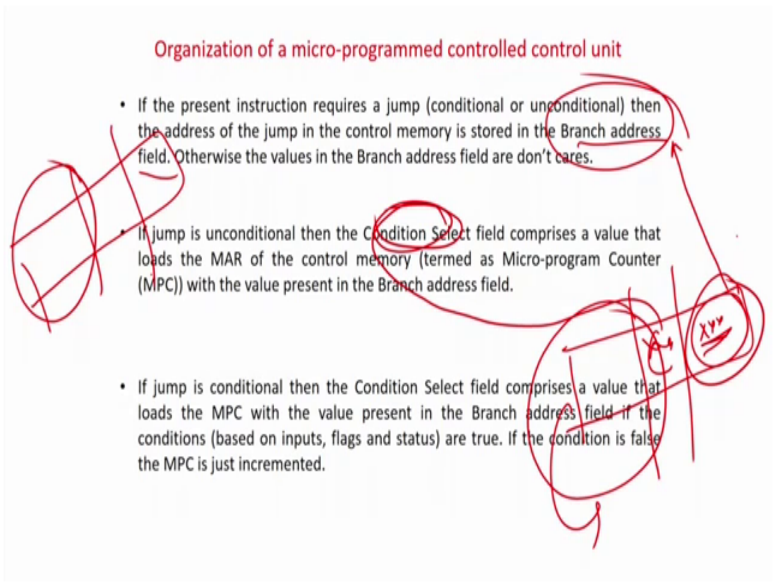
So, as this is not the finite state machine. So, in case of a finite state machine basically you can have some conditions, and the sequencing actually is maintained like. If this is the case you go over and if this is the case you go over.

So, this condition checks are put in these transitions and this destination. And these destinations are denoted by these states. So, here you do not have any kind of straight structure. So, one part of the memory we reserved for checking the conditions, and another part of the memory we say that do you want to go over here, or do you want to go over here.

So, that basically you want go over here means default the next, but this may not be the next instruction. So, to go to that state, location of these also has to be stored in the memory. So, to match this condition and the jump Address to this we have basically the memory has 3 parts this is for your control signals, this is maybe for your and next Address and this for your check of the condition. So, the memory has to be basically divided.

So, that is if the present instruction requires a jump, then the Address of the jump in the memory location is stored in the branch Address field that is not as sign.

(Refer Slide Time: 29:08)



Organization of a micro-programmed controlled control unit

- If the present instruction requires a jump (conditional or unconditional) then the address of the jump in the control memory is stored in the Branch address field. Otherwise the values in the Branch address field are don't cares.
- If jump is unconditional then the Condition Select field comprises a value that loads the MAR of the control memory (termed as Micro-program Counter (MPC)) with the value present in the Branch address field.
- If jump is conditional then the Condition Select field comprises a value that loads the MPC with the value present in the Branch address field if the conditions (based on inputs, flags and status) are true. If the condition is false the MPC is just incremented.

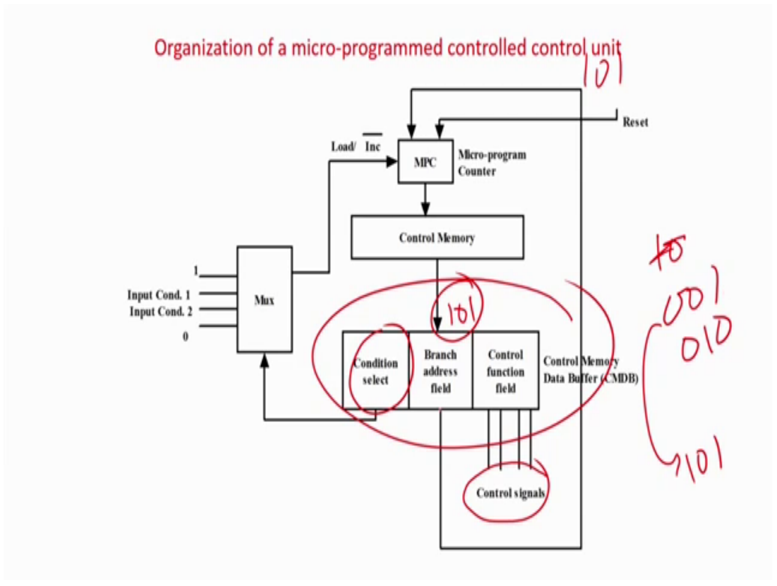
Basically, that means this is the whole instruction. So, if you have to jump. So, this is going to be your Address where it has to jump, but that is at this part is actually called the branch Address field.

So, therefore, you have to explicitly mention in the instruction itself, that if some condition is true you have to go to this Address. If the condition is not required or you need not jump from the instruction by any means you can put it is up default variable. If if the jump unconditional this is actually called the condition select field this is actually call the condition select field of course, because the jump acceptor we will happen based on some condition. So, as I told you can have different encodings that if the control has to be on some flags, or some very inputs from the IO devices that you can mention by some code over here.

So, if it is unconditional jump then, you can make c equal to 0 0 some coding you can give then it will not check anything, but directly jump to from this instruction similarly, if you do not want to have any jump in this case. So, you can have some these are the xxx and this will be some some arbitrarily encoding like 0 0 0 and 0 0 0. In such case it will directly go to this one without having to think about it. So, basically this micro-program actually has one part of the Address, which is the, which one part of the memory, which is actually called or in a row basically. In a memory word one part will generating a control signal one part will be basically for the control select. That is the depending on what you have to take decision and 1 will be actually the branch Address field. So, basically there are 3 parts.

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 will 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 program counter 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 1 0 1.

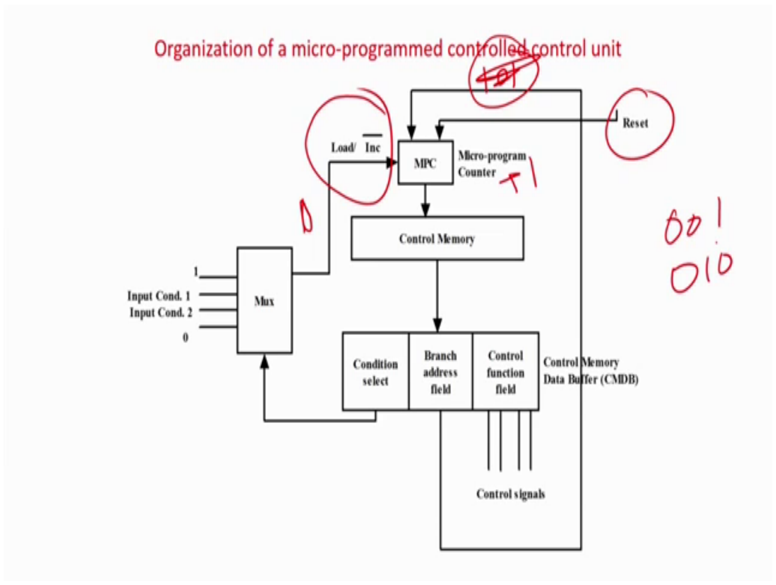
So, this is going to say that I am not going maybe a this present and, in the instruction, called say 0 0 1. So, I will go to 1 0 1, if some condition is true. That is what this thing and where I have to go, I have to go to 1 0 1. So, here we what telling which are the signals, here, we what telling which is the instructionly location you have to go, and you have saying that what is basically the condition it has to be meet.

Now, how can I tell that I have to go to this branch? This is very simple 1 0 1 means you have to update the daily of micro-program counter micro-program pc to 1 0 1. Otherwise it will become from 1 1 0 0 0 0 1 to it will 0 1 0 it will increment, but you are telling no I do not want to go to 0 1 0 whether I want to go to 1 0 1, 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 up 1 0 1, that is what is going to detail. So, this jump Address is basically you are feeding as one input to the micro-program counter.

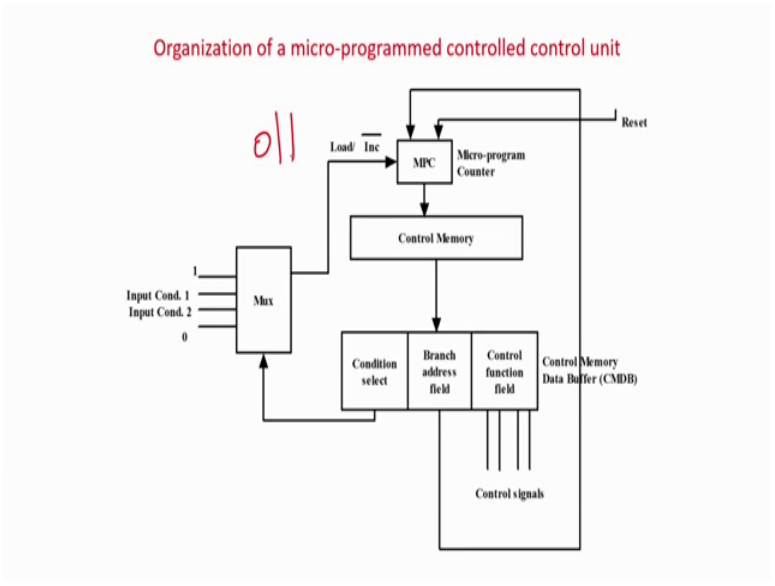
So, now, let us look at how the micro-program is designed one is the reset field; obviously, if you want to start from some 0 location you can reset it, and this is what is micro-program another jump address. Basically, this input corresponds to the jump address which will LOAD the micro-program counter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

(Refer Slide Time: 32:32)



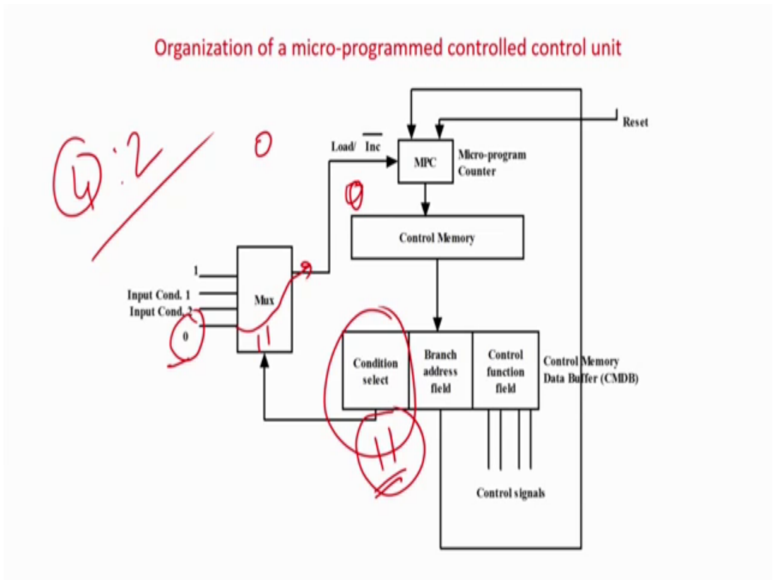
But, here you see this very interesting there is a control to MPC. So, what is the control to MPC one is LOAD and increment bar; that means, if this line is 1. So, what is this going to do it is going to LOAD the value which is available at this port so; that means, if this is 1 you are going to take the value of 1 0 1 in the micro-program control and you are going to jump to 1 0, but in this line is 0 then actually what is going to happen it is just going to make it as plus 1. So, basically it was 0 0 1. So, if this line is 0 you will go to 0 1, and you will be incrementing. So, the logic is very interesting. So, how implement it 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 d1.

(Refer Slide Time: 33:29)



How to decide that whether this line will be 0 and 1, 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)

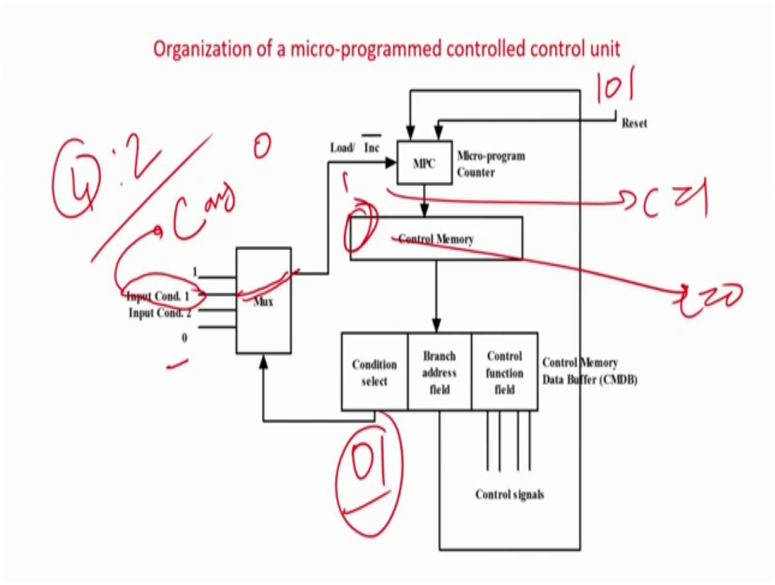


So, they actually say that say for example, I have only 4 condition codes. So, you have a 4 is 2 multiplexer as simple as that. If you have many conditions to be checked that is going to increase. So, I say that there is actually only 4 condition which has to check. And here I say that if I put 1 1 sorry if I put the value of code 1 1. So, if I put 1 1 in the multiplexer then of course, this line is going to be connected to this. This I am making our default 0; that means, if the condition select is 1 1 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 1 1, then the last part may be the last bit of the multiplexer is connected to the MPC control and this output is to 0.

So, if I give the condition has 1 1 you are always giving to get 0 and the micro-program will be just increment ; that means, if I give the condition code has 1 1, 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 see for example, I want to check if the I want to go to 1 0 1 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 0 1. So, if it is 0 1 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 which is there will be depend on the condition that which is connected to this.

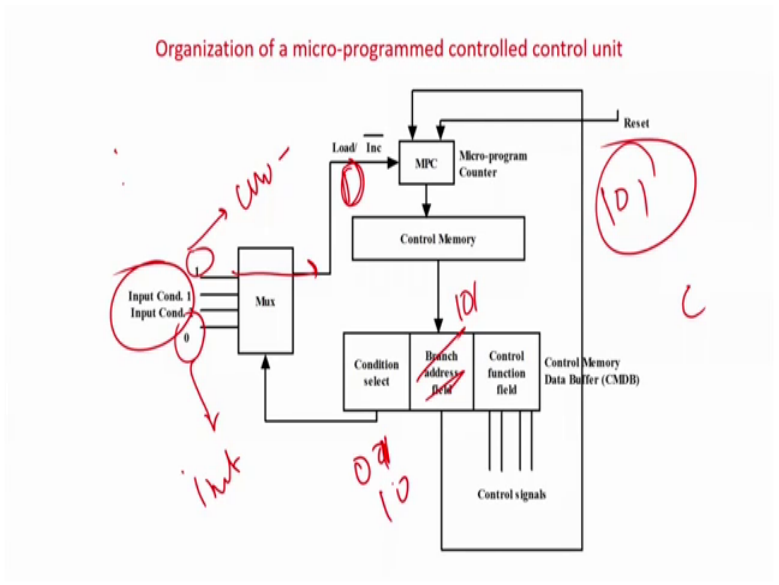
(Refer Slide Time: 35:02)



So, we have a connected carry 0. So, if I put the 0 1 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 0 1 and if the carry bit is a 1. So, you are going to LOAD 1 0 1. I put a 0 1 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 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 0 0 which is the default condition in which is the jump unconditional. So, if I put 0 0 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 1 0 1, which is specified over here. So, this is how actually it is implemented.

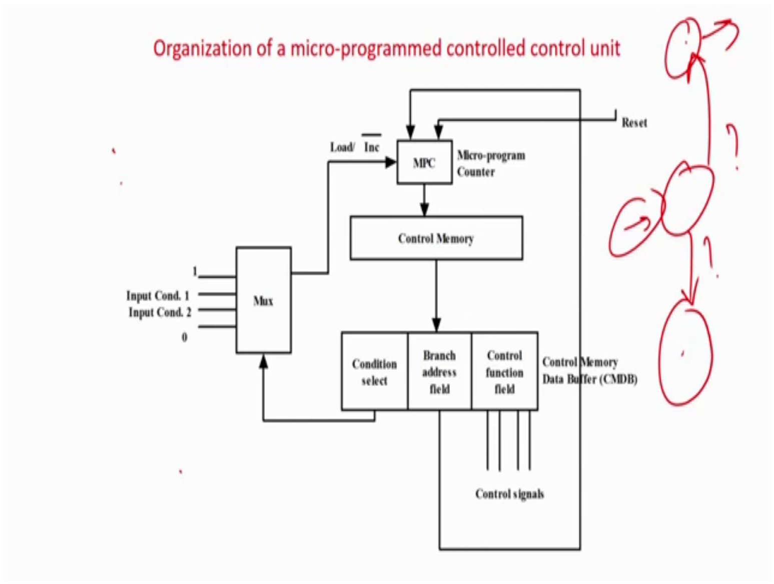
So, nicely looking at it you will find the interesting stuff over here. So, you are going to see that first bit is a 1. So, if I put 0 0 is 1 is going to be 1 and whatever is the value being Address field will come over here, and you are going to jump to that instruction. Last bit I am always explicitly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

So; that means, if I put a 1 1 this branch Address basically has no meaning, and if I put ah 1 1 then no condition has if I put 1 1 over here, the branch Address has no many because it is just going to be incremented but, if I say that if I put 0 0 over here. So, in that case it is a jump is very, very important here actually you are going to take 1 0 1 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 1 0 1. So, this 1 corresponds to basically unconditional jump, this 1 corresponds to just increment. So, in in case of increment this has no no value.

But for the 2 other conditions like 0 0 sorry 0 1 and 1 0 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, then, if I put 0 1 then you will jump to the instruction only there is a can, and if I put 1 0 as the control (Refer Time: 38:09) then you will tell that if and only if parity bit is set, then I will jump to instruction 1 0 1.

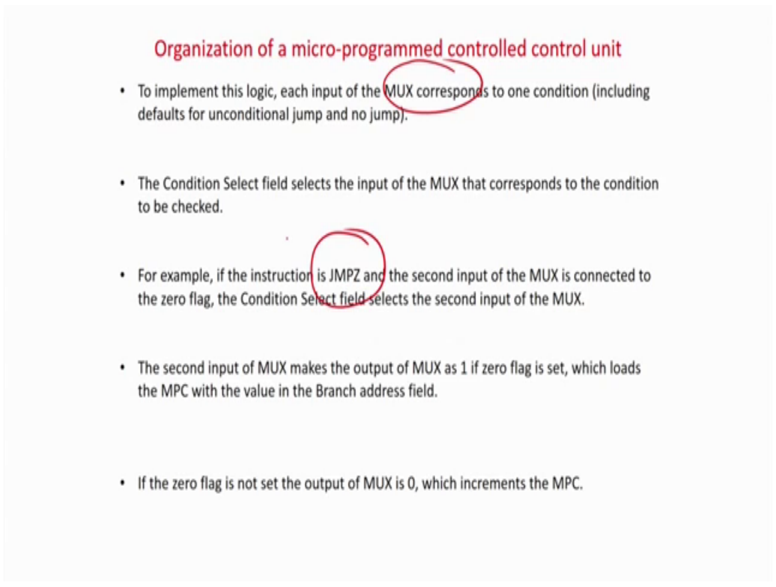
So, therefore, we have seen how this circuit basically implements the sequencing using a memory, because we have to remember that un unlike if I said we do not have some flexibility like this, where some conditi1d checks are here and you go from this state to this state based on some condition, and control from going to here or going to here is d1 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-program control unit.

(Refer Slide Time: 38:45)



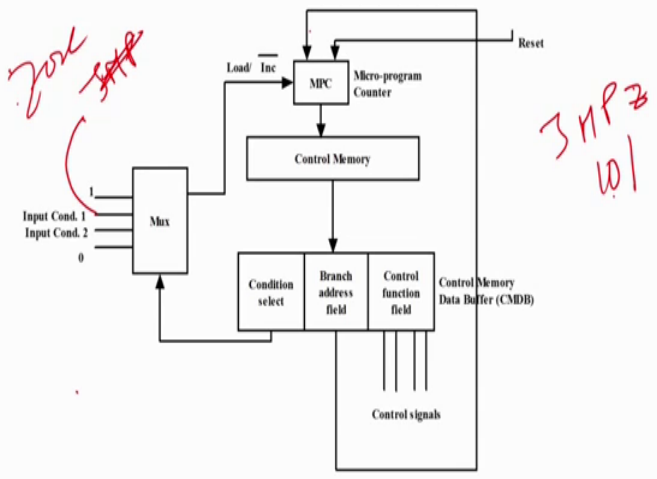
Organization of a micro-programmed controlled control unit

- To implement this logic, each input of the MUX corresponds to one condition (including defaults for unconditional jump and no jump).
- The Condition Select field selects the input of the MUX that corresponds to the condition to be checked.
- For example, if the instruction is JMPZ and the second input of the MUX is connected to the zero flag, the Condition Select field selects the second input of the MUX.
- The second input of MUX makes the output of MUX as 1 if zero flag is set, which loads the MPC with the value in the Branch address field.
- If the zero flag is not set the output of MUX is 0, which increments the MPC.

So, basically whatever I have told you is actually written in the slide. So, to implement this logic we actually this is 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.

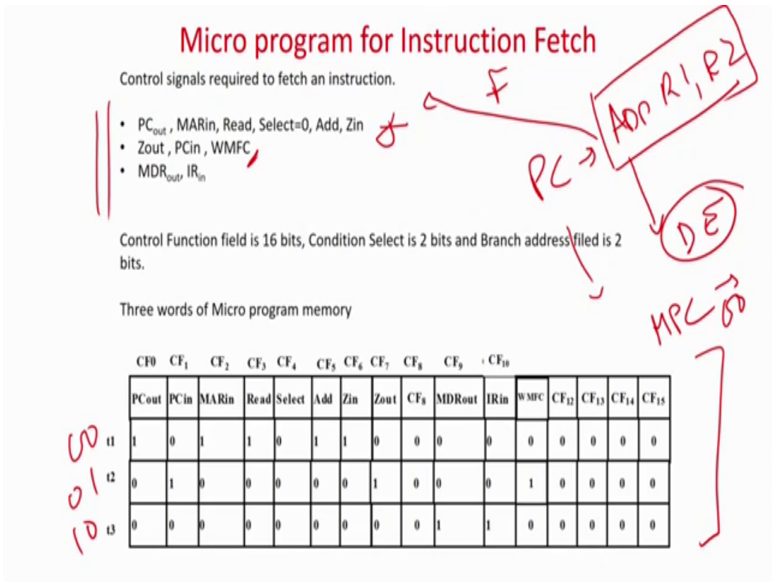
(Refer Slide Time: 39:09)

Organization of a micro-programmed controlled control unit



So; that means, if you have to ah. So, if you if you have some instruction like jump on 0 to say 1 0 1. 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, the how it is d1 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)


F
ADD R1, R2
PC →
DE
MPC → 00

Micro program for Instruction Fetch

Control signals required to fetch an instruction.

- PC_{out} , MARin, Read, Select=0, Add, Zin
- Zout , PCin , WMFC
- MDR_{out}, IR_{in}

Control Function field is 16 bits, Condition Select is 2 bits and Branch address filed is 2 bits.

Three words of Micro program memory

	CF0	CF_1	CF_2	CF_3	CF_4	CF_5	CF_6	CF_7	CF_8	CF_9	CF_{10}					
	PCout	PCin	MARin	Read	Select	Add	Zin	Zout	CF_8	MDRout	IRin	WMFC	CF_{12}	CF_{13}	CF_{14}	CF_{15}
00 t1	1	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0
01 t2	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0
10 t3	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Now, example so much theory you have told and you 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 discussed. 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 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 selects 0 ADD and Zin that is we are incrementing the value of pc, then the incremented value of pc by the where we are all as we have 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 plus constraint 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 in and for almost all instructions these are the 3 micro instructions requited to fetch it. So, let us see how it. So, whenever you have to execute this instruction automatically same example 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 0 0 0 1 and 1 0.

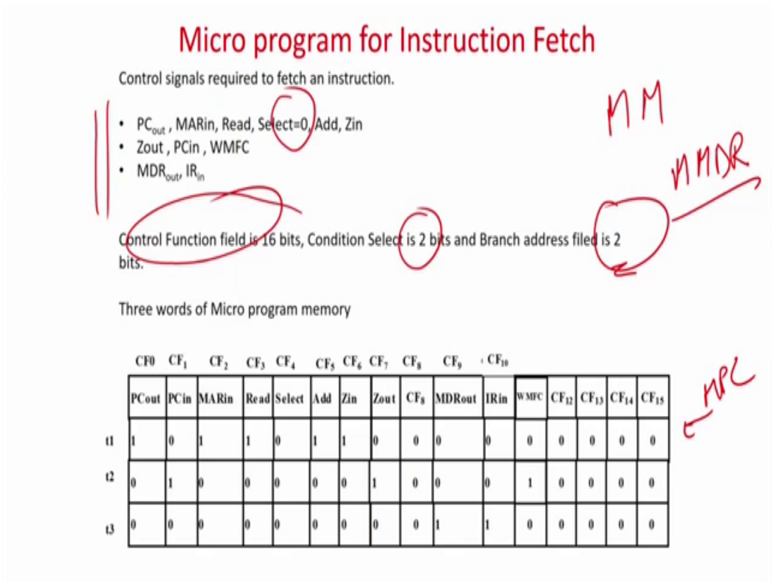
So, it will immediately make MPC equal to 0 0, which corresponds through the starting micro instruction for fetch that is this 1. So, whenever any instruction has to be executed it will make the MPC 0 0; that means, here we are assuming that the micro-program which corresponds to fetch off any instruction basically exists in this micro instruction memory micro instruction memory 0 0.

So, that is default. So, after this is fetching is d1 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 backend because, as I told you Address bits 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 PCin PCout, MARin. 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 they 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 they are can be 2 bits; that means, they are assuming that control will can depends only your 2 bits, that is 2 is to 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 Zin should be 1, that these are the different points you have to set it. So, it is very easy here unlike a fsn. For fsn you have to design a circuit and synthesizing here, you just write the first bit corresponds to pc. Second bits corresponds to PCin the first corresponds to PCout this corresponds to PCin this corresponds to MARin read that means.

(Refer Slide Time: 43:16)



Micro program for Instruction Fetch

Control signals required to fetch an instruction.

- PC_{out} , MARin, Read, Select=0, Add, Zin
- Zout , PCin , WMFC
- MDR_{out}, IR_{in}

Control Function field is 16 bits, Condition Select is 2 bits and Branch address filed is 2 bits.

Three words of Micro program memory

	$CF0$	CF_1	CF_2	CF_3	CF_4	CF_5	CF_6	CF_7	CF_8	CF_9	CF_{10}					
	PCout	PCin	MARin	Read	Select	Add	Zin	Zout	CF_8	MDRout	IRin	WMFC	CF_{12}	CF_{13}	CF_{14}	CF_{15}
t1	1	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0
t2	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0
t3	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

What this memory location will be read from the micro-program memory it will go to this is 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 connected to the read port of the main memory and so, forth.

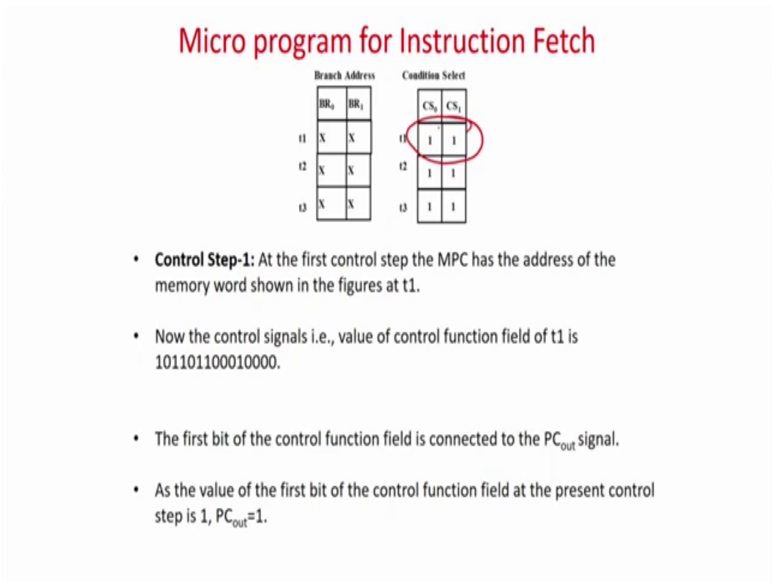
So, just this will be coming out of the main micro-program memory data bus, that is data is a the buffer register. And these points of the Address bus is directly connected to the corresponding positions in the hardwired block. And so it is very simple at the memory data is coming from the memory and it is directly connected to the different points and the job is that is d1.

So, in the first thing this is cycle you know know pc 1 equal to 1. So, I will store a 1 over here PCout is 0 all other signals which are not menti1d over here are basically 0 MARin. 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, Zin 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 Zout equal to 0. This correspond to some other 0, MDRout is 0, IR in is 0, WMFC is 0 and all other values you have to put it 0.

So, only whatever are menti1d 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 menti1d over you put 1 and all others has to be reset to 0 that is d1.

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 PCout 1 pc this is go to PCout 1 pc in PCout is going to be 1, sorry PCin is will be 0 are not to anything with it mar will be 1. So, all the connection should be d1 the values will be open d1 and your job is d1.

(Refer Slide Time: 45:17)



Micro program for Instruction Fetch

Branch Address

	BR_0	BR_1
t1	X	X
t2	X	X
t3	X	X

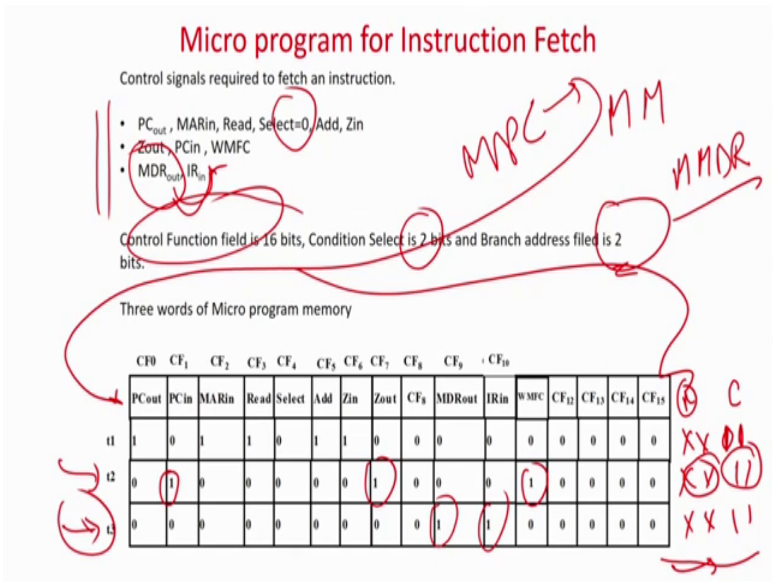
Condition Select

	CS_0	CS_1
t1	1	1
t2	1	1
t3	1	1

- **Control Step-1:** At the first control step the MPC has the address of the memory word shown in the figures at t1.
- Now the control signals i.e., value of control function field of t1 is 101101100010000.
- The first bit of the control function field is connected to the PC_{out} signal.
- As the value of the first bit of the control function field at the present control step is 1, PC_{out}=1.

Next because, in this case there is no question then I will show you. This is actually corresponding to the branch and control select. So, in this case first I am putting at 1 1. So, if you look at what was 1 1 1 1 is basically corresponding to this bit; that means, basically it is going to give a 0 over here. So, it is just increment it. So, if you look at it I am putting this one and branch Address is 0 0; that means, basically I can write it over here for your ease. So, here I am doing it as a xx and control bit is 1 1.

(Refer Slide Time: 45:44)



Micro program for Instruction Fetch

Control signals required to fetch an instruction.

- PC_{out} , MARin, Read, Select=0, Add, Zin
- Zout , PCin , WMFC
- MDR_{out} , IR_{in}

Control Function field is 16 bits, Condition Select is 2 bits and Branch address filed is 2 bits.

Three words of Micro program memory

	CF0	CF1	CF2	CF3	CF4	CF5	CF6	CF7	CF8	CF9	CF10					
	PCout	PCin	MARin	Read	Select	Add	Zin	Zout	CF_8	MDRout	IRin	WMFC	CF_{12}	CF_{13}	CF_{14}	CF_{15}
t1	1	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0
t2	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0
t3	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

So, control again I write it that is (Refer Time: 45:49) us ok. So, if when I am set that is this control is basically 1 1; 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 1 1 means whatever be the control means input I will just increment the value. So, I put 1 1 here and your job is d1. So, automatically MPC will become 2, it will be just incremented because 1 1 as I showed corresponds to increment and not load it to any variable next location.

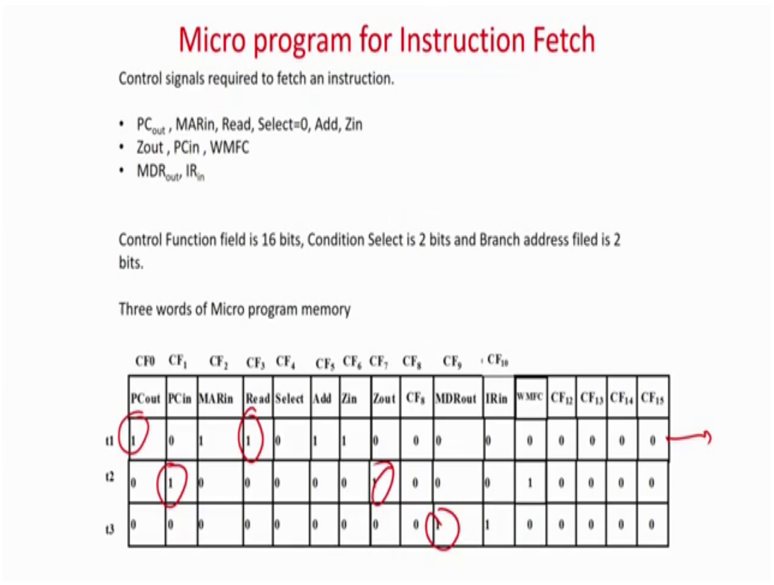
Next one next to see Z out PC in and WMFC, in this case PC in will be a 1 and basically Z out will be a 1, and WMFC is the 1. WMFC 1 means, basically it is a signal which is generated and; that means, you are waiting for another signal from the memory, which has to come if both of them comes then only 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 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 the black box, that unless this WMFC actually ensure unless another same similar acknowledgement come 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, this is the hardwired 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 1 1; 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 0 0 1 1 or 1 0 whatever here, but it is not going to jump to that it is because, conditions are 1 1 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 into the IR. So, in this case you are going to have MDR out equal to 1 and IR in is equal to 1, and then it is basically here after that this actually next program next part will be holt or m 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 m 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 (Refer Time: 48:14). Now, what happens basically the instruction register now, the value has come to the instruction register. It will required 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 will basically it will happen.

(Refer Slide Time: 48:55)



Micro program for Instruction Fetch

Control signals required to fetch an instruction.

- PC_{out} , MARin, Read, Select=0, Add, Zin
- Zout , PCin , WMFC
- MDR_{out}, IR_{in}

Control Function field is 16 bits, Condition Select is 2 bits and Branch address filed is 2 bits.

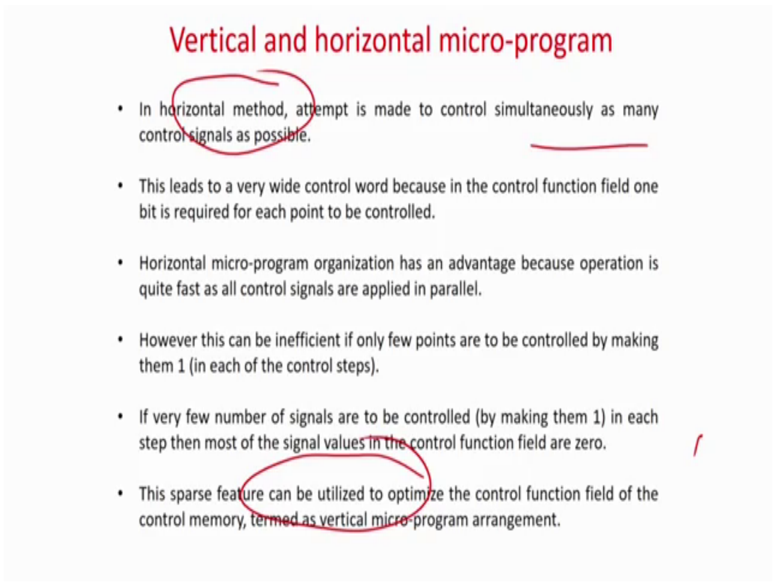
Three words of Micro program memory

	CF0	CF_1	CF_2	CF_3	CF_4	CF_5	CF_6	CF_7	CF_8	CF_9	CF_{10}					
	PCout	PCin	MARin	Read	Select	Add	Zin	Zout	CF_8	MDRout	IRin	WMFC	CF_{12}	CF_{13}	CF_{14}	CF_{15}
t1	1	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0
t2	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0
t3	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

So, as we were looking at basically these micro-program controls. So, one thing you might have observe 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 equal to 1 you are sitting it as 1 in the same time means, that you want to make read equal to 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 very spars matrix like it is second in time is a (Refer Time: 49:27) only require 2 signals to be 1. And in the third you just require 2 signals to be 1.

(Refer Slide Time: 49:36)



Vertical and horizontal micro-program

- In horizontal method, attempt is made to control simultaneously as many control signals as possible.
- This leads to a very wide control word because in the control function field one bit is required for each point to be controlled.
- Horizontal micro-program organization has an advantage because operation is quite fast as all control signals are applied in parallel.
- However this can be inefficient if only few points are to be controlled by making them 1 (in each of the control steps).
- If very few number of signals are to be controlled (by making them 1) in each step then most of the signal values in the control function field are zero.
- This sparse feature can be utilized to optimize the control function field of the control memory, termed as vertical micro-program arrangement.

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

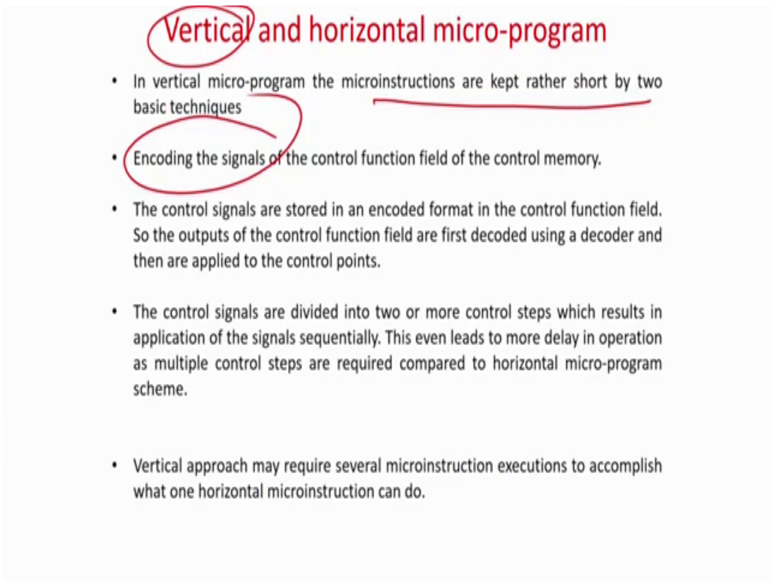
So, this actually method is called a horizontal method, and in which case you simultaneously control as many signals and possible. How it is possible? Basically, you have a memory in the in the micro-program control memory, and you put 0s and 1s as required and each bit of the memory word corresponds to one port. So, you parallely send all the signals to all the ports to be control 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 control the word has to be 32 and so, forth.

But on the contrary if you (Refer Time: 50:21) see you see lot of 0s so; that means, actually the fields parts in contained. So, if you have such parts contained, then you were wasting lot of memory. So, how can we actually implement in a better way so. But, the otherwise advantages like in one go you can control all the control signals required to be given at the 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 they lot of wastage because of spars.

So, therefore, this spars can be utilize to optimize basically; that means, somehow, we can if you can somehow through away these spars values like, whenever it is 0 to do not require them all 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)



Vertical and horizontal micro-program

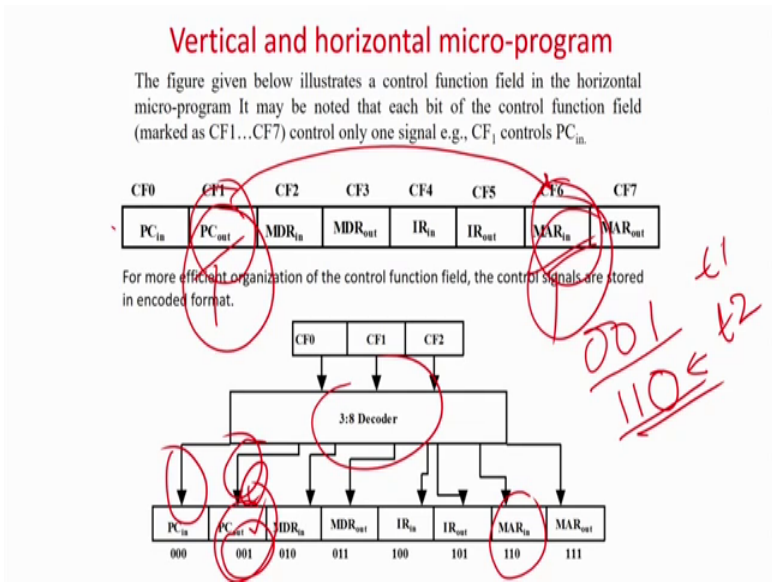
- In vertical micro-program the microinstructions are kept rather short by two basic techniques
- Encoding the signals of the control function field of the control memory.
- The control signals are stored in an encoded format in the control function field. So the outputs of the control function field are first decoded using a decoder and then are applied to the control points.
- The control signals are divided into two or more control steps which results in application of the signals sequentially. This even leads to more delay in operation as multiple control steps are required compared to horizontal micro-program scheme.
- Vertical approach may require several microinstruction executions to accomplish what one horizontal microinstruction can do.

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

So, whenever we want to optimize based on encoding or compressing of the signals in each of these cells or each of the words in the program control memory. We call it as a vertical micro-program in horizontal it is very flat and no optimization is there. So now, what way you can do the only way you can do is (Refer Time: 51:53) encode encoded by mean many way xyz. So, whenever we want to optimize on spars values of a matrix there is only one way to do is they have to encode it.

Now, encoding can be very flat like you if there are 16 bits as output, you go for a 4 is to 16 decoder that is the one way of enco, compressing them there can be several other as we look here. So, basically first the most formus formal or the preliminary way of doing it is basically encoding signals in the control memory. So, what do you do? So, whatever signals has to be applied you just encode the an use decoder, I will show you a figure and then we will come back.

(Refer Slide Time: 52:27)



Vertical and horizontal micro-program

The figure given below illustrates a control function field in the horizontal micro-program It may be noted that each bit of the control function field (marked as CF1…CF7) control only one signal e.g., CF_1 controls PC_{in}.

For more efficient organization of the control function field, the control signals are stored in encoded format.

CF0 CF1 CF2 CF3 CF4 CF5 CF6 CF7
PC_{in} PC_{out} MDR_{in} MDR_{out} IR_{in} IR_{out} MAR_{in} MAR_{out}
CF0 CF1 CF2
3:8 Decoder
000 001 010 011 100 101 110 111
t1
001
110
t2

So, for example, this is one word of the control memory or this is the format of the control memory word. So, PC in, PC out MDR in MDR out all these values are there.

Now, I encoding. So, how many are there? There are 8 bits which has to be controlled or 8 locations which has to control. So, I just put a 3 is to 8 decoder then, what will be the memory size in the memory size will be 3 bits. So, if you give 0 0 0 the corresponding row of the decoder will be 1 you generate PC in equal to 1.

Next if you if you give the signal 0 0 1 this is the memory. So, memory is now 3 bit. So, if you give the value of 0 0 1 0 0 1 in the memory. So, you are going to make this second line high of the decoder, and you are going to get the value 0 0 1 which actually makes this line high. So, in that pc 1 will be PC out will be equal to 1.

So, now, you have to observe that this is one of the drastic way of compressing. So, instead of 2 to the power n the size will just become n or if it is n you will go for log n upper selling. So, it is 8 you are going to get 3 highly compressed value we will have, but the problem is that at any point of time only one of this signals can be made 1. So now, the things will start becoming slower. Say for example. Generally, we know that what happens basically we always have program counter out and memory Address register in. So, this actually always happens simultaneously.

So, here what will be happen first you will have to give the value call 0 0 1. So, if you giving the value 0 0 1 PC out will be equal to 1. But now, simultaneously you cannot also make MAR in equal to 1. So, what we have to do it will be in 2 steps in first step the memory control memory will give the value 1 0 0 1. Which will make PC in PC out equal to 1, then the program counter there will be some stable register or 1-bit flop which will hold the value, but it has to be a 1.

Next step what you do? Next step you give the value of 6 that is mar you have to give the value 1 0 0 in the control memory; that means, you are going to this value will be fed from the control memory decoder. So now, it will need MAR in equal to 1. This is say t1 and this is say t2 then, what it going to happen in 2 times that this configuration will be there, we see program counter output we will go to the memory Address in.

Now, what happens now, but still now PC out is also equal to 1 this is also equal to 1. Now, we have to give a time step where you reset actually, in case if the horizontal micro-program you did not worry about resetting it or holding the value. Simultaneous you give this as 1 and this as 1 and your job is d1, but here we are compressing it we are calling as a vertical micro-program because, it is compressing the structure of the memory. So, you can only make any one point one at a time. So, I make pc 1 equal to 1 and at a time 1, then in the next time go if I make mar equal to 1 in that case there should be some guy to remember that, PC out was equal to 1 at that point of time.

Because, otherwise what is going to happen when I making this as 1, this line will become a 0. Because whenever I am giving the signal 1 1 0 basically PC out will become 1.

So, here there are 2 disadvantages one is that you have to you can you will have more number of stretch required to give simultaneous 1s to the different control positions. We have 2 1s we are required. So, you required 2 if multiple 1s are required you have that many number of time steps. Not only that whatever was one simultaneously required in a horizontal micro-program those values you have to remember like, PC out equal to 1 MAR equal to 1 together they should be 1, but here you have doing in 2 steps.

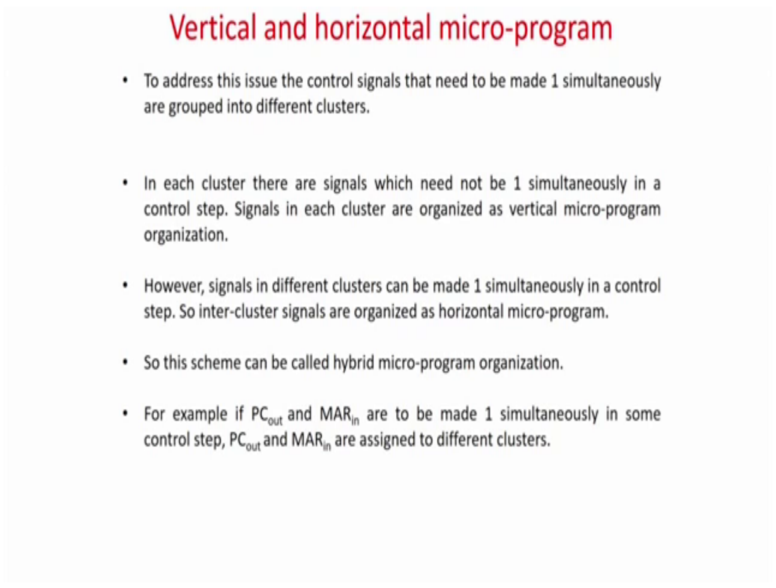
So, somehow the program counter out has to be 1 it has to be remember till you go to the second step, and after the second step is d1 then you again you have to reset this both and then again start from the third location and so, forth. So, that is slide these are these are the 2 disadvantages longer time you have to remember and again you have to reset. But, the advantage is that the memory size actually gets compressed in the value of in the terms of law from 2 to the power n, it will go to the order of 2 n. So, that is the big advantage.

So, if we have lot of 0s then such problems will become in such some kinds of number of stages will be less, but if we have more number of 1s then making them simultaneously 1 is not possible. So, the time steps will increase, but this vertical micro-program advantage comes because, they are all lots of 0s and therefore, only be go for the vertical technique. So, it is also very it is also slower compare to a horizontal micro-program, right?

So, now, we go to the theory. So, what it says that it encodes the signal as I have shown you. the output the control signals are stored in an encoded format in the function field. The outputs of the control function fields are first get into a decoder and then they are applied to the ports as we are having a decoder. So, only 1 bit can be one at a time of the output of the decoder. So, only one control point can be a one. And the control signals are divided into more number of control steps; obviously, because at each point of time only 1 bit can be a 1. So, more number of control steps will be required.

Vertical optimization required several micro instruction executions to do when one horizontal instruction can do it; that means, in vertical approach you required more number of micro instructions in horizontal a very less number of field to because horizontally is parallel approach, and vertically is a sequential approach of solving the problem.

(Refer Slide Time: 57:30)



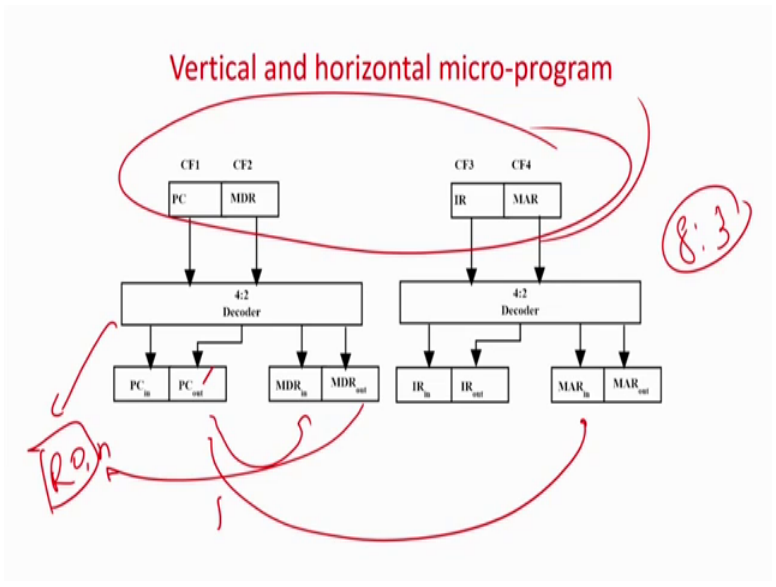
Vertical and horizontal micro-program

- To address this issue the control signals that need to be made 1 simultaneously are grouped into different clusters.
- In each cluster there are signals which need not be 1 simultaneously in a control step. Signals in each cluster are organized as vertical micro-program organization.
- However, signals in different clusters can be made 1 simultaneously in a control step. So inter-cluster signals are organized as horizontal micro-program.
- So this scheme can be called hybrid micro-program organization.
- For example if PC_{out} and MAR_{in} are to be made 1 simultaneously in some control step, PC_{out} and MAR_{in} are assigned to different clusters.

Now, that is one way of solving that is just optimize just full optimization full vertical just encode it by a technique which we have shown you required a decoder to solve it, but now, this another way as I told there are 2 techniques another way is very interesting that can I make a cluster. So, what is the cluster idea I will take a figure and then I will come back to the theory.

(Refer Slide Time: 57:53)

Vertical and horizontal micro-program



So, these are cluster so, we have clustering. Now, we will not make fully means full compression. What we will know say it will try to find out which signals are simultaneously required to be 1, like as I told you PC in and MAR sorry PC out PC out basically always requires to feed memory Address register in that is generally we have observe.

So, what I do? I actually make a cluster 2 clusters. In 1 cluster I will put the PC out in another cluster actually I will put memory Address register in. So, in if there are 2 different clusters basically. So, interestingly what is going to happen I can make this 1 1 and this 1 also simultaneously, but as there are 2 different clusters means there is this is 1 control memory and this is another control memory are there 2 cluster. I means we should not call it 2 different memory basically, logically we have partition into 2 different memory clusters. So, in this cluster only 1 bit can be a made 1 and, in this cluster, also 1 bit can be a made 1, but simultaneously 1 bit can be 1 here and 1 bit can be 1 here. So, that parallel is I have given. So, these actually call a hybrid kind of a multi programing approach, neither it is only horizontal neither it is fully vertical.

So, here what they have d1 they have made a cluster in which case if you look. They are made a cluster of PC in PC out MAR in MAR out 1. Generally, if you see we always write a put in and out of a port in 1 cluster because, generally PC in and PC out did not want to make it 1 simultaneously. Similarly, a memory Address register memory register in memory register out together you do not required to be in.

But, generally we can have memory data register out and instruction register in simultaneously. So, this one I can make 1 and this one I can make together. Similarly, program counter out generally goes over here. So, that will we can make a cluster thinking out that which of the 2 bits can be simultaneously 1, that you put the different clusters and your job is d1. So, by if I do it like this. So, you can see the hardwired requirement is slightly higher. In the previous case we required a 3 1 8 is to 3 decoder, but here you required 2 4 is to 2 decoder. So, 2 is to 4 decoder side area has I the memory number of bits where 3 in this case, but now, it will be 4 bits in this case.

But, this will be slightly faster compare to the full vertical case because, simultaneously I can take one signal here, and one signal here. And we can make one at a time. So, if you make full horizontal then if you want to make only 1 bit in 1 cluster that is 1 extreme which is called the horizontal, and here if you think every think is 1 cluster there is another extreme that is called the fully vertical micro program.

So, in this case the something called a hybrid. So, the basic guideline is you choose the signal which very frequently has to be once. So, that you put it separate cluster. So, here I have given example of 2 clusters you can make multiple clusters depending on the needs the motivation is in each cluster you should not put signals which requests to be want simultaneously. But, it may not be easily solvable because, if you go that go to that level that we are keeping individual bits in a cluster. Which can never be one together then we will go to a fully horizontal micro-program when each bit is independent.

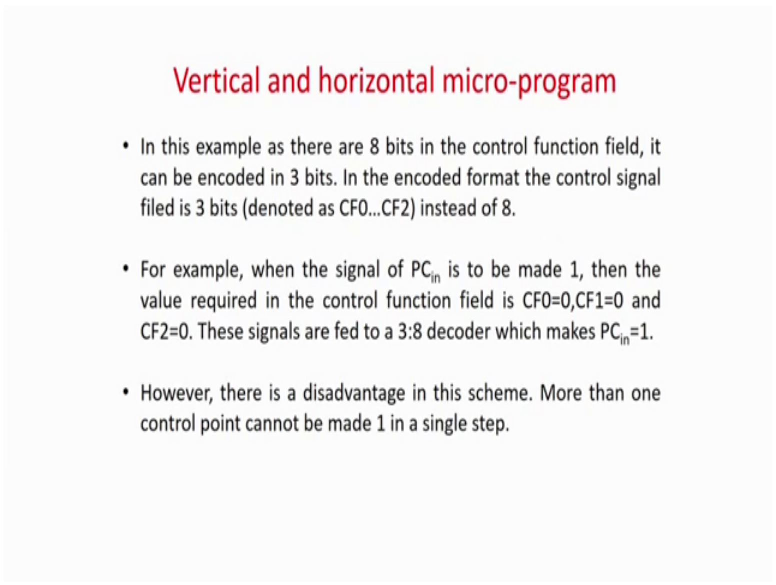
So, we can try to think that in very high number of micro instructions which signals are to be one at in one point of time. You give to give it to a different clusters like PCout and memory Address register. In this will always happen in the micro-program instructions corresponding to fetch of an instruction. So, you put it to a different clusters so, that is basically idea.

But, sometimes if you say that the program counter in may be you have to go to for example, I can say that somehow for some reason may be say that program counter out has to go to memory data register in that is a stray example then of course, you require 2 bits for example, if you also I am saying that here there is a another cluster which is having say R0 in there is some R0 registers also in this cluster. So, just example, in this case basically the memory read out and R data register in will not happen for that you will required 2 steps. So, that the cluster ; that means, whatever is in one cluster if you want 2 signals to be one simultaneously they has to be d1 in 2 steps, that cannot be d1 in a single stage.

In that case the time step will increase, but in between large number of clusters are there for for inter cluster multiple bits can be 1 simultaneously. So, this way of compressing is actually called a hybrid approach, and sometimes we call hybrid kind of a vertical organization. So, basically that is what is the idea.

So, another technique basically is to be made clusters. So, in each cluster signals which did not be 1 simultaneously in one step you put in 1 cluster. And which is 1 need to be 1 simultaneously you put a different clusters. For example, like PC out and memory in should be made simultaneously, similarly whatever is telling in the example is written in the slide you can go through the slides basically.

(Refer Slide Time: 62:38)



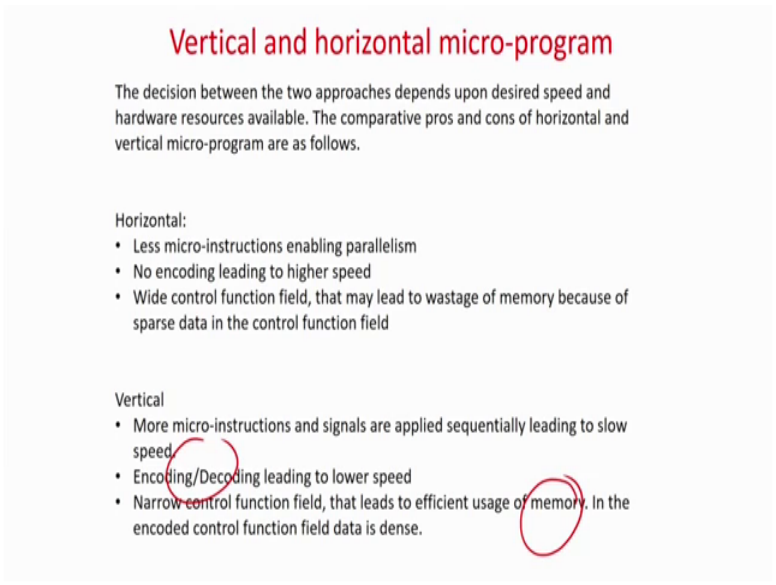
Vertical and horizontal micro-program

- In this example as there are 8 bits in the control function field, it can be encoded in 3 bits. In the encoded format the control signal filed is 3 bits (denoted as CF0...CF2) instead of 8.
- For example, when the signal of PC_{in} is to be made 1, then the value required in the control function field is CF0=0,CF1=0 and CF2=0. These signals are fed to a 3:8 decoder which makes PC_{in}=1.
- However, there is a disadvantage in this scheme. More than one control point cannot be made 1 in a single step.

So, now, if you look at a comparison, in this case basically these are basically what idea in this example there are 8 bits that if you look at this slide. So, means how do make a control etcetera all the comparisons are set. So, in this example there are 8 bits in the control field. So, you require a decoder to solve the problem. For example, if pc is to be made 1 then the values of the control field 0 0 0 and decoder will make it as the 1. So, it actually inter this slide actually the interpretation of the figure.

This interpretation of this figure is given in this slide, and I have already explained you this is just for you to read and get what was I was actually discussing. But, disadvantage is that more than one control point cannot be made in a step, if it is a full vertical arrangement. These actually this slide actually explains this field. This is what I have displayed and how do an inter cluster organization.

(Refer Slide Time: 63:31)



Vertical and horizontal micro-program

The decision between the two approaches depends upon desired speed and hardware resources available. The comparative pros and cons of horizontal and vertical micro-program are as follows.

Horizontal:

- Less micro-instructions enabling parallelism
- No encoding leading to higher speed
- Wide control function field, that may lead to wastage of memory because of sparse data in the control function field

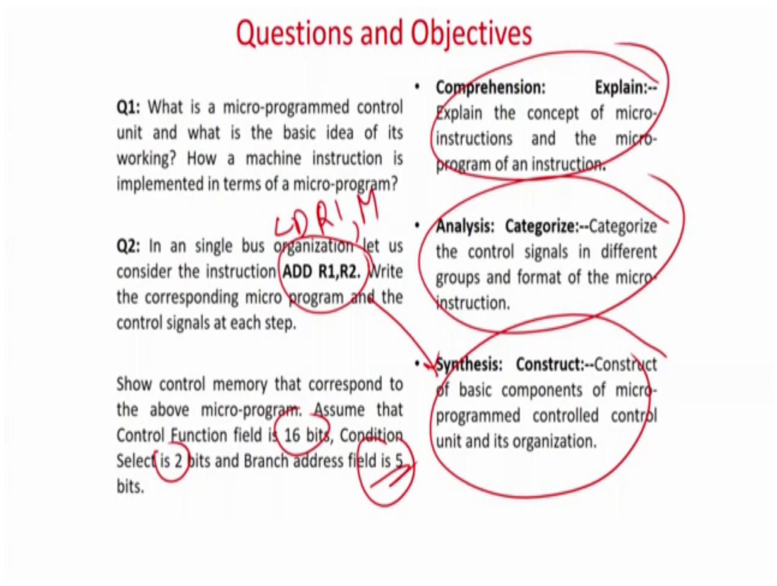
Vertical

- More micro-instructions and signals are applied sequentially leading to slow speed
- Encoding/Decoding leading to lower speed
- Narrow control function field, that leads to efficient usage of memory. In the encoded control function field data is dense.

So, now, consolidating what we have discuss in horizontal micro-program less number of micro instructions are required. It is a fully parallel architecture no encodings are extremely fast, but actually wide control field is required. So, the memory size will be higher, and there will be because there are lots of 0s involved generally because, only 1 or 2 points (Refer Time: 01:03:52) controlled simultaneously that is made 1. So, there is also memory wastage.

In vertical what we do the actually we are encoding the control units control signals. So, there is a less number of memory is there is required. So, wastage is less, but you required decoding and encoding. So, leading it to slower speed, and not only because of decoding you require multiple number of states to solve the same control problem. So, it is a long time taking approach, and what we have also seen that there is some kind of an hybrid approach in which you can make nice clustering based on the fact that 2 signals. Which has to be 1 should be given 2 different clusters then, you get something good trade off that without wasting much memory, you get a almost near speed of as a fully horizontal micro-program.

(Refer Slide Time: 64:32)



Questions and Objectives

Q1: What is a micro-programmed control unit and what is the basic idea of its working? How a machine instruction is implemented in terms of a micro-program?

LD R1, M

Q2: In an single bus organization let us consider the instruction **ADD R1,R2.** Write the corresponding micro program and the control signals at each step.

Show control memory that correspond to the above micro-program. Assume that Control Function field is 16 bits, Condition Select is 2 bits and Branch address field is 5 bits.

- **Comprehension: Explain:--** Explain the concept of micro-instructions and the micro-program of an instruction.
- **Analysis: Categorize:--**Categorize the control signals in different groups and format of the micro-instruction.
- **Synthesis: Construct:--**Construct of basic components of micro-programmed controlled control unit and its organization.

So, basically with this we come to the end of this unit. In the next unit basically, we will look in more depth basically that type depending on different type of instructions or macro instruction. How exactly the full instruction will basically go through? Ah the micro instructions are how they will be executed? Because, (Refer Time: 01:04:55) you will be have try to focus on the fetch fetch part, but after fetching actually there is that instruction decoding and depending on the type of instruction whether it is an ADD or subtract or something you have to go to different micro different location from where the corresponding micro instructions are there, that is how of all instruction can be executed.

So, all those comp1nts will be looking in the next unit this will be actually give you an idea, but what is the basic micro-program architecture how you can optimize? What are the basic micro-program? In memory looks like what is the micro-program counter? The basic idea we have given. In the next unit actually, we will try to go in slightly more depth with the concepts and try to see basically main idea will be how we can basically take a full code, and try to see the full execution in terms of micro instructions. That is, we will try to go in elaborate more on some of the points, which we have the basics which we have discussed in this unit.

So, before we end as we already discuss some questions and try to see the objects are made. So, the first question is what is a micro-program control unit? What is the basic idea of working? How a machine instruction is implemented in terms of micro program?

So, after discussing this unit I think we will be able to answer this and once it is answered basically this comprehension object that explained the concept of micro instruction, and basically construct the basic comp1nts because, how a machine instruction is implemented in terms of a micro program? So, the basic architecture etcetera, this comprehension and this construct synthesis objective will be made because, we will be describing a basic architecture you can you should be able to basically describe the basic architecture of a micro-program control unit which you have shown and then, just you take how our code actually gets broken down into some terms of some signals the basic theories? So, this actually fulfills these objectives.

The second question is I asked you that you take a instruction ADD R1 and R2. And you take a control memory 16 bits signals condition bit is 2 bits and Address bit is 5 bit 5 5 bit; that means, the Address spaces of 2 to the power 32. I ask you to that you please try to see how this full instruction can be executed because, mainly in this unit we have discussed the fetch part of it. After fetch part if it is an ADD some other set of micro instructions will be ADD it is. If it is LOAD some other set will be execute that is fetch part is common, but after that it gets divert is bifurcated depending on the type of instruction.

So, now, please try to your because already we have discussed how to do for the fetch. Please try to extend it for ADD and basically see how the micro-program memory looks like. And of course, we show the 3 bits 3 memory locations are required for fetch, but as I have asked you to go for the other steps also that is executing the instruction. So, we require a higher memory size. So, I have kept the Address bit field is 5. So, that you have the places to put the other signals which will be required for exact execution of the, and I R1 and R2 which basically comes after the instruction is decoded. So, that part you can try.

So, this actually questions will directly help you in meeting this objective that construct the basic comp1nts of micro-program control unit and it is organization. Because here, you are been have the fetch stage as well as have the decoder decoding and the execution stage also. In fact, again we will be able to categorize the control signals in terms of groups and format of the micro instruction. Because, in this case we will have basically 2 basic parts will be there, one is for the fetch and the other part is for the execution. So, you will be able to categorize also if you want to try with ADD then, also you can try with LD R1 memory if you try this 2 basically then we will be able to find out how I can categorize the control signals based on the different format of micro instructions for different rate of codes etcetera. So, by at this 2 questions actually will be easily able to you should be able to solve these questions based on the discussion in this unit, and actually you made this objectives which was the target of this unit with this we come to the end of this unit.

Thank you.