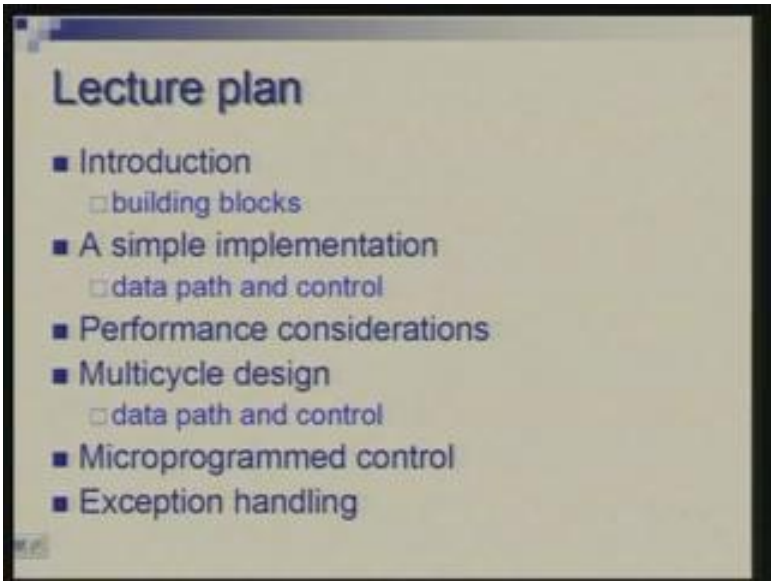
Computer Architecture
Prof. Anshul Kumar
Department of Computer Science and Engineering
Indian Institute of Technology, Delhi
Lecture - 22
Processor Design – Microprogrammed Control

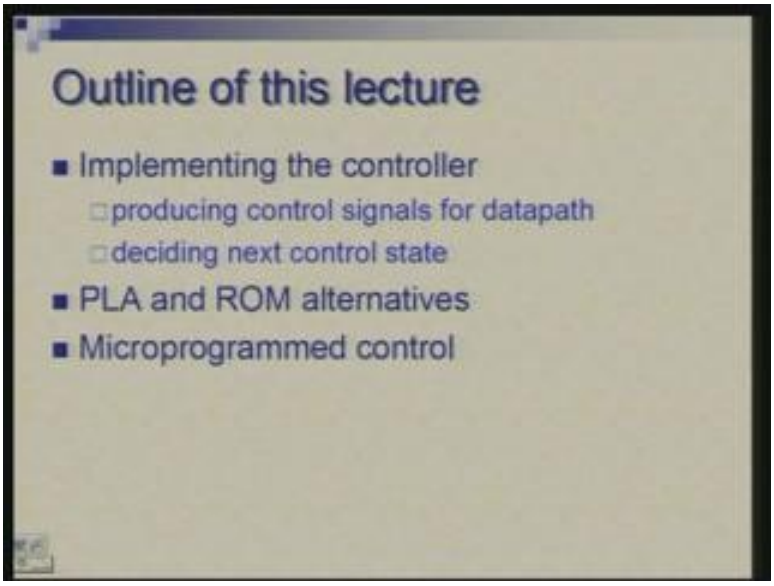
In the previous lecture I started design of controller for the multi cycle data path for MIPS processor. We will complete that design and I will also take another style of design called microprogrammed design.

(Refer Slide Time: 00:01:11)



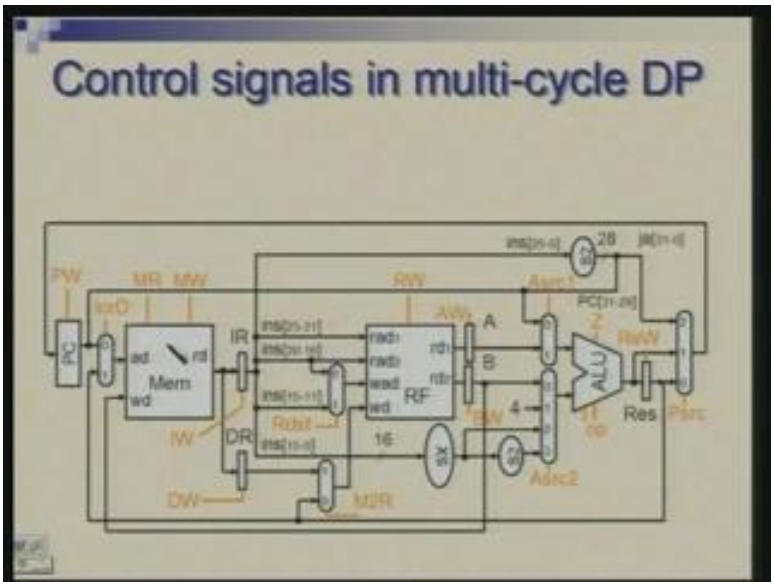
In the overall scheme we are here; we are going to concentrate on microprogrammed control and also finish this control part design.

(Refer Slide Time: 00:01:25)



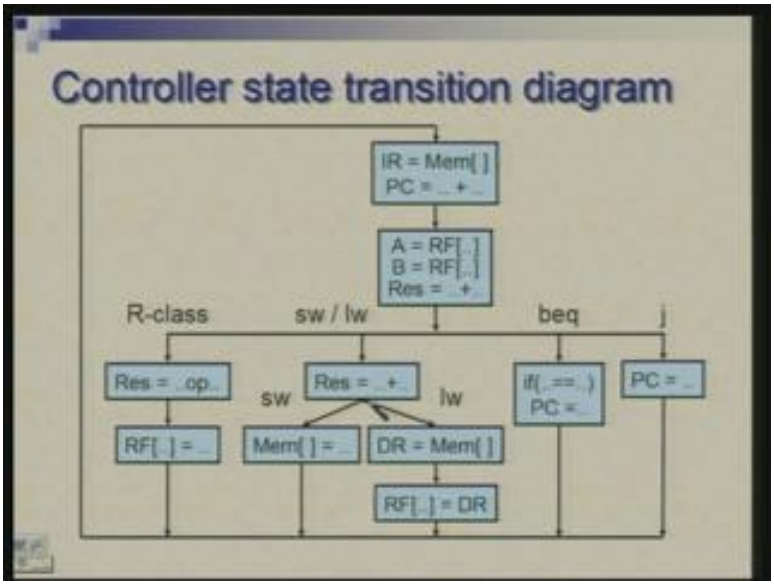
We will also look at two different alternatives of PLA and ROM. Sso we will look at the implementation of controller which has two parts: one part produces the control signals; other part decides the next control state and each of these could be either implemented by a PLA or a ROM as we will discuss later and then finally we will talk about microprogrammed control design.

(Refer Slide Time: 00:01:54)



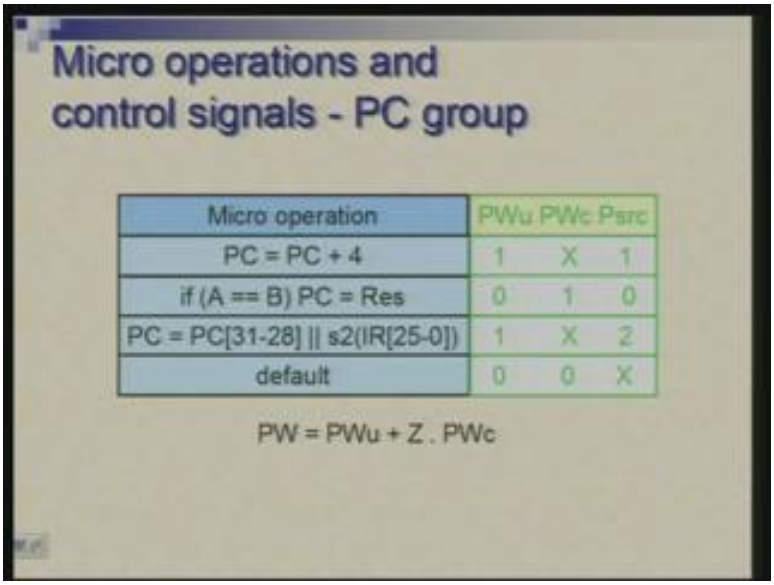
So this was the data path we arrived at for multi cycle approach. The picture shows all the control signals which are useful for controlling various components.

(Refer Slide Time: 00:02:12)



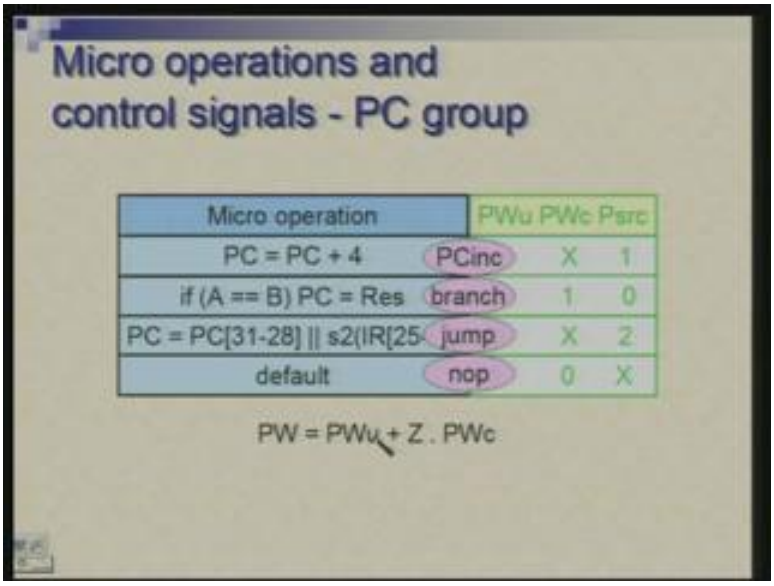
And this is the overall flow chart indicating how control state changes from cycle to cycle and what action is taken in each control state.

(Refer Slide Time: 00:02:22)

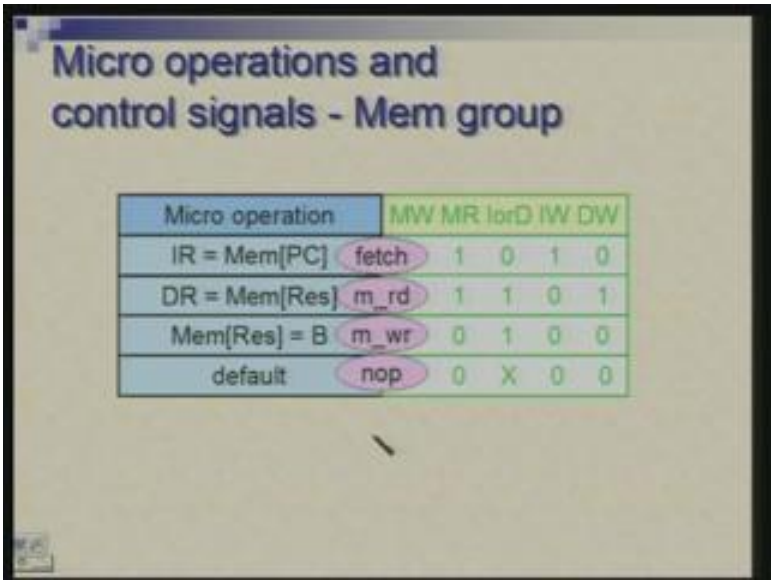


We have identified groups of control signals which need to be controlled together to do some basic operation which we call as micro operation. So all micro operations were identified and the pattern of control signals required to make them effective have also been identified; some symbolic names were assigned to these micro operations.

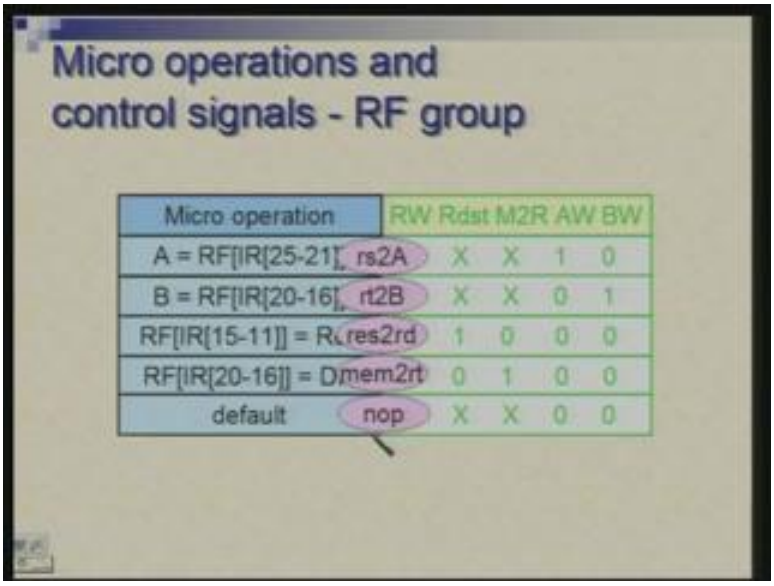
(Refer Slide Time: 00:02:43)



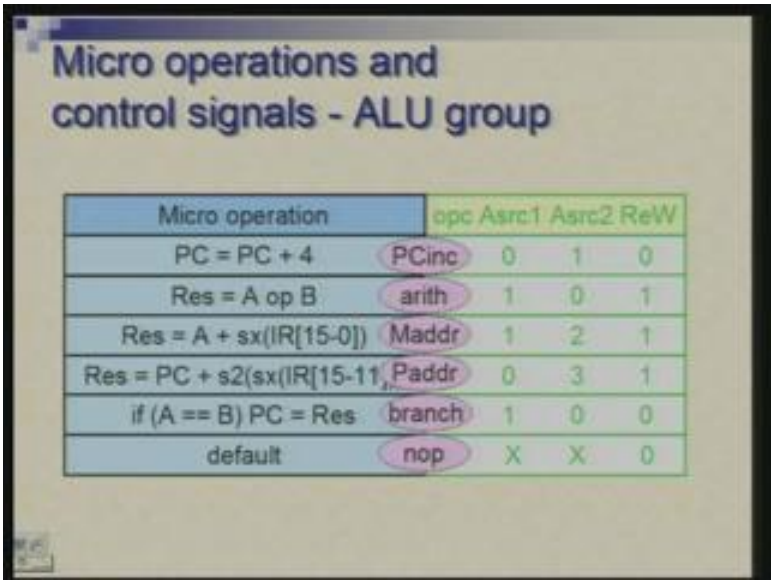
(Refer Slide Time: 00:02:46)



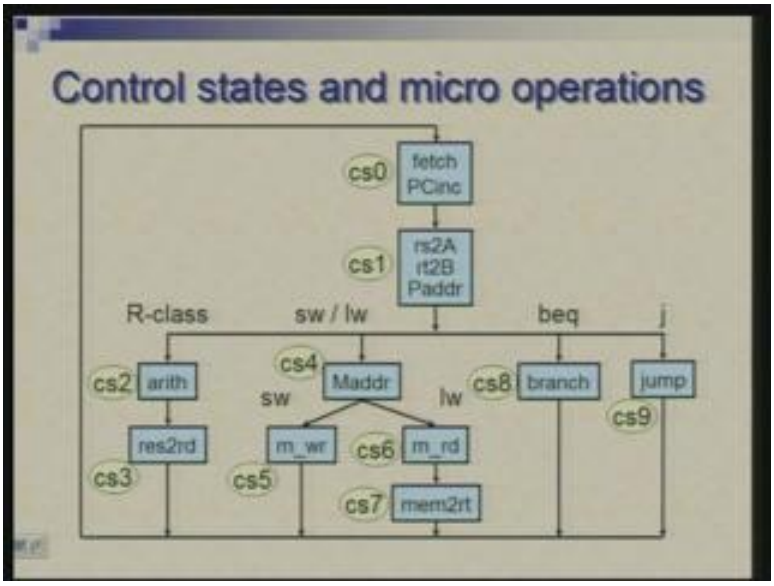
(Refer Slide Time: 00:02:47)



(Refer Slide Time: 2:49)

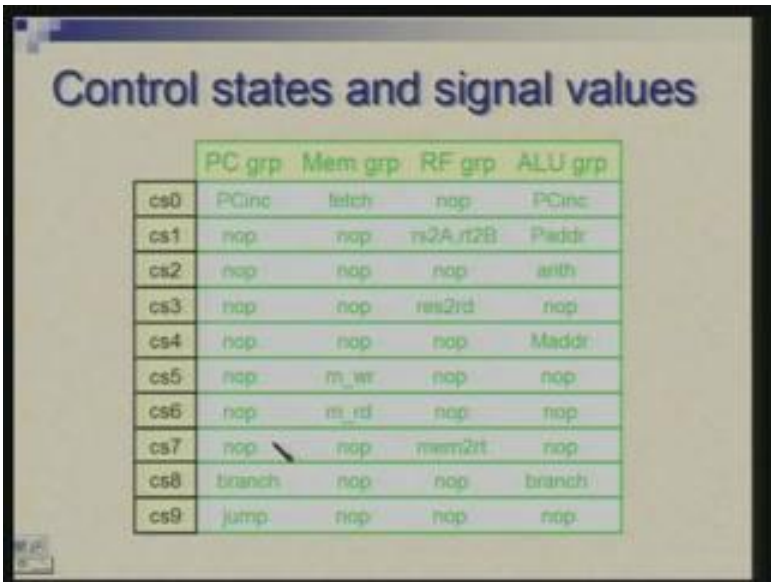


(Refer Slide Time: 00:02:50)



And then in brief we replaced all the assignments and all the transfers by names of these micro operations and also we labeled all the states. So now this is the most crucial part of the design, after this the implementation starts.

(Refer Slide Time: 00:03:09)

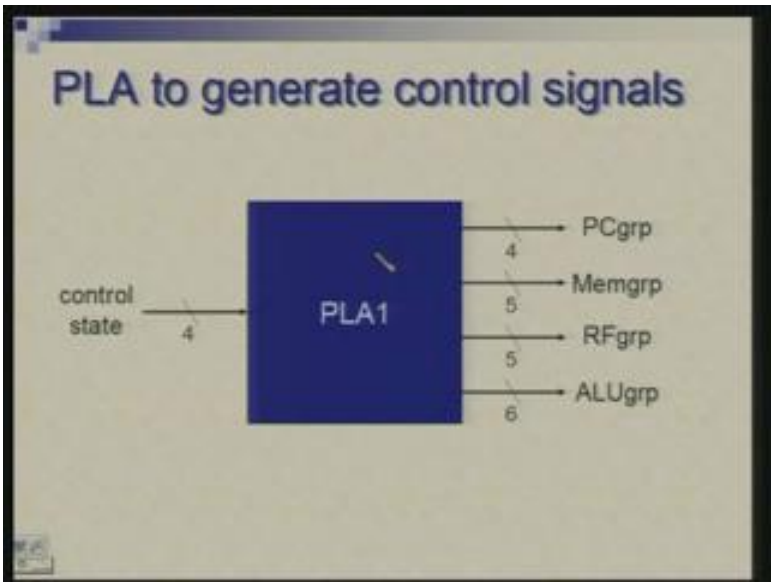


So we tried to capture all this by two tables. One table describes the control signals for each control state. So, for different groups of control signals we are indicating what micro operation they are supposed to perform and each of these corresponds to some pattern of control signals.

So, if you substitute that; if you replace each micro operation by the pattern of control signals from one of these tables (Refer Slide Time: 00:03:42) what we get is each micro operation has been replaced by a pattern of 1s and 0s and also there are Xs indicating don't care wherever appropriate.

So now this basically is like a truth table description and all that is required is that we encode these control states also in binary form which you see here that each state can be encoded in 4 bits since there are ten states; this forms the truth table where the part shown in black is the input and part shown in green is the output. So basically it is a 4 input and this I think numbers upto 20 so 20 output and four input is what this table describes.

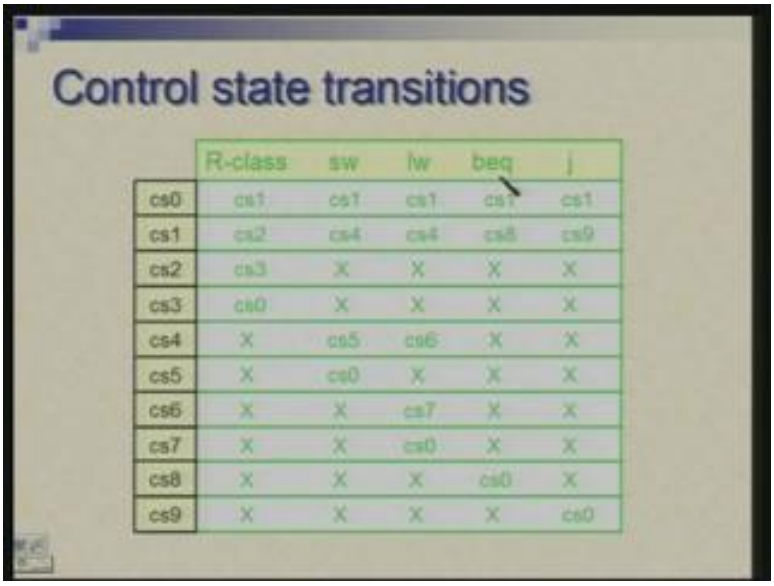
(Refer Slide Time: 00:04:35)



So this can be implemented by a PLA as we have discussed earlier for single cycle design. This PLA will have 4 input coming from the control state and four groups of control signals; group of 4, group of 5, another 5 and 6 so altogether twenty signals which go to the data path.

The other table is what describes the next state given the present state and which instruction or instruction group we are talking of.

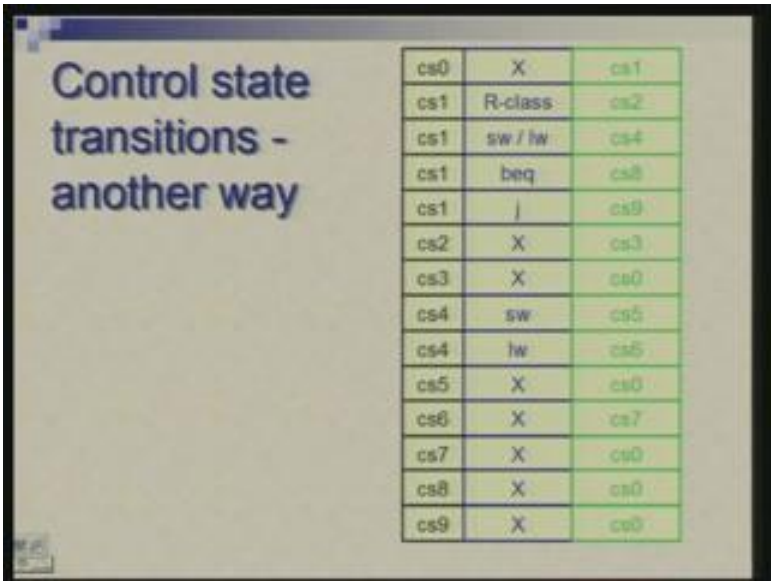
(Refer Slide Time: 5:10)



So we have the same ten states and instruction has been grouped as usual; R class sw lw beq and j and each row and column combination we are describing what is the next state. So these two tables together (Refer Slide Time: 5:31) are basically capturing this flowchart or state transition diagram. This table can also be replaced by a binary equivalent where we substitute code for each of the control state and also we specify the opcode value for various instructions. So, after having replaced the control state codes in this table we get this. On this column also we replace; all the states are replaced by their codes and opcodes also are put there.

So now it describes the truth table with a group of 4 inputs here and group of 6 inputs there. This table is defining 4 outputs and there are 4 plus 6 10 inputs. So we can have it implement it by a PLA but before that I am showing another way of looking at it.

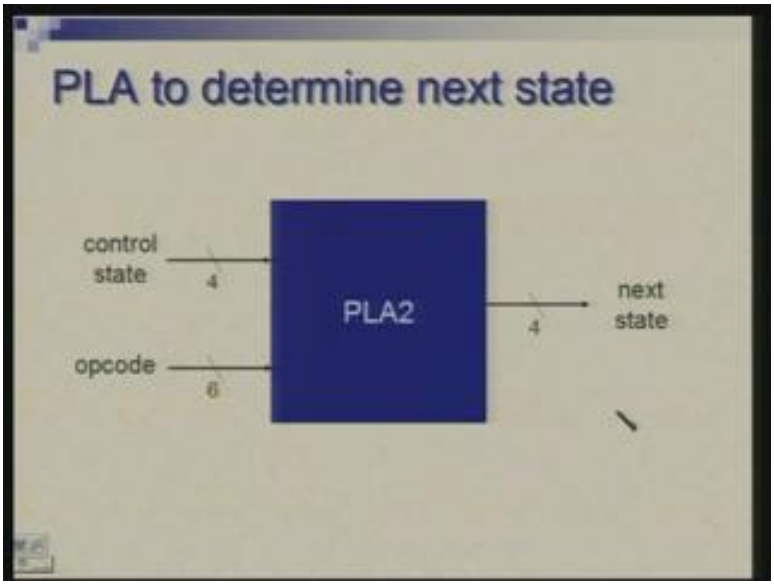
(Refer Slide Time: 00:06:41)



So here this is a more compact representation because here you notice lot of sparsity so we can, instead of having a 2D table I am showing it as the one dimensional table where all the input combinations are listed vertically. So it has the current state and instruction combination but we do not have 2 raised to the power 10 entries here although there are 10 bits of input because there are lots of Xs or don't cares.

So all the relevant combinations are captured. So, for cs0 irrespective of what when the other input is the next state is cs1 and so on for all these the next is a fixed state. So same table is rewritten in a different form and once again you can replace the state numbers state labels by their code to get this. Now you can look at it in a more conventional form; a truth table where you have first two columns represent the input put together a 10-bit input and 4-bit output.

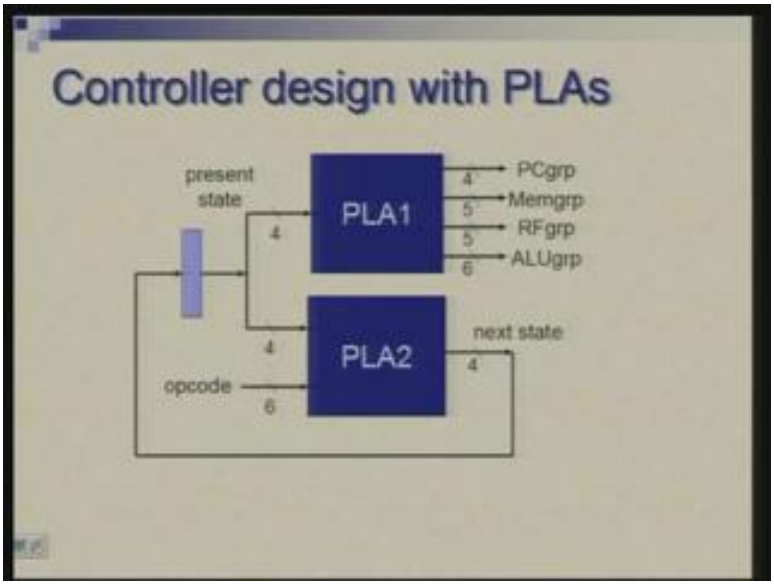
(Refer Slide Time: 00:07:44)



So here is the PLA which will implement these: 4 bits of control state and 6 bits of opcode to produce the next state which has 4 bits.

Now, what is the overall picture of the controller. We have these two PLAs implementing those two specific tables; all put together we also need register which holds the state value and at every clock this value will change as per these tables.

(Refer Slide Time: 00:08:17)

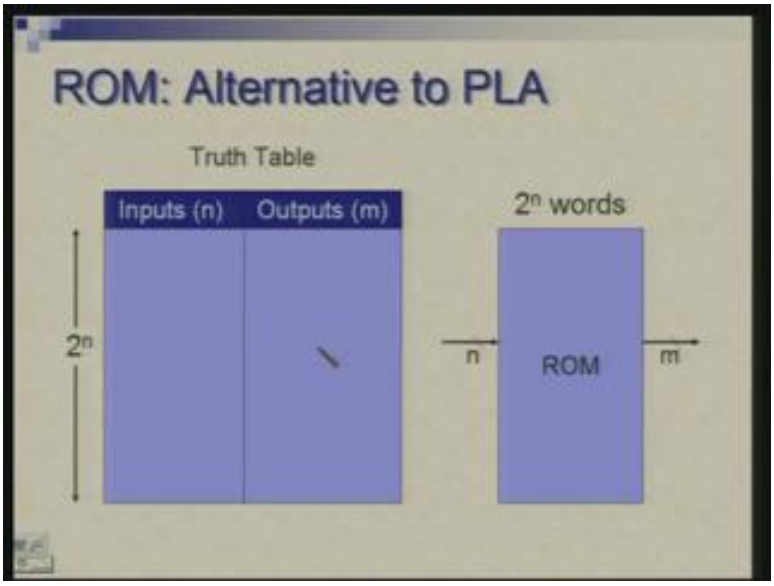


So here is the state register (Refer Slide Time: 8:18) it is a 4-bit register which contains the current control state and this drives both the PLAs; this PLA is generating the control

signals going to the data path and this PLA is also looking at the opcode and deciding what is the next state

So one could have thought of this as a single combination circuit with total of 10 inputs and 24 outputs but that is possible and it will be correct but that will be much more complex than these two smaller PLAs put together.

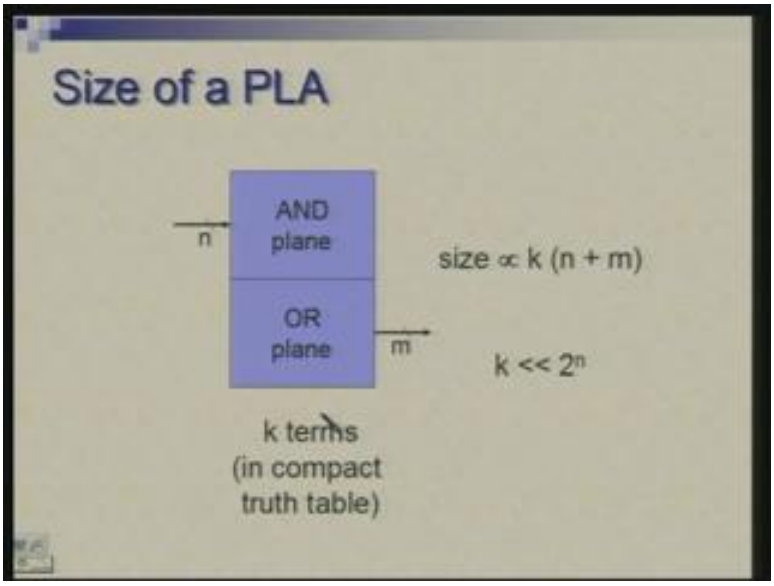
(Refer Slide Time: 00:08:58)



So at this point I want to bring in another alternative to PLA; again another general purpose component which can be derived directly from a truth table. So, suppose you have a truth table a fully expanded truth table with n inputs corresponding to 2 raised to the power n rows and defines m outputs this can be implemented essentially by putting this pattern of outputs in a memory. So in particular we are talking about read only memory the kind of memory we use for instructions where you are not modifying the contents you give an address as input and out comes the data. So these n inputs which you want to apply to the combinational circuit you are designing you apply as address to the ROM and you read out contents of the addressed word so it effectively works out as an n input and m output combinational circuit.

That definition of the function which this implements is directly given by the output column of the truth table. So each row of the output part corresponds to one word in this memory. Now, how do this alternative is compared with the PLA?

(Refer Slide Time: 00:10:24)

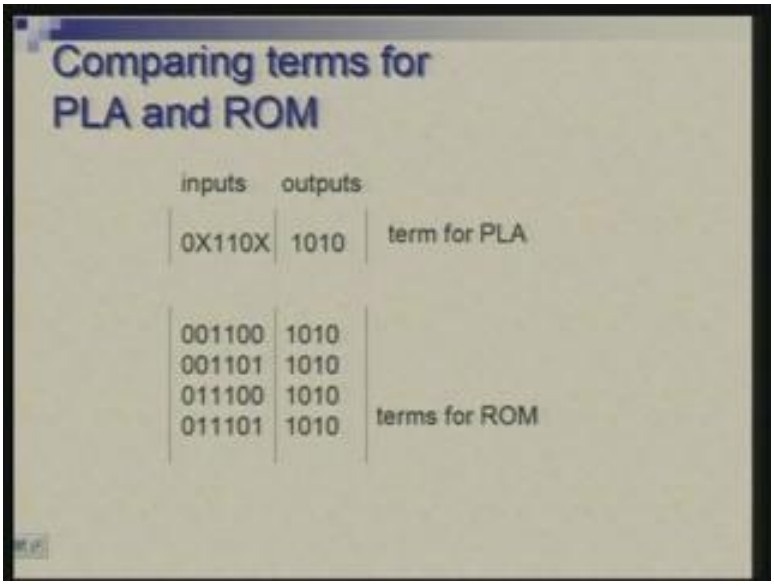


The difference is in terms of size whereas the memory like this will have 2 raised to the power n words each of m bits so total number of bits is m multiplied by 2 raised to the power n. On the other hand, a PLA which is implementing an n input m output circuit will have an AND plane and OR plane so there will be rows running for all the inputs, rows corresponding to true logic and false logic; there will be vertical lines which will implement AND gates and there will be rows here one corresponding to each output which will form the OR function of some of the product terms.

So what is the size; what governs the size of this…….?
Size is governed by these three factors: one is n, other is m and the third is the number of columns number of vertical lines you have to run each vertical line here corresponds to a product term or you can alternatively say a row of the truth table. So it corresponds to…. suppose there are k term in the truth table but remember that the truth table here could be not in a fully expanded form it could be in a compact form with lots of don't cares. So if there are k terms then the total size would be which you can see as roughly area of this rectangle will be proportional to k, that is this dimension (Refer Slide Time: 11:58) multiplied by n plus m because this n plane accommodates rows corresponding to n inputs and the OR plane corresponds to m rows corresponding to the output. So the height of this would be proportional to n plus m and the width proportional to k.

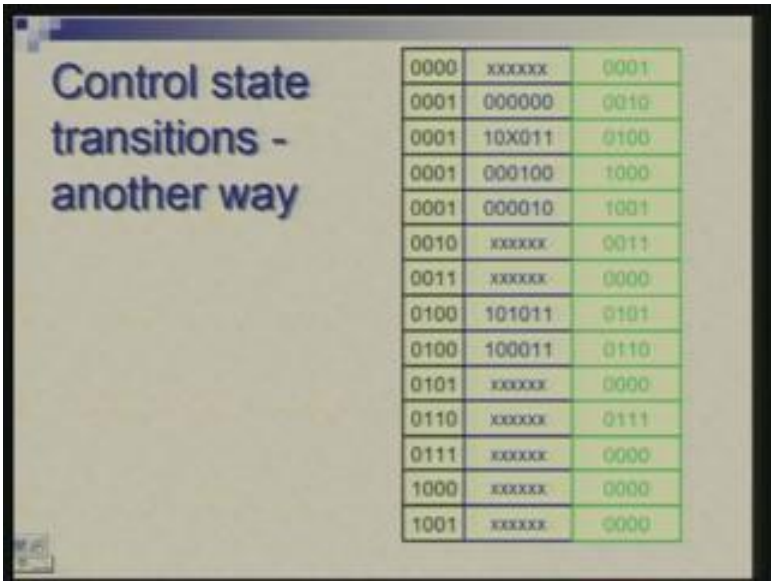
In general, k is likely to be much much smaller than 2 raised to the power n and that is what makes a PLA much more economic and compact as compared to ROM. So, if a truth table has lot of sparsity there are lots of don't cares and you can compact the whole thing in the form of PLA otherwise ROM is a very reasonable alternative.

(Refer Slide Time: 00:12:48)



So, to illustrate this point of compact representation suppose in your truth table there was a term like this 0X110X and for this the output is 1010 so in PLA you can take it as it will contribute 1 to that k whereas for a ROM you need to define output for each combination exhaustively. So such a term will actually show up as four terms where you take all possible value for these Xs you substitute 00 011 011 you get four possibilities and for each of these you need to ensure that the same output is there because with any input the output must be defined here. All we are saying here is that four terms are grouped here so irrespective of the value of the second input and the last input if this is the pattern 0 110 here (Refer Slide Time: 13:53) then this should be the output. So, in a fully expanded truth table this will correspond to four terms and this is just one simple illustration; on the whole you could see an example of a……

(Refer Slide Time: 00:14:07)

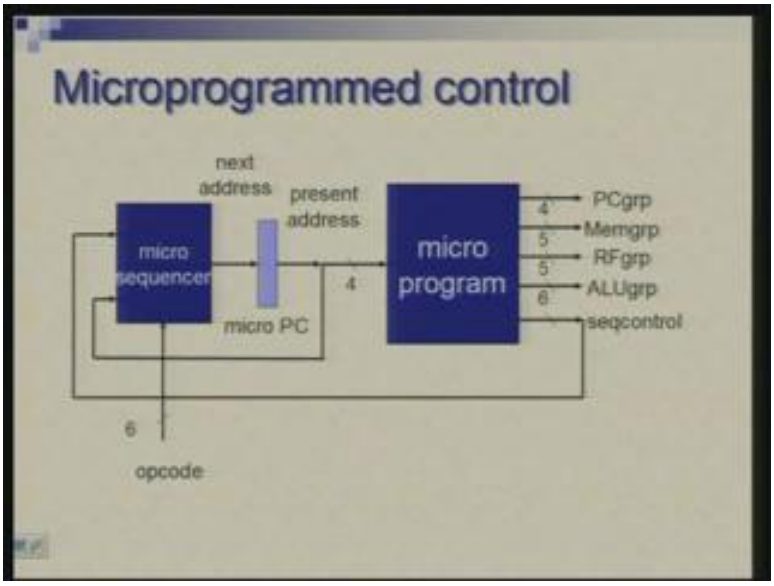


Now, let us say if you were to implement this by a ROM this will require 4 plus 6 which is 10 inputs that means 2 raised to the power 10 or 1024 words in the memory; each word will be 4 bits and total number of words will be 1024. But in PLA you will have 1 2 3 4 5 6 7 8 9 10 11 12 13 14 terms so k will be 14 here. Each of these will correspond to product of. For example, what you are saying here is that if this is 000110 irrespective of any value of this and 011 here that should be the output.

So a term like this (Refer Slide Time: 14:56) expands to two terms in case of a ROM; things like this will expand into 2 raised to the power 6 or 64 terms and so on so that is why PLA representation will be much more compact here.

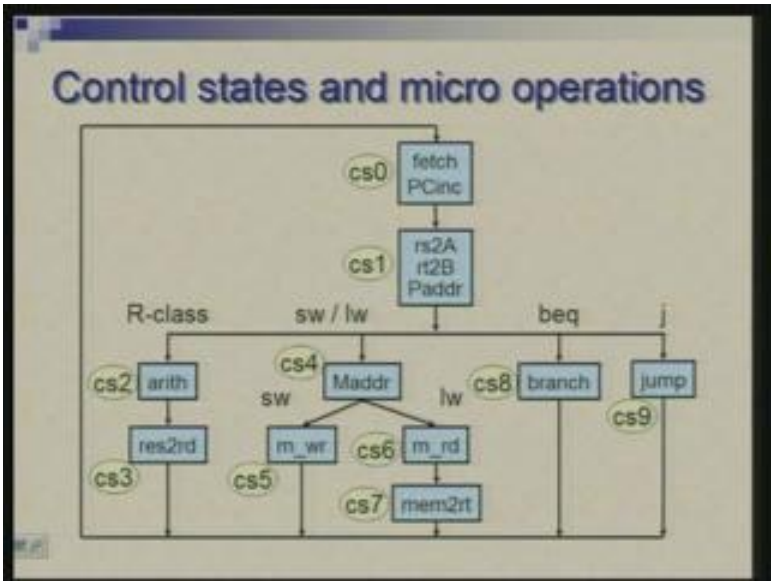
Now, what is this other style of control design which I mentioned; it is Microprogrammed control.

(Refer Slide Time: 00:15:21)



Here we try to view the controller as something which is a small program; something like a small computer trying to control the data path.

(Refer Slide Time: 00:15:38)



If you look at this, for example, you could think of this as a program flowchart. If you take that view why not think of a controller as a small computer which executes a simple program that program has the basic operations which are micro operations and all it does is generates control signals for the data path. This is a program which does nothing, else it does not deal with manipulating data in the main memory, it does not deal with register file and so on; all it does is that it goes through steps each step involves generating some

control signals for the data path. So with that view we think of a memory here containing microprogram and all we are doing is we are reading out words of this memory.

So you might think that now I am again talking of memory although I have mentioned some disadvantages of a memory based design but there are differences you will notice. So, imagine a memory which contains a microprogram. what this microprogram is nothing but each word contains a bit pattern which needs to be applied as control signals to the data path so those signals are not generated by circuit they are simply read out from this memory. You step through different words and each word generates appropriate control signals for the data path. So the word will have 20 bits plus a few more bits that I am going to describe in a moment. So these 20 bits will simply go as inputs to the data path and control it as required.

Now the question which now remains is that how do we sequence through different words of this microprogram. So sometimes we will need to go sequentially word after another and sometimes we need to mainly jump from one point to the other point. So here is a little arrangement to make sure that the right address is presented to this memory so that the correct word is read out in every clock cycle and how we generate this address is by another box which we are calling as micro sequencer.

So, micro sequencer is ensuring that right address is put in this register which we are calling as micro PC or microprogrammed counter so it is the counterpart of PC as we saw in the main program. In main program the PC is stepping through memory locations which contain the instructions; here it is micro PC which is just a 4-bit register it steps through different words of this microprogrammed memory to ensure that the right signals are generated at the right time.

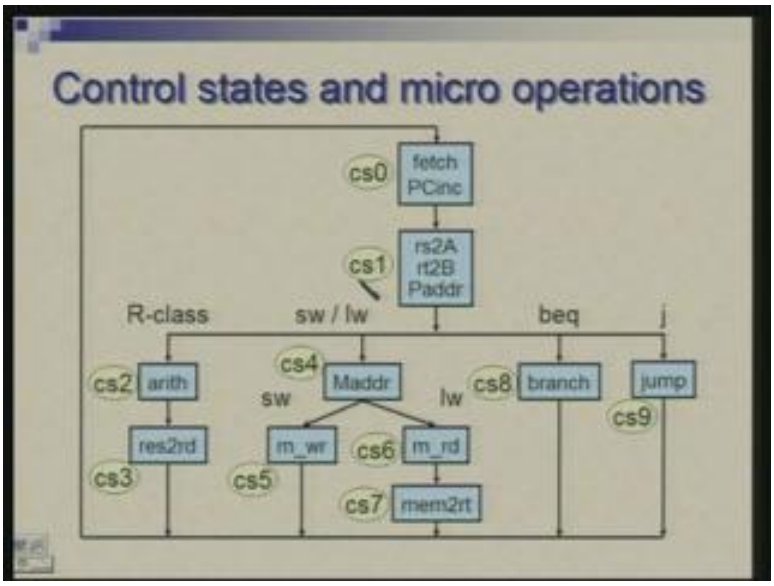
To ensure that this works correctly each micro instruction will say something about where the next micro instruction is. So each micro instruction may say that either we follow sequentially in the address or we branch to something else. So, that information somehow will be communicated by a few bits of the instruction and this micro sequencer will look at these bits, it will look at the current value of the microprogrammed counter and if necessary look at some signals coming from the data path so in this case it is the opcode bits.

Now you might be wondering that we are now talking of a box which has these 4 inputs plus these 6 inputs 10 and a few more here so isn't this going to be very complex? The answer is no because you would notice that quite often you would simply go to the next instruction or the next micro instruction or the next word. So it is only in those cases where we are not doing so the logic has to do something else but otherwise a simple incrementor adder will take care of bulk of the task and we need some more logic to take care of those conditions where you are branching off.

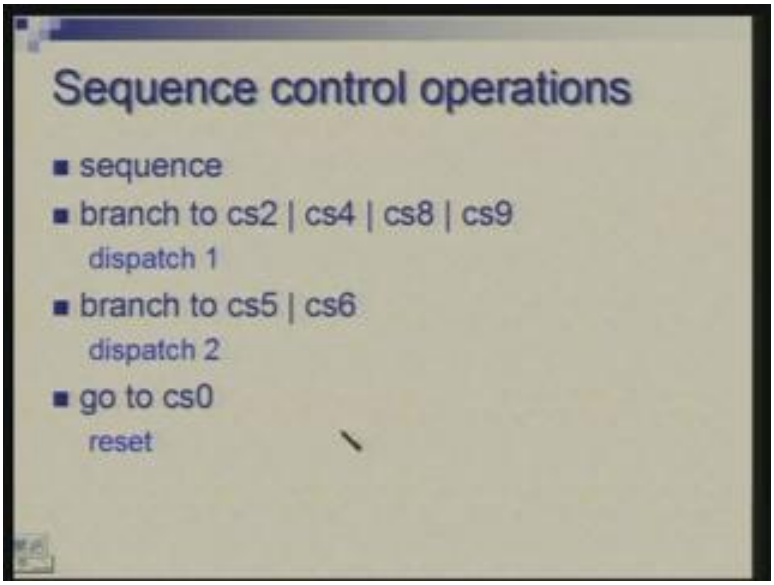
So, in our flowchart what all was happening let us see. We are going sequentially for example here, here, here, here, (Refer Slide Time: 20:31) the points where we are branching is here; this is one point we are branching, another point we are branching and

these are few points where we are restarting the whole thing. So we need to take care of these.

(Refer Slide Time: 20:44)



(Refer Slide Time: 00:20:59)

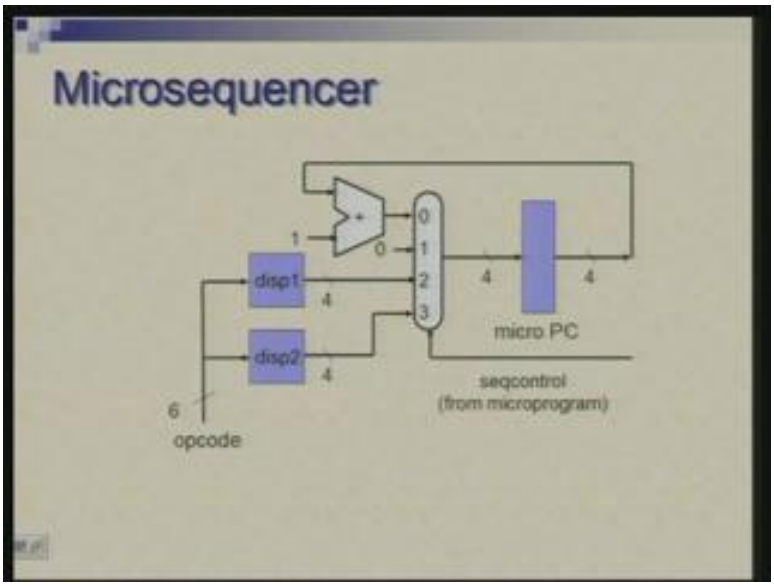


These are the four possibilities that you follow in simple sequence. In one of the steps we are branching to cs2 or cs4 or cs8 or cs9 that is one branch point, we call it dispatch 1. So in microprogram terminology a multi way branch like this is called dispatch. There is another point where we are branching to either cs5 or cs6 let us call that dispatch 2 and there are several states where we are going to cs0 so let us call that reset.

Now, that few bits in the instruction which I was talking of here, it needs to specify one of these possibilities (Refer Slide Time: 21:42). So the possibilities are sequence, dispatch 1, dispatch 2 and reset. These are the four possibilities we have identified; this is a specific case.

In general, when you are talking about more complex design these possibilities could be larger. But in our case a 2-bit field here will be sufficient to tell what we want to do as far as determining the next address is concerned.

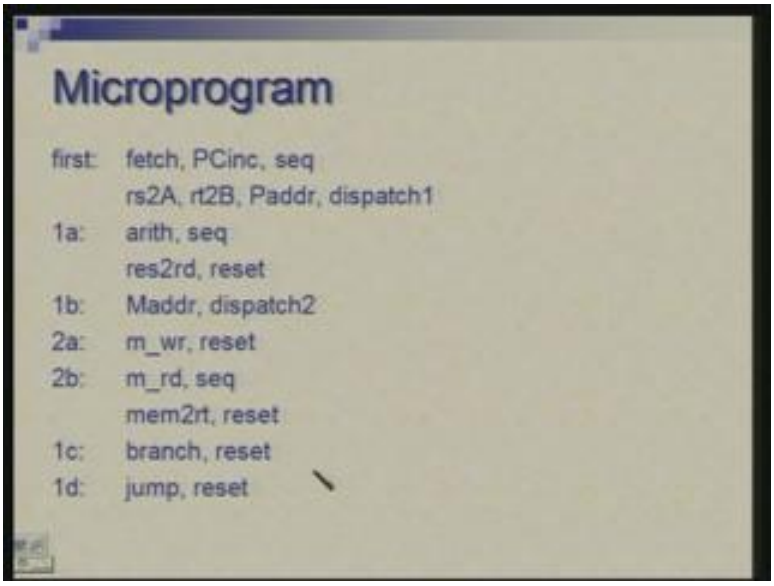
(Refer Slide Time: 00:22:17)



So, the micro sequencer needs to work with these four possibilities and it will be selected by the 2 bits coming from the microprogram which we call as sequence control. So this will select either current mu PC microprogrammed counter plus 1 or 0 or an address which we have called as dispatch 1 or dispatch 2. So these are small boxes which are looking at the opcode and generating the correct address. So once again you can think of these smaller tables depending upon the opcode value you want to pick up one of those four possible addresses. Here (Refer Slide Time: 23:02) depending upon opcode you want to pick up one of the two addresses so these could be small ROMs or small PLAs so we leave that here.

The main point is that there are a very small number of ways in which you need to determine the next address and a few bits in the microprogram will control this multiplexor.

(Refer Slide Time: 00:23:23)

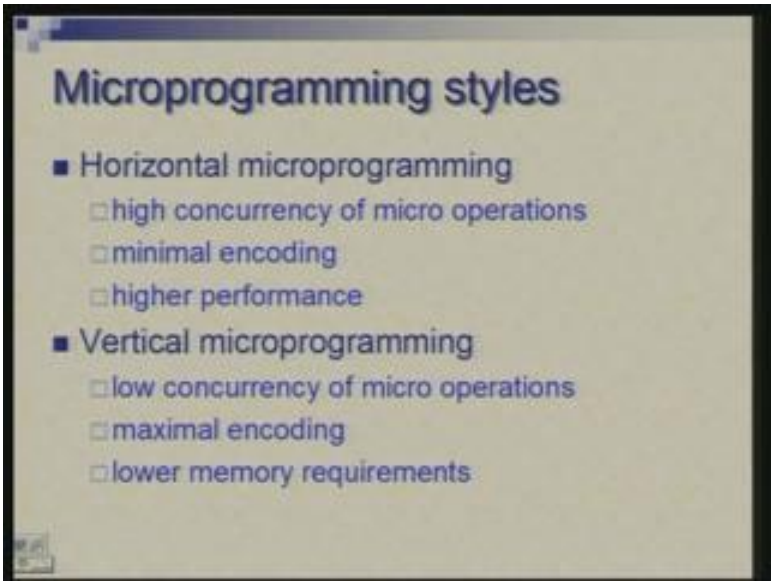


So now how does a microprogram look like?
We have a microprogram containing ten instructions. This is a very low level program lower than the assembly language program which we studied initially in this course and very simple primitive operations are being done. Here I have written so that each line represents one micro instruction or one word in the microprogram memory.

Incidentally a microprogram memory is also called control store. So the first instruction I have labeled where you fetch do PC increment and the sequence control is that you do in normal sequence.

In the second one we are doing this, this, this and doing dispatch 1. Here we do dispatch 2 and here again example of sequence, example of sequence and then rest are reset. So there are labels I have put. When you do dispatch 1 you branching into one of these labels beginning with 1; 1a 1b 1c 1d is the choice of these; in dispatch 2 there is a choice of 2a or 2b. So this is a microprogram written in a symbolic form. And once we understand what each of these symbols is, a micro assembler can translate this into the contents which will go into the control store of the microprogram memory. There are variations in the way you structure your microprogram; there are two styles.

(Refer Slide Time: 00:25:02)



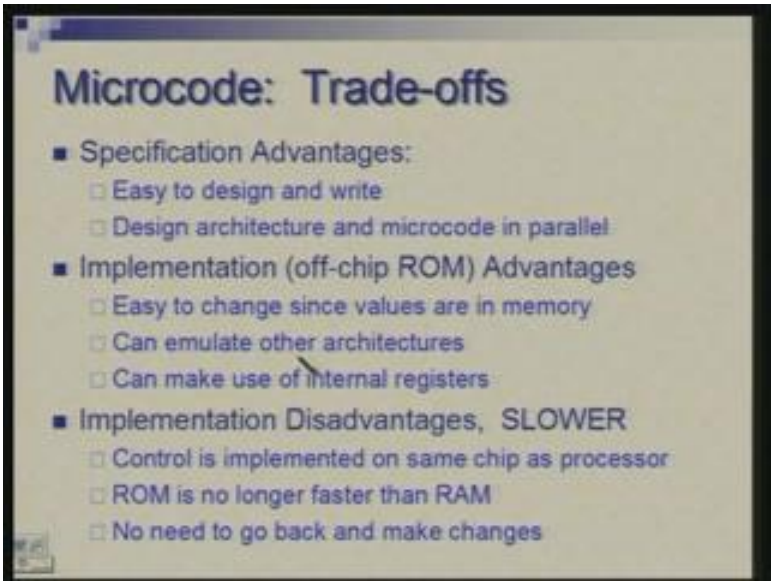
One is called horizontal microprogramming and other is vertical. The approach we have followed is what is actually horizontal microprogramming where you allow many operations to be done concurrently; whatever the data path can support you have a provision of doing many micro operations together within same instructions. There is a high degree of concurrency vertical would be low degree that means you try to do only one thing at a time. So, fetching an instruction in pc incrementing they would be done in two different instructions. The idea there is to conserve the space.

Here as you have seen, altogether the space we require is just ten words and each word having 22 bits so it is not too much of space and you will not worry. But in the past many processors have been designed which have been very complex and programs run thousands of words.

Apart from the concurrency of micro operations there is also a question of how you encode each micro operation.

The way we have encoded was that we should directly get the control signals out of the pattern of bits. But since the number of patterns which are really utilize is much less than the log of the number of bits sorry the number of patterns in general could be 2 raised to the power the number of bits but the number of useful patterns is much less and we can use a more compact encoding. So a vertical approach would try to do a more compact encoding and the idea here is to have low memory requirement whereas in horizontal microprogramming the idea is that you do not lose performance.

(Refer Slide Time: 00:26:54)



But on the whole the microprogrammed approach versus non microprogram which in contrast is called hardwired approach or finite state machine based approach.

What are the pros and cons?
The specific advantages of microprogrammed approach is that it is often easy to write it like a program. The definition of control can be written in a flowchart like form and then you can easily capture in the form of a program and one could work with design of the architecture and design of microprogram independently. In terms of performance however there are issues. In past it was thought that processor could be a one chip which will have data path but the control could be moved out into a separate memory chip which is ROM and you could change that so that suppose you want to make a change in your processor design you want to add a few instructions you could simply modify just the contents of this memory and rest could remain the same. So, in fact several times the families of processor were designed like this that you would have different microprogram implementation although the data path may be the same.

Also there were attempts like emulation of one architecture by the other. That means you have a processor which was designed may be with a fixed architecture in mind but you can have a microprogrammed hardware with which can implement one or more architecture so that process is called emulation. So you try to emulate; emulation is something like simulation in microprogram. You try to simulate the effect of another set of instructions and a microprogram would have access to all the internal registers, temporary registers.

For example, in our design we have register A and B. A person working at assembly level will not know that there are two registers A and B but one who is working at microprogrammed level could make access to those registers if it is required for implementing something. but penalty of all that is that this approach makes it slower

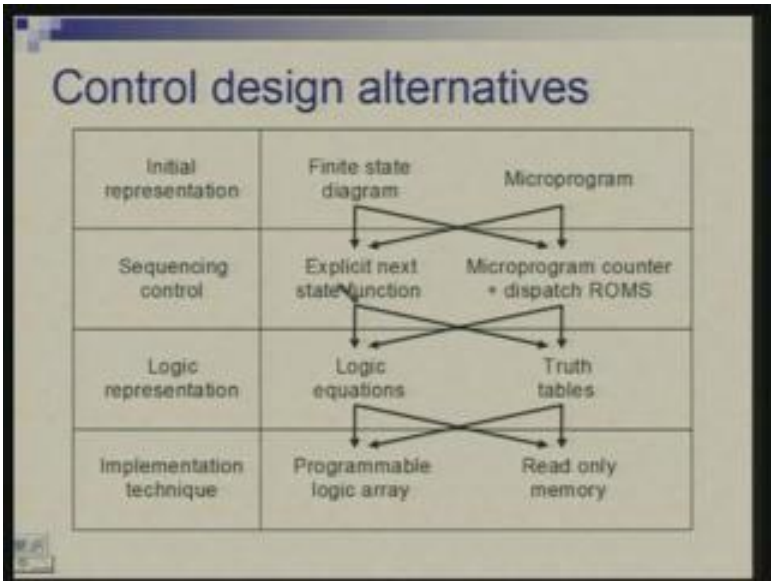
because memory tends to be slower than much much slower than the PLAs for example which is the other alternative.

So now this approach is no longer popular partly because there is a lot of performance penalty and partly because you have now tools where even those finite state machine design are not to be done by hand. So you could leave for example at a point like this and the tools will take care of the rest.

So coming up to this point (Refer Slide Time: 30:08) is not difficult and this was considered as an advantage of microprogram earlier. That is from here one could by hand or by an assembler fill up those bits in the memory. But today tools can be designed very efficient, finite state machine or hardwired controllers starting from this kind of description.

So now looking at all these possibilities we have talked of we have multiple options at various levels various stages in the process of design.

(Refer Slide Time: 30:58)

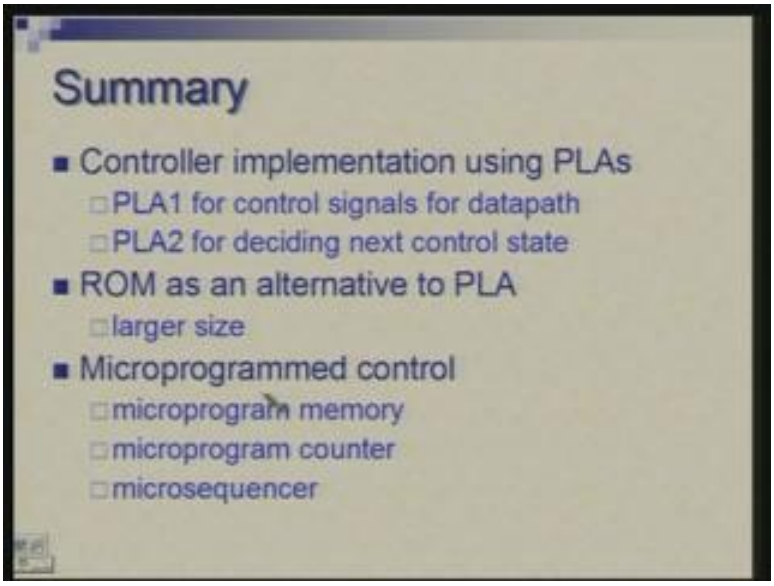


At the initial representation level you can draw a finite state machine diagram or you can write as a program microprogram. At sequencing control level you could have either explicit next state function coming out of a PLA or memory or you could have microprogram counter plus dispatch ROMs and these arrows shows that you can start with this go to this alternative or the next alternative at the next level. Similarly, you can start with this and go either way.

At the next level you have logic where you can write logic equations or you can write as truth tables and finally at implementation level you can have PLA or Read Only Memory. So you have multiple options at each level and technically all combinations…… you can start here, go to this, come to this, finally implement like that all those are possible.

To summarize; we completed the design of the controller which we started last time and we saw that we require two PLAs: one takes care of generating the control signals for the data path and the second generates the next state for the controller.

(Refer Slide Time: 32:08)



We saw ROM as the alternative to PLA but there is a problem of large size and this difference could be very very significant in many cases. And thirdly, we looked at microprogrammed approach to control where the three main things are; there is a microprogram memory which contains so called micro instructions, there is a microprogram counter which drives this memory and there is a microsequencer which determines contents of the microprogram counter.

I will stop with this, thank you.