So, what happens in the fourth routine sorry in the fourth case which is corresponding to the explicit jump, which has to do because of the macro instruction that is the macro program the jump we are looking at it. So, this instruction was the implicit jump because of the common micro programs corresponding to different macro instructions, that was the was happening because of optimization, but in this case this is happening because of basically this is explicitly required the fourth one the fourth jump is mandatory, without this your program will not run correctly, but the jump at the third location actually corresponds to optimization.

So, what happened in fourth again I am repeating because this is slightly tricky, we are giving 01; that means, you are checking the code corresponding to the second word second line of the mux. The second line of the mux is connected we are connecting it to output of the 0 flag bar. So, if the 0 flag is there, 0 flag bar we are connecting over here there and in fact, what happens if the 0 flag is not set if the 0 flag is not set; that means, $\overline{ZF}$ is going to be 1; that means, in this case you have to load the address; that means, there is a jump in the micro instruction program.

So, in this case where the jump will do? Jump will go to 6 that is going to be end and in that case what happens in the macro program in the micro program there is a jump, but in the macro program what happens basically? This updating is not reflected over there. So, the macro program will not jump it will just execute the next instruction, but if the 0 flag is set. So, we are connecting 0 flag bar there. So, if the 0 flag is set, you are going to get the 0 in the mux output.

So, 0 in the mux output means there will be no jump in the micro program that is it will not jump to 6, but you will just increment and go to 5. So, in go to 5 what happens? Z_{out} is PC_{in}. So, in this micro instruction basically loads the program counter that is the macro program counter with the new *PC* value that which is actually $offset + Y\_in$ and actually it corresponds to jump in the macro program. So, if this is the *PC* in is being updated. So, it will jump and basically it corresponds to jump to the memory address instruction where which M stores. So, there will be jump in the macro level and after that the micro routine will end and then what happens basically? The next *MPC* that is micro-program program counter in case there is a jump in the macro routine, it will correspond to basically this part. So, if there is a jump in the macro instruction macro routine.

So, the *MPC* will correspond to basically the micro program corresponding to add and in fact, if basically there is the 0 flag is not set basically. So, in this case this macro routine macro

program instruction will execute. So, the *MPC* will start pointing to basically the micro routine corresponding to $MUL\ R1\ R2$, But if you think about optimization, so we will have a single micro routine for both MUL and add and in fact, the diversification will happen only when you come to instruction number 4 which will tell whether it's a MUL or an ADD.

But in fact, this is actually shows you how basically a total routine executes for a given macro instruction like jump, and also we have explicitly shown there can also be jumps based on if you have a single micro routine for similar type of instructions like here we had a single routine for jump on 0 and jump on carry. We are assuming that the different instruction is placed in the memory location number 12 which responds to jump on carry. So, if this is the case there will be a implicit jump from memory location 3 to 12 and again it will come back to routine number or location number 5 and it will execute 5 and 6 which is common to both.

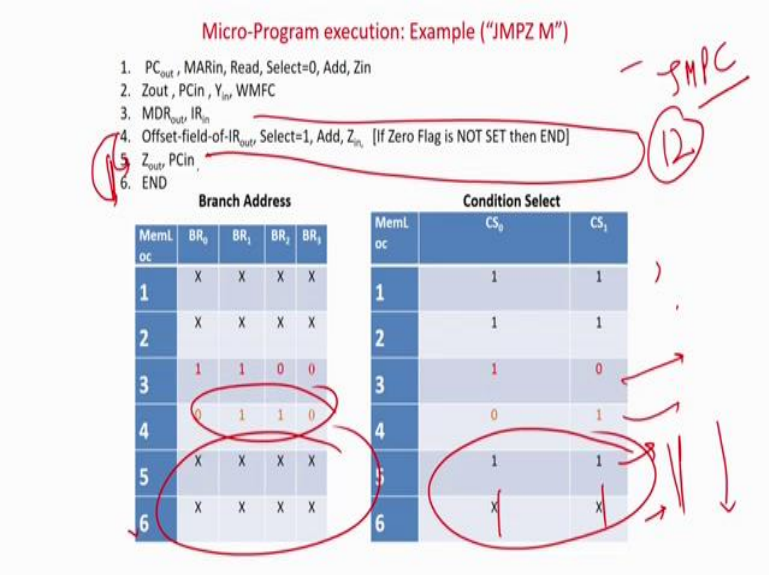
This is an implicit one and this is there for optimization, but the other one it explicit it has to be there for the correct operation of this macro instruction jump on M, in that case what happens? You use the 01 condition which checks 0 output 0 flag bar. So, if the flag is set you basically if the 0th flag is set then what happens basically? You directly execute this one so if the 0 flag is set you execute this which loads the new value of updated value of $PC + offset$ that is your M into *PC* and in the macro routine there is a jump otherwise you directly come to end and that is basically no jump in the macro routine.

So, this is how it happens basically. So, you can see this one implicit this one explicit and this is the address you have to jump in this case if it is true otherwise it is similar. So, this one just you have to I am erasing out if you just look at it in your easy pace and you have to try to solve it, then you can easily find out what happens. So, slightly roundabout just you have to check. So, and then you can see the other 2 parts at from 5 and 6 basically there is no jump and no com there is nothing difference in between this micro routine these 2 routines are similar for the 2 macro instructions we are considering. So, there is no requirement of any kind of jumps over here. So, you can see that I have put 11 or in it's end is 6 so, no there is no questions I have put xx or you can again put 11. So, basically there is no any kind of jump for that. So, it will be single smooth increment of the micro program counter it will go through and that is basically going to happen.

So, this is very very important. So, just go through this example in a very nice and a very slow manner write down yourself on your notebook and see how it executes and you all things will

be clear to you. So, only again just repeating only one thing you have to keep in mind the third one is for your implicit jump, fourth one is for explicit jump and we are assuming that from memory location number 12 the jump on carry instruction the explicit different micro instruction involving that will be placed over there.

(Refer Slide Time: 70:59)



So, in case there will be a jump from 12 and again it will come back to 5.

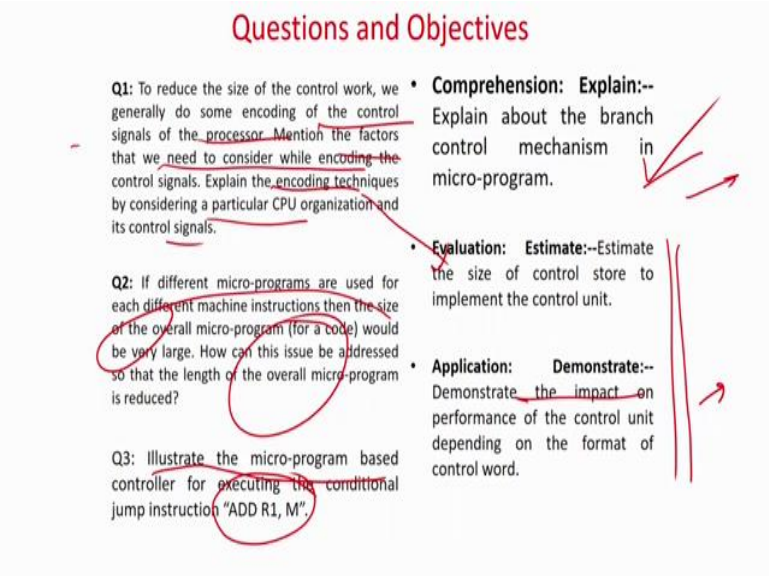
Because 1 2 3 5 and 6 are common for both and in the last one this is an explicit jump which is jump corresponding to jump on 0. And again slightly tricky because when there is a jump from here to here there is no jump in the macro instruction macro program, but when there is no jump in the micro routine for 4 to 5 there is a jump corresponding to the macro routine slightly tricky you solve it and you are going to understand.

So, in a nutshell what we have seen in this? We have seen that how micro programs can be optimized 1 is by compressing the by encoding and compressing the control bits and one way by actually merging most of the common macro instructions the micro programs are merged. So, if you merge it there will be lots of jumps here and there implicitly required, because the common parts will be written only once and for the different part of the different macro instructions you have to jump here and there and come back and also there will be some explicit jump instructions like jump on Z, jump on carry etcetera, for which there will be explicit jumps.

So, in that case obviously, the micro routine for jump instruction will carry out those explicit jumps and this is an example by which I have shown how a complete micro routine is executed for a macro-macro instruction and also how if there are 2 macro instructions for which there is a single micro routine how implicitly is they will be handled.

So, this is a slightly complicated example, which actually involves both the type of jumps in a single micro routine.

(Refer Slide Time: 72:37)



So before we come to this unit let us see the some 1 or 2 small questions, which will see that how we have met the objectives. So, we say that to reduce the control word we generally do some encoding of the control signal. Mention the factors we need to do the encoding that is clustering etcetera explain the encoding techniques in a particular CPU organization, single bus architecture and its control signals.

So, basically it will easily satisfy those objective estimate the size of control store to implement the control unit. Demonstrate the impact of performance of control unit depending on the format of control word. So, these two objectives will be satisfied once you at the enable to solve this problem. In fact, you should be able to do it because you have taken explicit example, to show how it can be optimized both at this level as well as this level. This question, question number 1 mainly tries to optimize the word sizes based on encoding. So, we will be able to do so, do so, because we have given some practical example and we will be able to satisfy these 2.

If different micro routines are used for each different instruction then the overall would be very large how this can be addressed? that is by merging at this level though this also we have explicitly addressed and I have given you a simple example although not very elaborately explained, then I have taken one jump on 0 and jump on carry, similarly you can take very similar instructions like add, sub, multiply and try to write a common micro routine for that and then see how it how it basically optimizes.

So, in that case you are going to also have to basically satisfy this objective because there will be lot of implicit jumps because of this merging of large number of micro routines for different macro program macro instructions. So, not only you will satisfy this, also you will satisfy the objective because you have to know about basically the branch control. And finally, I have asked you to write a micro program control for executing the instruction $ADD\ R1, M$.

So, of course, this one will basically requires this 2 objectives because I would request you to write this $ADD\ R1, M$ and try to optimize the signals in the horizontal level and also try to add another instruction like $subtract\ R1, M$ and try to optimize the signals at this level and then try to see the impact of both how many branches you require and also how many you save it both this have action both the horizontal level as well as vertical level. Horizontal means you reduce the word memory micro program control memory word size and in this case you reduce the number of instructions by merging the common micro routines for different macro your macro instructions right.

So, with this we come to end of this unit and the next unit basically we will be trying to focus we will just give you an idea of the whole you whole different units or different lectures we have done in this module, we basically have given an undertaking or assume that everything in a single bus architecture. Next will next unit will be dedicated we will try to show how things change if there are more number of buses 3 buses, 4 buses or a multi bus architecture.

So, next will not deal in details that how your micro program will change, how your finite state machine based implementation will change if there are multiple buses you will be able to do it that will be very simple once we completely the discussion on the multi bus architecture. So, in multi bus architectures you will find out how these micro control signals will change, what are the difference if you are having a 2 bus or a 3 bus architecture, 1 bus or a 3 bus architecture, and accordingly you will be able to adopt those concepts to develop micro program control routine as well as your hardwired that is finite state machine based program.

So, next unit will be dedicated on looking at how things are going to change if we are having a multi bus architecture. Those simple idea will be will be will enable you to take whatever we have learned till now from single bus architecture to multi bus architecture.

Thank you.