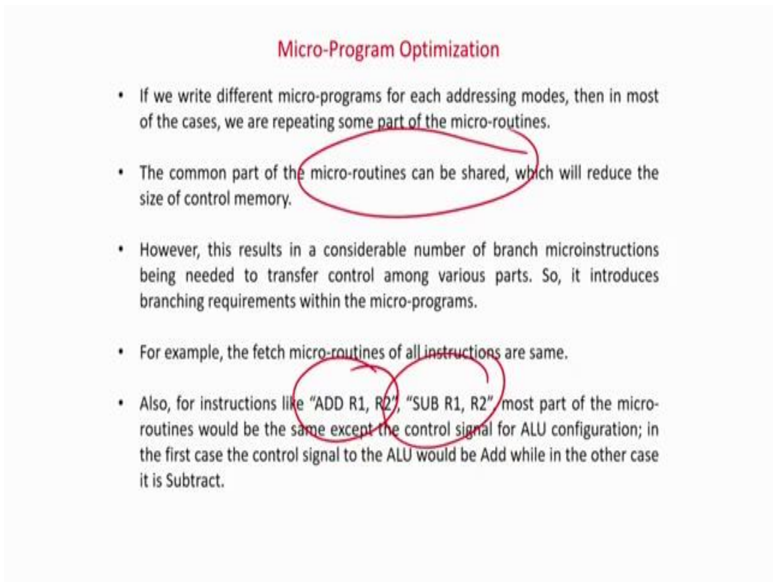
(Refer Slide Time: 38:35)

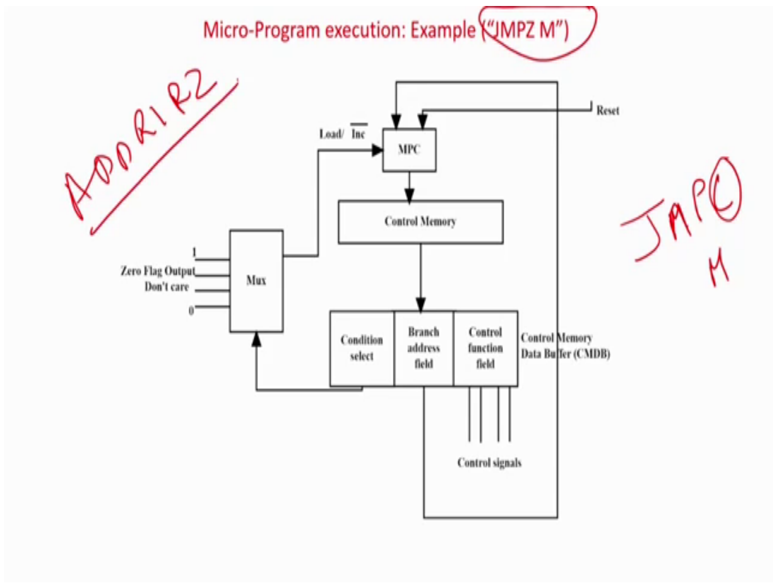


Micro-Program Optimization

- If we write different micro-programs for each addressing modes, then in most of the cases, we are repeating some part of the micro-routines.
- The common part of the micro-routines can be shared, which will reduce the size of control memory.
- However, this results in a considerable number of branch microinstructions being needed to transfer control among various parts. So, it introduces branching requirements within the micro-programs.
- For example, the fetch micro-routines of all instructions are same.
- Also, for instructions like "ADD R1, R2", "SUB R1, R2" most part of the micro-routines would be the same except the control signal for ALU configuration; in the first case the control signal to the ALU would be Add while in the other case it is Subtract.

So, basically what we can do is that, we can actually write a micro routines which can be shared. For example, as I told you like add and sub. So, most of the case will be similar excepting 1 bit position or the 1 control signal corresponding to add or subtract of the ALU. So, you can try to do basically write a basically a single micro program for similar type of basically macro micro instruction macro instructions, which are common in type this slide actually these 2 slides basically these 2 slides basically tells you what I have discussed. So, you can read in offline.

(Refer Slide Time: 39:10)



Now, let us again now actual I will tell you two things, two things are very important over here. So, as I told you in the last class that in this unit we will also see how a complete macro instruction is executed in terms of micro instructions. Because in the last unit we just saw that how to do a fetch here we will see how a total instruction is executed. Secondly, as I told you we also give some idea that how micro instructions can be macro instruct micro instructions corresponding to different type of similar macro instructions can be clubbed.

Right for example, we are going to show you how a complete macro instruction is executed in terms of micro instruction, and we were going to take the help of jump on zero M ($JMPZ\ M$), that is the that is the example of the macro micro instruction we are going to take, and we will try to see how it is implemented in terms of micro instructions. But again as I told you we can optimize based on a single micro routine for difference similar type of macro routines, macro instructions basically.

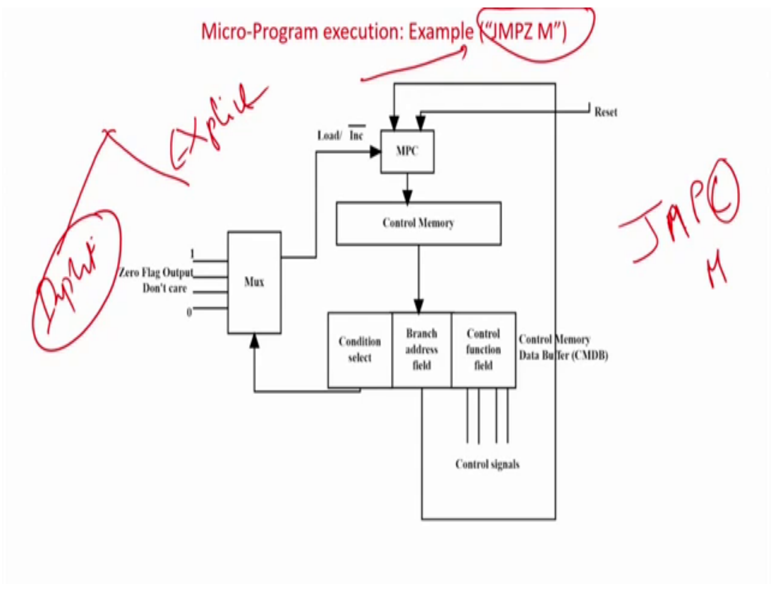
So, let us take another may be jump on carry and may be you and we say that to jump on M, jump on M another instruction which is very macro instruction which is very similar. So, this is jump on 0 and basically this is on jump on carry to memory location M. So, basically they are very two similar macro instructions, they will have almost similar format and similar type of control signals will be generated, except in the case in one place you are going to check the 0 flag and the other case you are going to check the carry flag that will be the only difference here.

So, we will see how to try to optimize and do this. Again there is very another very important thing you have to note over here. If I say that I will have a similar macro program micro program for different type of macro instructions which are common in type and then you will based on the different part we have to implement, you will jump from the main micro program and again come and execute the different part and then again come back and execute the common routine so; that means, there will be so many branching to and forth in the micro program.

So, this type of branching will be actually called the implicit jumps we have to do, because you are writing a common micro program for different macro program. So, this is not a requirement of the macro program. So, for example, if I have add $R1$ and $R2$. So, if this instruction is there you do not require to explicitly write jump instructions in the micro routine because this is this is not a jump instruction, but as I told you we will not have a single macro micro program for add and sub and may be multiply, we will have a single routine and based on whether it is add, sub or multiply, you will jump.

So, this is actually the implicit jump routine, which will be there in the micro program, and then another type of jumps which will be there which are actually explicitly mention in the macro; macro instruction corresponding to that right for example, $JMPZ$. So, it says that if the 0 flag is set you jump to memory location M, that is an explicit jump instruction which has to be in built in the micro routine. So, there are basically 2 parts.

(Refer Slide Time: 42:00)



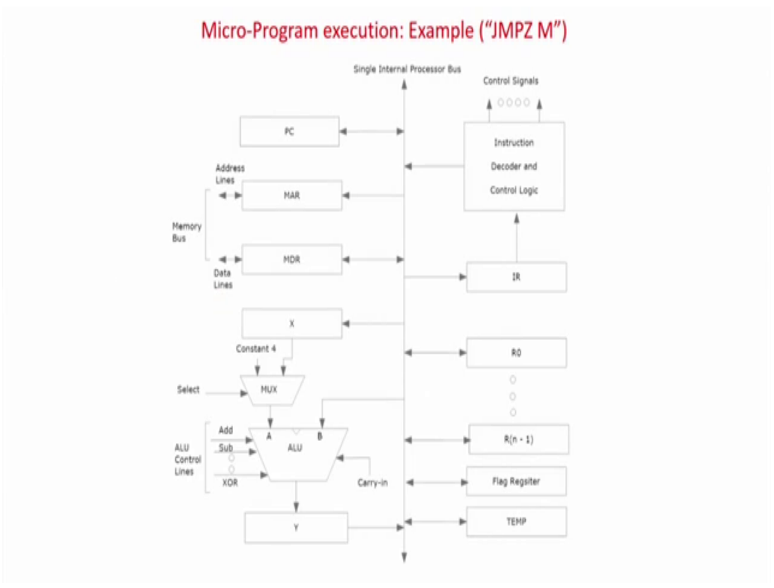
One is called the explicit that was because of the instruction type if it is add, sub this will not be there, and there will be some implicit type which will be coming in to the micro routines because you are doing an optimization. So, there are 2 types of things will be required. So, there for I have explicitly chosen this instruction which is called the jump M. So, that I can illustrate both type of jumps which has to be there in the micro routine.

And let us assume that we have we are going to write a common routine corresponding to jump on Z to M and jump on carry to M. So, this figure if you look at. So, this is already we have discussing in the last unit just a refresh. So, these are the control signals to be generated, this is the branch field address; that means, if you have to jump to some location that value will be given over here that will be loaded to the micro program control unit if and only if the load value is 1. If the load value is 1 it will take the value of the branch control address, and if it is 0 then basically it is going to just increment the value already we have seen.

And therefore, who selects whether the you have to jump or whether you have to load the branch address field or just increment depending on the control select. For example, if you are control select contain condition select is 1 one the last bit of 0 will go and you will always be incrementing mode. If you give 00 that is again unconditional jump, if the condition select is 00 then basically what happens? The 1 is going the first line of the mux will be connected which is one. So, this will be a 1 and always it will be load the value from the branch address and you will make a unconditional jump.

The other two things basically are connected to different conditions like 0 flag instruction register output and so forth. So, accordingly we will we have already seen this thing and accordingly we will fill up the values means these 2 basically things in this case can be reprogrammed based on connectivity and this is the example of a single bus architecture.

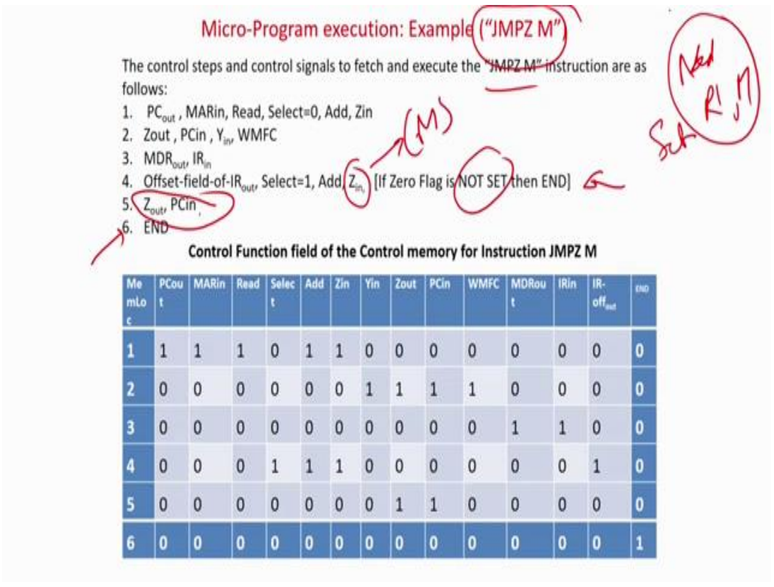
(Refer Slide Time: 43:48).



Micro-Program execution: Example ("JMPZ M")

We have discussed in so many days just have a look at it, because we will be discussing our example for jump on Z on M on this bus architecture.

(Refer Slide Time: 43:58)



Micro-Program execution: Example ("JMPZ M")

The control steps and control signals to fetch and execute the "JMPZ M" instruction are as follows:

1. PC_{out} , MARin, Read, Select=0, Add, Zin
2. Zout , PCin , Y_{in}, WMFC
3. MDR_{out}, IR_{in}
4. Offset-field-of-IR_{out}, Select=1, Add, Z_{in}, [If Zero Flag is NOT SET then END]
5. Z_{out}, PCin
6. END

Control Function field of the Control memory for Instruction JMPZ M

MemLoc	PCout	MARin	Read	Select	Add	Zin	Yin	Zout	PCin	WMFC	MDRout	IRin	IR-off$_{out}$	END
1	1	1	1	0	1	1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	1	1	1	1	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	1	1	0	0
4	0	0	0	1	1	1	0	0	0	0	0	0	1	0
5	0	0	0	0	0	0	0	1	1	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	1

I have again come to that. So, again recollecting some 2 3 lectures back. So, what is jump on 0 does. So, basically if you look at it first will be program counter will be output that will be going to the memory address. That means, whichever instruction you have to read that addresses in MAR in memory is in read mode, $select$ is 0; that means, you are going to take default value of incrementing the PC. So, $select = 0$, ALU is in add mode which will do $PC = PC + constant$, which will be going to the temporary register Z_{in}.

Next is you have to $Z\ out$ to $PC\ in$; that means, $PC = PC + constant$ and basically and it will go to PC and also you are waiting basically it will still for the memory to give the read signal. So, these things are very common just already we have discussed so many times, just I am giving out the values and just I am repeating it for ease of readability.

Then we are saying that the memory data register after it has read will go to instruction register now the instruction register has the value of jump on Z to M. Now look at this, this instruction is very important here it is actually the explicit jump command in this program in this macro program. So, this is the micro instructions or micro control signals for this macro instructions. It say that offset field of IR out already we have told you that the already we have discussed some lectures back that offsets field of IR you add with the that value of program counter and therefore what you get basically that we the PC is also stored in Y_{in} for that reason. So, Y in that is the present value of $PC + offset$ if you add it, you are going to get the value of the jump which will be actually stored in Z.

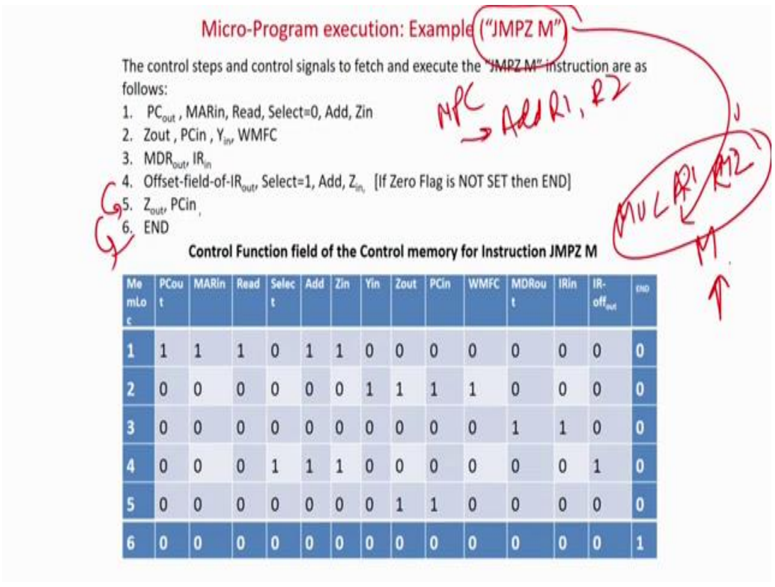
So, now by this instruction offset value of $IR, select, Add$. So, 1 input of the basically of the ALU is basically going to be Y_{in} and the second value is going to be the offset value you add it, and actually Z which is connected to the output of the ALU you have to just recollect some 3 4 lectures back. So, Z_{in} is actually going to have the value of memory location M. So, that is a repetition. So, till now what happened if you execute up to 4, Z is going to have the value of memory location M.

Now, it says that if it is jump on 0, if it is 0 then basically you have to load the value of $PC = M$ otherwise you don't have to do that. So, what you have to execute the next macro instruction ok. So, what we will do? Basically if 0 flag is set, if the 0 flag is set then you are going to execute this one Z_{out} equal to PC_{in} otherwise basically you have to go to number 6 micro instruction which corresponds to end now again what is happening maybe after jump on Z M because the 0 flag is not set; that means, 0 flag is not set in that case you have to jump to end, after end what happens this micro routine ends and may be after this macro program of this macro instruction after jump on Z maybe say $ADD\ R1, M$ is there in your macro routine.

So, therefore, again the micro program corresponding to this will load or basically what is going to happen is the micro program counter micro-program program counter will start pointing to the micro instructions corresponding to this; that means, the next macro instruction will be executed if the 0 flag is not set. So, if the 0 flag is not set, you are going to jump over here. So, it says that if the 0 flag is not set then go to end basically the; that means, basically you have to load the value of *PC* with *MPC* with the value of 6.

Else you do not have to do anything, if the 0 flag is set if the 0 flag is set; that means, it is jump on 0 you don't have to do anything you have to go to memory instruction number 5 micro instruction number 5 Z_{out} will be loaded to PC_{in}, and basically you have and come after that you will go to 6; that means after, at present the value of $PC = Z_{out}$ that is equal to M memory location M. So, the in the macro program basically we will jump from this to memory location M which is may be the instruction which we want to execute which may be same $MUL\ R1\ and\ R2$ and may be just after this maybe we have some called $ADD\ R1\ and\ R2$.

(Refer Slide Time: 47:53)



Micro-Program execution: Example ("JMPZ M")

The control steps and control signals to fetch and execute the ~~"JMPZ M"~~ instruction are as follows:

1. PC_{out}, MARin, Read, Select=0, Add, Zin
2. Zout, PCin, Y_{in}, WMFC
3. MDR_{out}, IR_{in}
4. Offset-field-of-IR_{out}, Select=1, Add, Z_{in}, [If Zero Flag is NOT SET then END]
5. Z_{out}, PCin
6. END

MPC
→ Add R1, R2
MUL R1 R2
M

Control Function field of the Control memory for Instruction JMPZ M

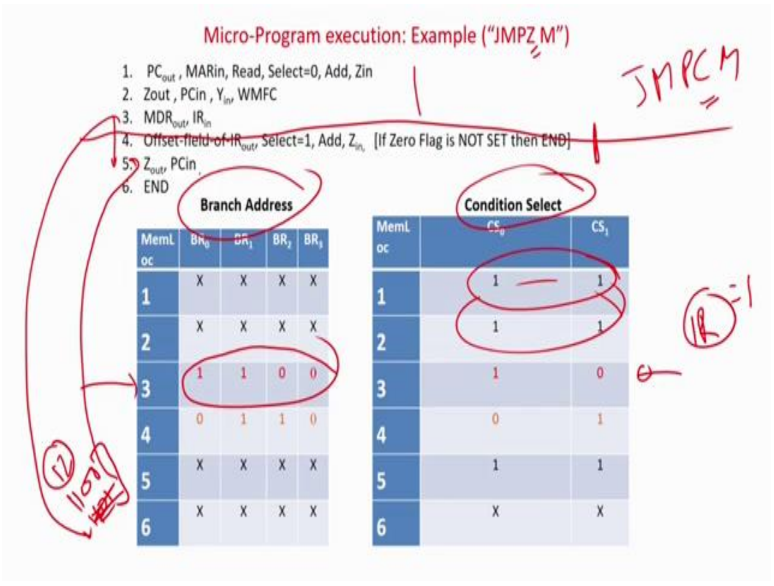
Mem Loc	PCout	MARin	Read	Select	Add	Zin	Yin	Zout	PCin	WMFC	MDRout	IRin	IR-off_{out}	END
1	1	1	1	0	1	1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	1	1	1	1	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	1	1	0	0
4	0	0	0	1	1	1	0	0	0	0	0	0	1	0
5	0	0	0	0	0	0	0	1	1	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	1

So, what is going to happen? So, if the 0 flag is not set then you go to end; that means, I will jump for directly 6 directly to 6 basically and in the this is the micro program and the corresponding macro program what is going to going to happen is that, you will end the micro program for jump on 0 and before the MPC will start pointing to the next instruction basically which is nothing but $ADD\ R1\ and\ R2$ that corresponding micro program will start executing.

And if the 0 flag is set and if the 0 flag is set basically if you see then the micro program will not jump it will go to 5 and in that case what is going to happened? $Z_{out} = PC_{in}$ the macro level multiple $R1$ and $R2$ which is in the memory location M is going to be loaded and after this these a this micro program will end, and the next micro program PC will start pointing to this instruction micro program and not to $ADD\ R1, R2$. So, this is what is the basic flow.

So, now, let us see how actually it happens in terms of micro program. So, if you look at the first micro instruction PC_{out}, MAR_{in}, $read$ and all these things. So PC_{out} is 1, MAR 1, $read$ 1 and basically add and $R1 = 1$ corresponding signals are all 1 two also if you look at it $Z_{out}, Y_{in}, Z_{out}, PC_{in}$ and so forth, 3 is basically if you see MDR_{out} and this one. So, it is very simple you can just trace out the field and you can find out the corresponding signals are one. So, this is very simple nothing much to look at the control memory here.

(Refer Slide Time: 49:34)



Micro-Program execution: Example ("JMPZ M")

1. PC_{out} , MARin, Read, Select=0, Add, Zin
2. Zout , PCin , Y_{in}, WMFC
3. MDR_{out}, IR_{in}
4. Offset-field-of-IR_{out}, Select=1, Add, Z_{in}, [If Zero Flag is NOT SET then END]
5. Z_{out}, PCin
6. END

Branch Address

MemLoc	BR_0	BR_1	BR_2	BR_3
1	X	X	X	X
2	X	X	X	X
3	1	1	0	0
4	0	1	1	0
5	X	X	X	X
6	X	X	X	X

Condition Select

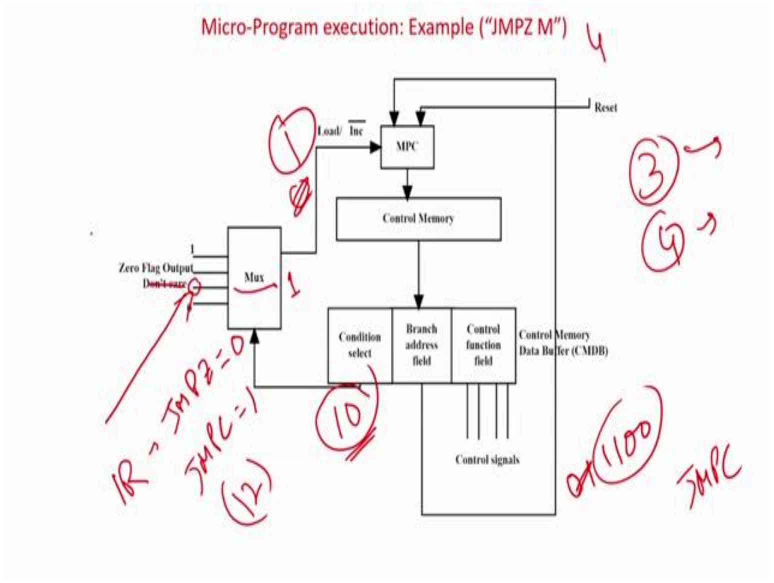
MemLoc	CS_0	CS_1
1	1	1
2	1	1
3	1	0
4	0	1
5	1	1
6	X	X

JMPCM
IR =1
12
1100

What is very important is, what we have to see in the slide. So, basically if you see if this is your condition select and this is your branch address select now I will have to do some connection in the figure, which I am going to do. So, in this case I am this don't care I remove over here and here I have to put instruction register output.

(Refer Slide Time: 49:50)

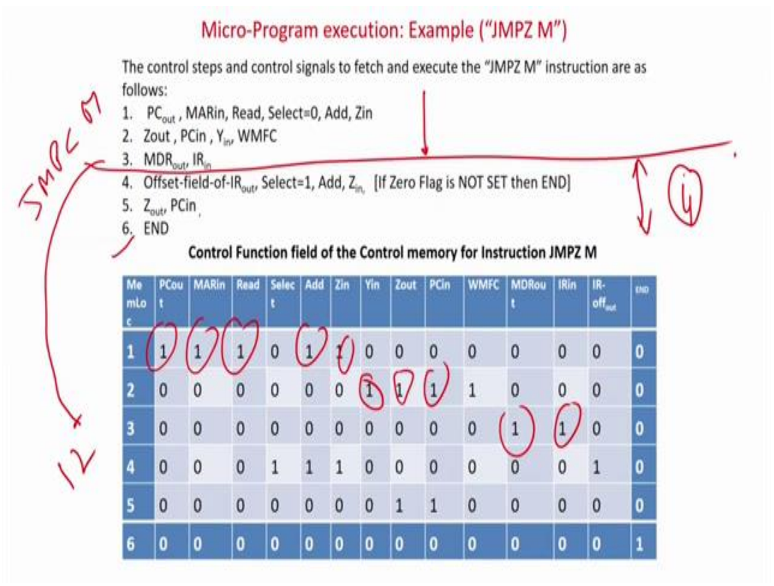
Micro-Program execution: Example ("JMPZ M")



Because the instruction register is now going to tell you in this case whether it is a jump on zero instruction or maybe it is telling a jump on carry because I am just assuming that, I have only single macro code sorry micro routine for jump on Z and jump on carry.

So, basically in this case I assume that because if it is the jump on 0. So, basically I assume that the same micro routine has to be executed, and if the instruction register says that it is jump on carry, then I may have to jump to some other memory location micro memory location micro program memory location may be 12, which is actually exactly contains the value. For example, as I tell you up to 3 is it nothing but it is your basically fetching the instruction, and this is corresponding to the execution of jump on Z.

(Refer Slide Time: 50:38)



Micro-Program execution: Example ("JMPZ M")

The control steps and control signals to fetch and execute the "JMPZ M" instruction are as follows:

1. PC_{out} , MARin, Read, Select=0, Add, Zin
2. Zout , PCin , Y_{in}, WMFC
3. MDR_{out}, IR_{in}
4. Offset-field-of-IR_{out}, Select=1, Add, Z_{in}, [If Zero Flag is NOT SET then END]
5. Z_{out}, PCin
6. END

Control Function field of the Control memory for Instruction JMPZ M

MemLoc	PCout	MARin	Read	Select	Add	Zin	Yin	Zout	PCin	WMFC	MDRout	IRin	IR-off_{out}	END
1	1	1	1	0	1	1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	1	1	1	1	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	1	1	0	0
4	0	0	0	1	1	1	0	0	0	0	0	0	1	0
5	0	0	0	0	0	0	0	1	1	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Maybe I here from here actually after if I am if the instruction is jump on 0, from 3 I have to go to 4 5 and 6.

But assume that if this instruction or the macro instruction is not jump on Z maybe it is on jump on carry to M maybe it starts from memory location number 12 micro memory micro instruction memory number 12, then it has to jump and execute the corresponding instruction from 12. So, this is actually the implicit jump. So, the micro routine will come up to here, after that it will look at the instruction register decoder output. So, if it says that basically it correspond to jump on Z there will be no jump, but if it correspond to jump on carry it will jump to 12.

So, basically what I will do, I will actually connect instruction register to this the third line of the mux and I will say that if jump on 0 is there basically it is going to give you 0 instruction decoder and if it is the jump on carry micro instruction it will give us 1. So, and the condition select at the time is obviously equal to 10. So, at this point after a third micro instruction when the fetch is over, I will give condition select equal to 10 and of course, branch address select will be 12. So, it will be 1 sorry 1100 I assume that from 12, the jump PC, jump on carry micro instruction micro program starts.

So, at third program at third microinstruction of this common routine which corresponds to jump on carry and jump on Z, the fetch is over which is common for everything at third instruction it will check the condition select 10; that means, it is going to check the IR line which is the third line and branch address I have kept it as 110. So, it is checking that if it is jump on 0. So, it will just increment. So, from third it will go to fourth memory location of the micro program, which will be corresponding to jump on Z.

But if the macro program that is the instruction decoder tells that the jump on carry is a jump on carry it will make this line has 1. So, if it is making this line as 1 it is going to load and it is going to load the address of 1100; that means, it will micro program corresponding to jump on carry is going to execute so, like that. In a simple thing that is going to be going to happen over here.

And here I am connecting the 0 flag output that is going to check. So, this again I am repeating that is the implicit jump instruction that has to give in because of 2 micro macro programs 2 macro instructions are clubbed together, that is jump on 0 which basically starts from 4 and assuming that jump on carry starts from 12. So, that is how it has to be settled. So, I am just going to look at again.

So, up to 3 as I told you it is fetch which corresponds to it is very similar to even jump on carry M. So, that will be similar. So, therefore I don't require to check anything on the condition select. So, I have made both of them 1. So, 11 means as you can see. So, if I make 11 means the last bit is selected which is 0 if the condition select is 00 then mux output is 0. So, it is always basically the increment mode. So, it will keep on incrementing. So, again and it will start coming at here. So, here it is very very important.

So, here the condition select as I told you is 10; so 10 means as I has already shown you. So, 10 means you are connecting to the third line here I have connected the IR output the instruction decoder output, which actually happens in this way that if it 0 and 1 because here we are making a simple assumption but because I am 2 micro programs common, but of course, if you have large number of micro programs clubbed together, then the mux will also be not be a 2:4 de multiplexer, it will be larger size of mux because corresponding to the instruction decoder output the number of input for the mux will rise, because in fact, there are 2 constants one is 1 and one is 0, one is for default load and 1 is for default increment, but the other inputs will be corresponding to different conditions basically.

So, here we have just assumed that the 2 instructions of there. So, we have a taken a 4:2 multiplexer, but. In fact, if the number of instructions which are combined and number of I/O signals are large. So, generally in a practical sense this mux is quite large in size and the condition select also is larger in width basically. So, for the time being I am just making simple assumption for you. So, if it is a jump on 0 the instruction register will give 0 to this and other wise instruction register will give 1 for this.

So, if it is a 1 basically if this is a 1 if this is a 1 as you can see. So, if this is a 1 it will correspond to basically load which is nothing but memory location number 12, micro program memory location number 12 which corresponds to jump on carry. And if it is jump on Z you are going to get a 0 over here which will load the next value. So, if you look at it. So, it is saying that the condition select is 10; that means, you are going to check IR output IR decoder output. And if it is a 0; that means, you are actually talking of this jump on 0. So, you have to just increment from 3 to 4, but it is a jump on carry from 3 you will jump to another location which is memory location number 12.

And giving assume assuming that basically 1 2 3 4 5 6 is common sorry 1 2 3 is common for both because 1 2 3 actually corresponds to jump on Z and jump on carry, but 4 5 6 is very much explicit or in fact, you can say 4 and 5 is very much explicit basically for jump on Z and maybe 11, 12 and 13 memory location will be fixed for jump on. In fact, I should say that not even that only this one only this instruction that is offset IR, $select$ 1, if 0 flag is not set only that part actually is different from for jump on carry and jump on Z.

In fact, here we are going to check the 0 flag and the other case maybe you have to just check the carry flag. So, that is the only different part. So, maybe memory location number 12 is the only part which is going to be different. So, if the deco instruction register is actually giving the value of 1 which corresponds with jump on carry. So, in this case it will jump to this memory location because the condition is 10. So, it is going to check if it is 10; that means, the third row of the multiplexer is selected and if it is a jump on carry, instruction register value will be 1 and it will jump from this 2 memory location number 12.

So, in the 12 we will have just this instruction which is number 4 over here, which is only difference only difference between jump on carry and jump on 0. So, that will be executed and after that you will again jump back and come over here because after that it is very similar again you have to do Z_{out}, PC_{in} and M. So, only one bit will be different in this case or only 1 memory word will be different which is the only difference between these 2 micro program that is jump on Z and jump on carry that different will be executed at 12 and again you have to come back.

So, they are be 2 implicit jumps. So, one jump will be jump from 3 to 12 depending on the type of micro instruction you are considering at the time and after executing 12, you are going to have a basically unconditional jump which will bring you back to 15.

So, basically this is the implicit jump instruction, which you have to put in because you are adding 2 macro micro programs corresponding to 2 macro program macro instructions. So, I have shown you up to 6 here maybe you can put dot memory location 12 will also have similar thing which will again correspond to basically the fourth instruction which is the difference. Now interesting now basically till now we have discussed basically the explicit implicit part of it, now this macro routine or the micro routine for this micro instruction is also a jump instruction.

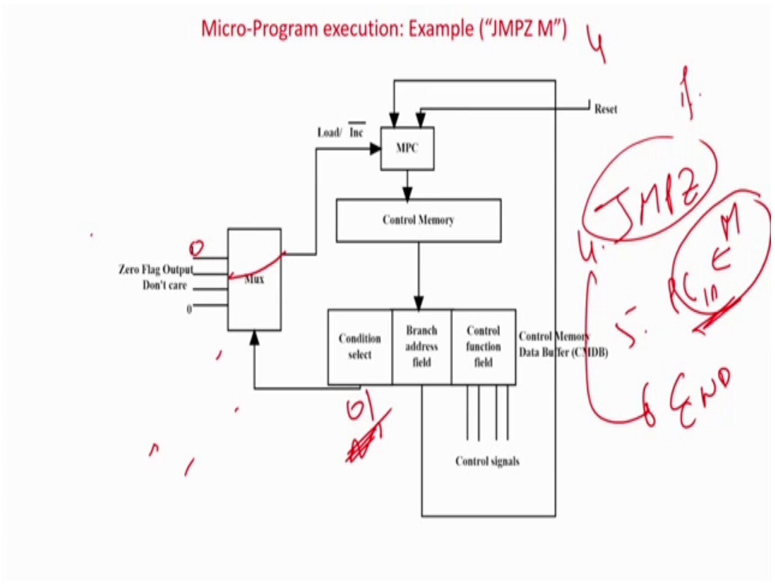
So, what it says that, if the 0 flag is not set then actually you have to go to M and if the 0 flag is set you have to just go and increment it because Z out, PC in basically is going to load the program counter with the macro instruction which is located in memory location end. So, this is actually nothing but a basically the jump which has to be explicitly taken in this macro instruction micro instruction. So, this is the explicit jump the micro routine should have corresponding to this macro instruction.

So, of course, again at 4 also there will be condition check, but other than you can see I have put it one. So, they do not correspond to any kind of jump they just correspond to increment, but here again 01. So, you should remember that 00 and 11 they correspond to default load and then they correspond 2 jump or not jump basically, but all other combinations basically like 01 and 10 in this case, they correspond to check of the basically condition and they will depend and they will need to jump on some address which is present over in the branch address. So, these 2 are very very important over here.

Because 11 corresponds to basically a 0 in the mux, which corresponds to just a default increment and basically 00 corresponds to default unconditional jump we have already seen in the figure basically again just revisiting. So, 00 means this one which actually corresponds to load if there is default jump and 11 in the condition actually which corresponds to the fourth line of the mux which is 0, corresponds to simple increment, but all other combination in this case 01 and 10 they correspond to branch.

Now, this is your implicit branch and the last one the third fourth instruction that is 01 which corresponds to the second line of the mux basically let me erase it, the second line of the mux, now your answer is basically 01. Now the condition select in this case is 01. So, 01 is basically connected to; that means, this line is connected to here. So, what it says it is jump on 0 to some memory location.

(Refer Slide Time: 60:09)

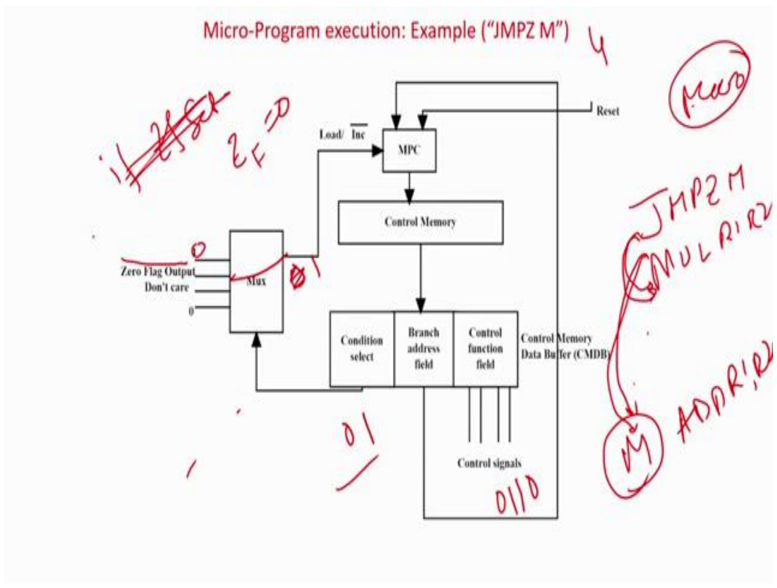


This is say fourth location, this is fifth location and this is sixth location. So, fifth location actually load the value of PC_in will be connected to the memory location M that is what is going to happen in PC and sixth location is M we have already seen in that program.

So, in that is in the fourth instruction you are going to check the 0 flag output, that is for that you are making condition select as 01 and if the 0 flag is set if the 0 flag is set in that case you have to basically jump; that means, in the macro program; that means, you have to load instruction number 5, but if the 0 flag is not set in that case you have to directly jump to end because in that case what is going to happen this thing will not happen and the next program in the macro routine will execute so; that means, what if the if 0 flag is not set; that means, if the 0 flag is not set let us look at the routine once again. So, if the 0 flag is not set then you are going to jump to end; that means, what? That means, they will be a jump if the 0 flag is not set.

So, in this case what it says you have to if the 0 flag is not set then you have to have a jump so in fact, that is why you have to make it a zero flag output bar that is the zero flag output inversion will have to collected over here because again let us look at this example slightly the more elaborate manner. So, what I have done, I have connect here the condition select is 01 in this case which is an explicit condition select.

(Refer Slide Time: 61:43)



Let us try to something assume something here, jump on Z to M ok and this M location may be having called $ADD\ R1\ and\ R2$ and the next it is the macro program and here you may have $MUL\ R1\ and\ R2$.

So, what it is saying? If it is jump on 0 remember in the macro program this the macro program this is the macro program if it is jump on 0 that is 0 flag is set then basically you have to load the program counter corresponding to this, and if the 0 flag is not set just the program counter will be incrementing that is at the macro program level now you see what happens in the micro program level. In the micro program level we will make the condition select as 01 because you are going to talk about the 0 flag, which is connected to the second bit of the mux. And here we are connecting zero flag output bar now why bar that will be made clear so; that means, if the 0 flag is set. So, you are going to get the answer as a 0 over here because if the 0 flag is set inversion means if the 0 flag is and you will get a 0 over here.

So, if the 0 flag is set over here then what you are going to do? If the 0 flag is set because you are going to get 0 which is going to be here and which is going be increment at the micro program level; so in the micro program level if you look at it so; that means, what just remember if the 0 flag is set if 0 flag set; that means, I am connecting 0 flag bar here. So, you are going to get a 0 over here and it is just going to increment this is at the micro program level. So, just see over here. So, what happens? If the 0 flag is set. So, if the 0 flag is set you are going to get a 0 in the micro program control it is just going to increment. So, you are just going to go to memory location number 5, so which says that $Z_{out} = PC_{in}$; that means, the program counter value will now have the value of Z, which means in the macro program level basically the program counter is now pointing to M.

Now, if the 0 flag is not set that is 0 flag is 0 that the value of 0 flag is now basically equal to 0. So, if this is the case, then you are going to have a 1 over here. If you have a 1 over here it is the condition select check it will be load and the load value here I have put 0110 that is 6 in the micro program level; that means, if the 0 flag is not set; that means, in that case is going to jump and where you are going to jump you are going to jump at memory location number 110, 110 that is 6 at the micro program level will now come to 6 which is actually end.

So, if it is end, we observe at the macro program level the PC is not updated. So, if the PC is not updated basically what happens you are going to go from jump on 0 to MUL. So, in that case in the macro program level the jump has not happened, slightly tricky. Whenever it’s a jump in a macro program level that is here the micro program does not jump basically it just act basically what happens, when the micro program oh sorry when the macro program jumps basically in the micro program you go from 4 to 5 which loads the PC with a new value and if the mac micro program jumps that is it comes from here to see that is end you are not actually updating the value of PC in that case the macro program does not jump.

So, slightly actually tricky once you look at it carefully it will you can understand, what basically is happening slightly tricky means slightly the program has been written in a turned about way, that is when the macro program has to explicitly jump, the micro program is not jumping and when the micro program is jumping basically the macro program is not jumping anyway that you can look at it, but just look at the philosophy that is very important.