

(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.

Handwritten notes: "ADDRIR2" is circled in red and underlined. A red bracket is drawn on the right side of the slide.

So, basically the micro-programmed control unit ensures that the signal will be generated in correct sequence. So, sequencing 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 row 1 memory row 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 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 zero flag, carry flag so all these things we must take into 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 where 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 you 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 difference, they are 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 then it will be easy, like for example if you have some instructions called *ADD R1 R2*. So, first phase is called the fetch an instruction.

So, we have discussed many many times that for most of the instruction fetch phase is similar. So, whenever instruction has to be fetched the 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 invoked.

So, that series of micro instructions will execute, which will actually execute *ADD* which is from register to register operation, and then it will go slow. So, we will take some examples then it will be more clear. For the time being let's take the fact that instruction fetch, decode and execute.

So, fetch everything is similar. So, whenever a new instruction has to be fetched, ah fetch block of micro instructions which corresponds to its memory fetched will be invoked. Invoke means micro-program counter will start pointing to the first microinstruction, which corresponds to instruction fetch.

Then, when instruction fetch is done then you are decoding in the instruction register, and then accordingly it will tell that it's a *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 has been actually is the job of a micro-programmed control unit. So, apart from branch address these micro programs can specify which status flags conditions etcetera has to

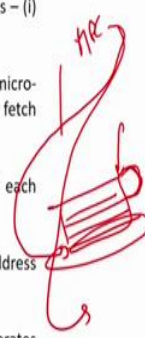
be taken for the condition check. So, that that means, in a very nutshell basically you just go micro-program counter increases one by one, but whenever there is a condition which depends on the input, depends on the flags, depends on some I/O input. Which is which you get it from I/O device then all 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 memory, maybe this part is reserved for the instructions 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 will have the instruction like *ADD R1 R2* store 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 corresponds to the micro instruction starts where the micro instruction starts which corresponds to *ADD register to register*.

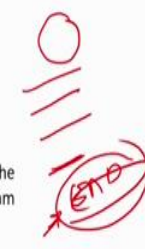
If the *STORE*. So, it may be pointing to some other location in the micro-program which corresponds to the start address of the micro-program which corresponds to *LOAD* or *STORE*. So, basically this is how it actually happens. That is for any instruction the first phase is similar that is the fetch, and then after that 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 corresponds 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:

- ① • 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.
- ② • 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.
- ③ • 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 which I was saying that during the execution of a micro-programmed control, the *MPC* is always incremented. So, it goes excepting 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 an end instruction? Basically, say for example, as I told

you fetch is a default it means, default sequence that is to be executed whenever any instruction has to be taken from the memory and execute.

So, whenever an instruction is there 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 then it does not go ahead; obviously, because maybe these 3 micro instructions are required for fetch. So, after that this part actually ends for that micro instruction

So, whenever it ends then it has to again wait whenever it reaches the end instruction it does not go that is obvious. Basically if you take an instruction like *ADD R1 and R2*. So, that is the macro instruction it will have many small small micro instructions programs for that like fetch will comprise for one micro-program then *ADD 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 comprise 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, like 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 it 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 micro-program the *MPC* points to 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 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 one 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 it is *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.

Handwritten annotations: A red circle around the title, a red circle around the first paragraph, a red circle around the second paragraph, a red circle around the third paragraph, and a red circle around the fourth paragraph. A red arrow points from the first paragraph to the second paragraph. A red arrow points from the second paragraph to the third paragraph. A red arrow points from the third paragraph to the fourth paragraph. A red arrow points from the fourth paragraph to the fifth paragraph.

So, basically this is what in a nutshell we have seen micro-program, for a given program the 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 step, that is we have to do the sequencing that is what the jobs are. So now, we will see how basically a 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 has to be made 0 and 1.

(Refer Slide Time: 24:54)

Organization of a micro-programmed controlled control unit

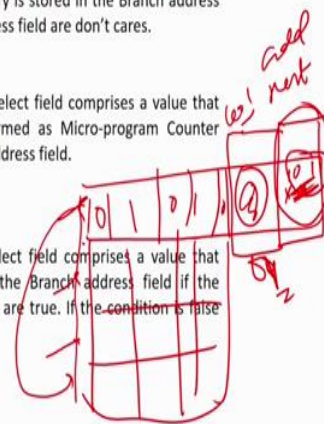
- However, as there can be branching based on inputs, status variables and flag variables, the address of the control memory in the next control step may not be simple increment.
- The next address of the control memory depends on the next executable micro-instruction which depends on inputs, status 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 one case there is a branch instruction. So, in this case basically you have to think of actually next 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 filling the values 0110 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 another first this location this location, this location and so on.

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 is this is the present state next state 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 we 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 concretely 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 011011. Some part of the memory I will reserve at the next address register, if I say that it will be xx or 00 or something then it will say that I have to default go to the next instruction. But, if I say that some values are there may be 1011 some coding is there which will tell maybe 101; that means, it will tell that the next instruction is not this 1, but it has to go to 101. If some condition based on this memory location holds maybe if you say that 00. It tells that no condition has to be checked it will directly go to 1 that is default jump to memory location 11.

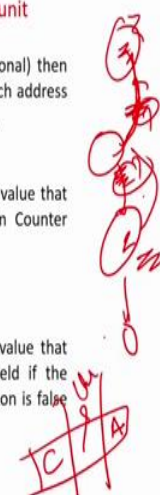
But, sometimes you may say that 01 and 101 and this 01 may correspond to saying that basically the carry flag should be 1. So, if the carry flag is 1 then you jump to 101 else you increment. So, the, this is the way a micro-program memory will look like. You will have one part which is the most important part of it that is the control signal. Some part you have to reserve for the conditions which you have to check we can say 00 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 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.



IC	A
1	2
3	4
5	6
7	8

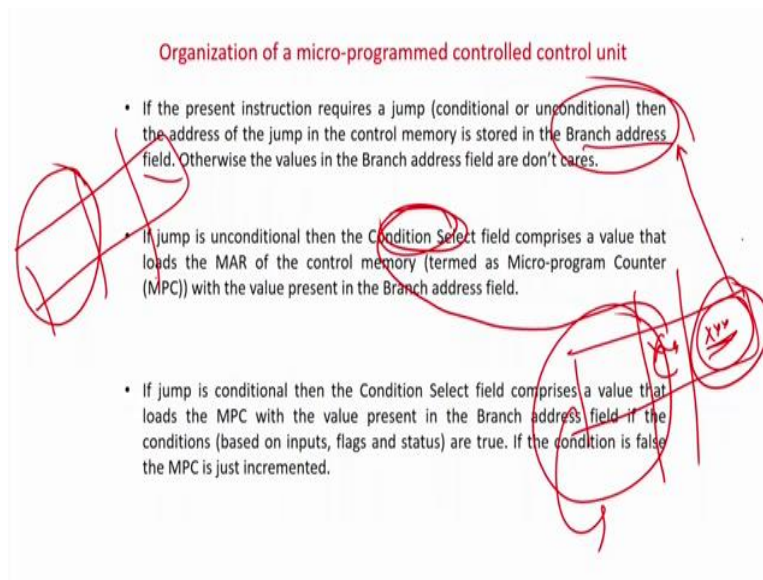
So, as this is not a finite state machine. So, in case of a finite state machine basically you can have some conditions, and the sequencing actually is maintained. Like if this is the case you go over and if this is the case you go over.

So, this condition checks are put in these transitions and this destination and this destinations are denoted by the states. So, here we 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, so location of this 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 corresponds to 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 what I was saying.

(Refer Slide Time: 29:08)



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's a default variable. If the jump unconditional this is actually called the condition select field this is actually called the condition select field of course, because the jump etcetera 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 inputs from the I/O devices that you can mention by some code over here.

So, if it is unconditional jump then, you can make $c = 00$ 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 arbitrarily encoding like 000 and 000. 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 generate your control signal one part will be basically for the control select. That is the depending on what you have to take decision and one will be actually the branch address field. So, basically there are 3 parts.