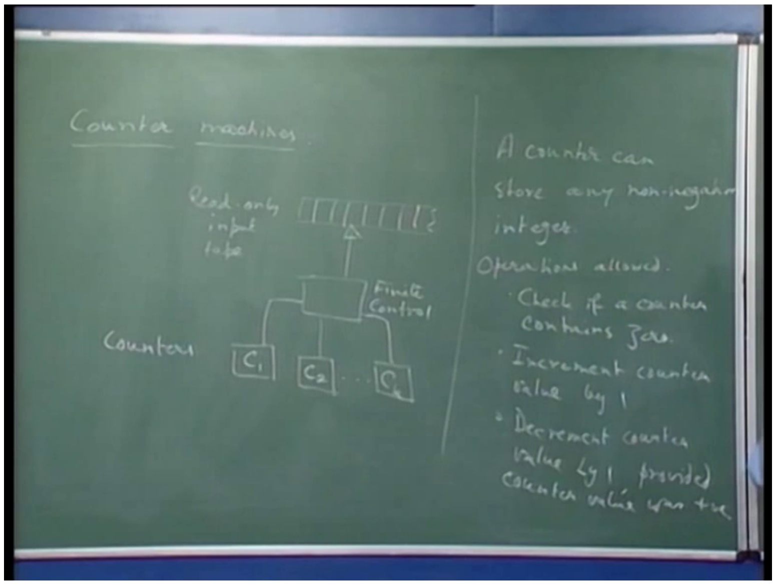
Course on Theory of Computation
By Professor Somenath Biswas
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur
Lecture 39
Module 1
Counter machines and their equivalence to basic TM model.

We will continue our discussion on restricting basic Turing machine models in some natural fashions and the in fact we will find all such restrictions do not really cut down the power of recognition.

(Refer Slide Time: 0:41)



A counter can store any non-negative integer.
Operations allowed.
· Check if a counter contains zero.
· Increment counter value by 1
· Decrement counter value by 1 provided counter value was +ve

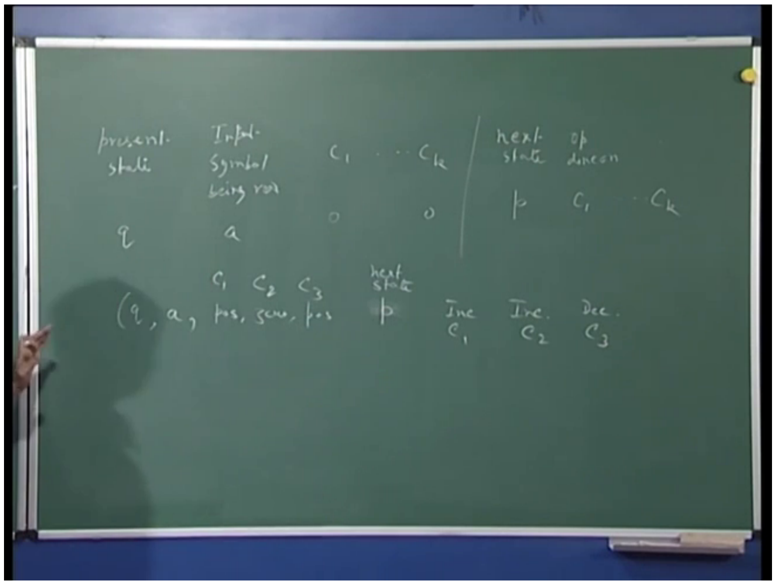
So today we will consider counter machines, counter machines we can represent pictorially in this fashion that it has a read only input tape and this is the finite control, so this finite control the machine itself can be in a number of finitely many states.

Now what it has besides of course these two a number of counters these are counters, so this machine has k counters these are counters and a counter is a counter can store any non-negative integer therefore it can store 0, it can store any other non-negative integers and the kinds of operations allowed on a counter are this check if a counter contains 0 value that is the number which is stored in the counter you can check whether it is 0 or not, if it is not 0 of course it is going to be something positive increment counter value by 1 and decrement counter value by 1 provided this decrement you can do provided counter values stored before the decrement is positive counter value was positive, okay.

So once more a counter is something which can store any non-negative integer 0, 1, 2, 3 we have no bound on the size of the integer it can store it what this machine can do with a counter is that it can check if a counter contains 0 value or not. So in other words what it cannot check or what it cannot make any decision on the particular non-zero value of a counter all it can check is the value 0 or not, if it is not it can be anything 1, 2 whatever that might be.

It can always increment a counter value by 1 it can decrement a counter value by 1 that is subtract 1 from the counter value provided the value was positive to begin with because you see if it was 0 essentially that means we cannot decrement otherwise we would have got a negative number of course negative numbers are things which counters cannot store.

(Refer Slide Time: 5:51)



Now a counter machine therefore will in a particular step what it will its next step will depend on of course it will depend on its present state input symbol being read so that is the symbol under the read only input tape and because it is read only this tape head keeps moving only to the right it can never come back in the sense of course it is not because it just read only but read only one way input tape and that is the kind of convention we had for input tapes for PDA’s even stack machines, right?

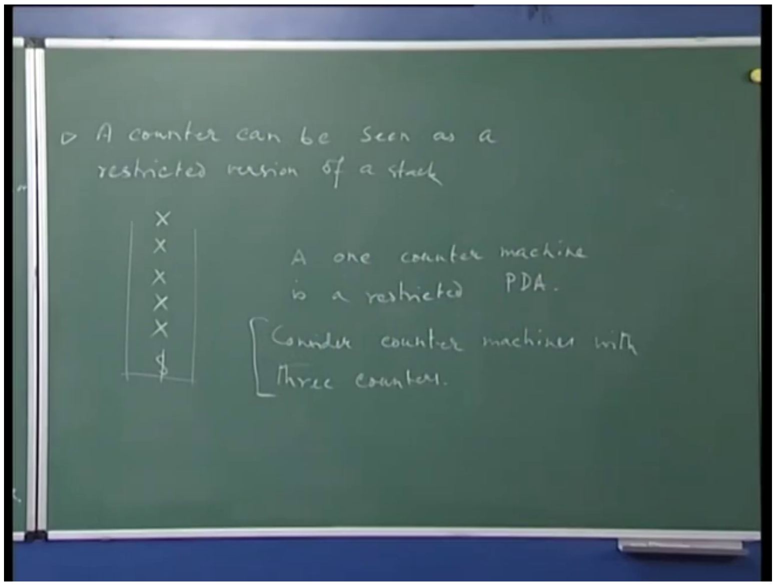
So that is a standard restriction that we have read only input tape where the head moves only from left to write and so the present symbol of course it knows at any given time it is in some particular state let us say q reading an input symbol a, alright? And now it can check whether these any one of these or all of these 0 or not. So essentially the information for C1 to Ck it can check is 0 or non-zero, for example if they are all 0 maybe the next state will be something let us say p, okay and then it should say what are the operations done on C1 to Ck the operation would be one of these basically either increment by 1 or decrement by 1.

So once more a move of counter machine we can describe by something like this so just for the sake of example let us say we have three counters so that move we can say the present state the input symbol being read and let us say we can say here positive, 0, positive what we mean is suppose we had three counters and C1 is positive that is or contains a more than 0 C2 is exactly 0 and C3 is a number which is positive then depending on if this is the situation then you might say increment first of all go to next state is going to be p and operations on C1, C2, C3 so you can say maybe increment C1, right? Increment C2 and decrement C3.

So a typical move of a counter machine in this case it is a three counter for this example can be described in this manner that depending on the present state and the input symbol that is being read and depending on the nature of values of the three counters and that nature is whether they are positive or 0 depending on this information the machine determines what it is what its next state going to be and what operations to perform on each one of the counters.

So it looks therefore is a fairly restricted model compared to our basic model in the sense in our basic Turing machine model we had a tape we could have both ways we could have written anything so on and here whatever operations we are performing they are very restricted in nature at the same time it turns out that in a way few counters will be necessary to capture the full power of our basic Turing machine, right?

(Refer Slide Time: 10:52)


▷ A counter can be seen as a restricted version of a stack

A one counter machine is a restricted PDA.

Consider counter machines with three counters.

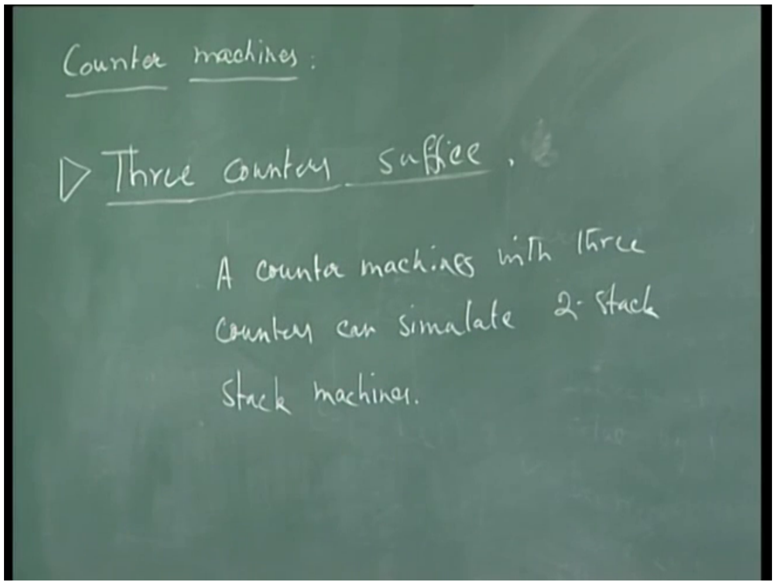
So first of all one should realize that a counter can be part of as a restricted version of a stack, why? Because imagine a stack which has other than the bottom of stack symbol can contain only one some particular symbol as its stack symbol a counter can be seen as a restricted version of a stack and this restricted stack other than the bottom symbol bottom of the stack symbol which is a special symbol which will not occur anywhere else in the stack except here the symbol that it can contain is exactly one symbol that is for example here only x can occur and x can occur any number of times.

So you see such a stack you can think of it as a counter because you can check whether the corresponding counter as 0 value or not, that will be the case when there are no access only the bottom of the stack marker is there. So if you are looking at the stack and accessing the top stack element and if the top stack element happens to be this symbol then you know the corresponding stack corresponding counter contains the value 0.

Increment counter value by 1 we it just means push a symbol x decrement counter value 1 by 1 provided counter was positive it of course means that if you are seeing an x as the top of the stack symbol you can pop it, therefore you can see that counter any counter can be modeled as a stack of a very restricted kind and once we see this we immediately know that a one counter machine is a restricted push down automata, okay by the way we will assume that our counter machines that we are considering they are deterministic and even then ultimately what we are going to show is that with just two counters we can simulate Turing machines.

Now the point is a one counter is of course very restricted especially because it also deterministic so it is a deterministic PDA and that too the stack can contain only symbols in this manner so it is a very very restricted kind of a machine. Let us now consider the case counter machines with three counters, see our strategy is first of all we show that three counter machines can simulate Turing machines and then we will show that just by using two counters you can also simulate a Turing machines and the reason for not going directly to three counter two counter machines is because the idea of the first proof that is sufficiency of three counters will be required for the proof of two counters in fact we will start with a three counter machine simulating a Turing machine and we will see how we can do with only two counter later on, alright.

(Refer Slide Time: 16:21)



Counter machines:

▷ Three counters suffice.

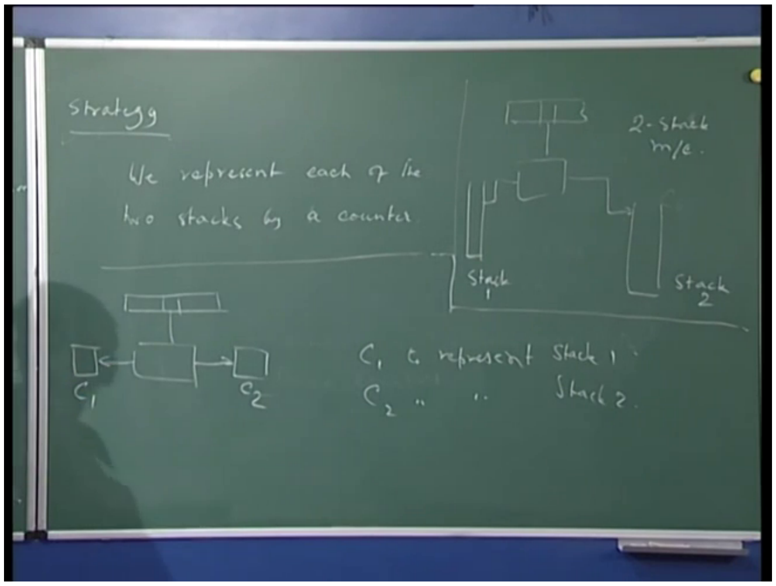
A counter machines with three counters can simulate 2-stack stack machines.

What we are trying to prove now that two counters suffice this is what we are trying to prove first of all not to the case of the two but three counters suffice, suffice in which sense? Suffice in the sense that with three counters a counter machine with three counters can simulate Turing machines, right? So our idea is going to be that you give me a Turing machine and I will give you a counter machine with three counters which will accept the same language as that of the Turing machine.

The way we will do this is through the intermediate result we proved earlier that two stack machines a stack machines with two stacks can simulate Turing machine. So what we will actually show will be that three count a counter machine with three counter can in general can simulate two stack stack machines, right.

And the strategy of this proof will be how do you show this that we will essentially what we are going to do that we will have one counter to represent one stack another counter to represent the second and the third counter we shall use to do the manipulations stack manipulations on these two stacks.

(Refer Slide Time: 18:50)



Strategy

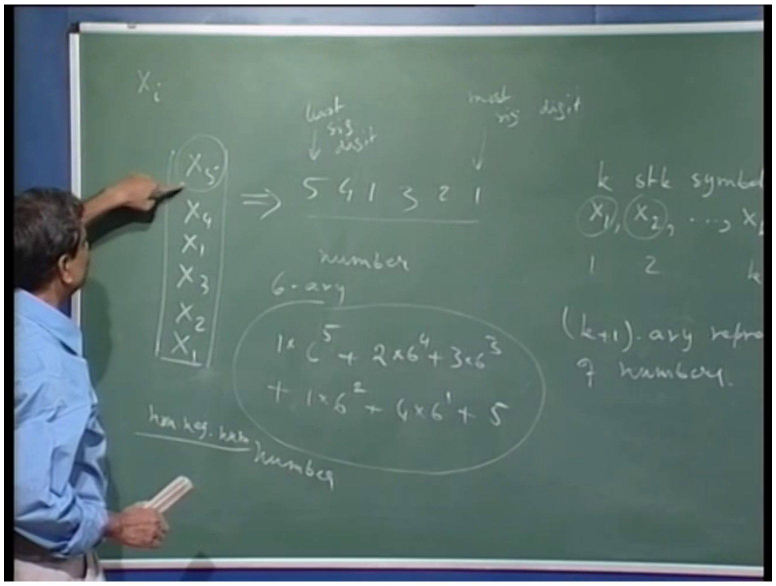
We represent each of the two stacks by a counter

C_1 to represent Stack 1.
C_2 " " Stack 2.

So strategy for demonstrating the truth of this sentence is as I said that we represent each of the two stacks by a counter, okay by one counter by basically what we mean is so here is we have a stack machine we call a stack machine looks like that has some input tape and it has two stacks, okay.

So this is stack 1 and this is stack 2 this is a two stack machine. So you give me such a machine what we are going to do is to consider a particular counter machine which has just two counters this is C1 and C2, right? And we will what we will do is C1 to represent stack 1 and C2 to represent stack 2, how do we make this representations so that you know all our operations will be conveniently done by counter machine operations the stack operations to be simulated by counter machine operation.

(Refer Slide Time: 21:27)



X_i
least sig digit
most sig digit
X_5 X_4 X_1 X_3 X_2 X_1
$\Rightarrow$ 5 4 1 3 2 1
number
6-ary
$1\times 6^5 + 2\times 6^4 + 3\times 6^3 + 1\times 6^2 + 4\times 6^1 + 5$
non neg. int. number
k stk symbols
$X_1, X_2, \ldots, X_k$
1 2 k
(k+1)-ary repr. of numbers

What you can think of suppose we have a stack and we had some symbols appearing in the stack and of course.

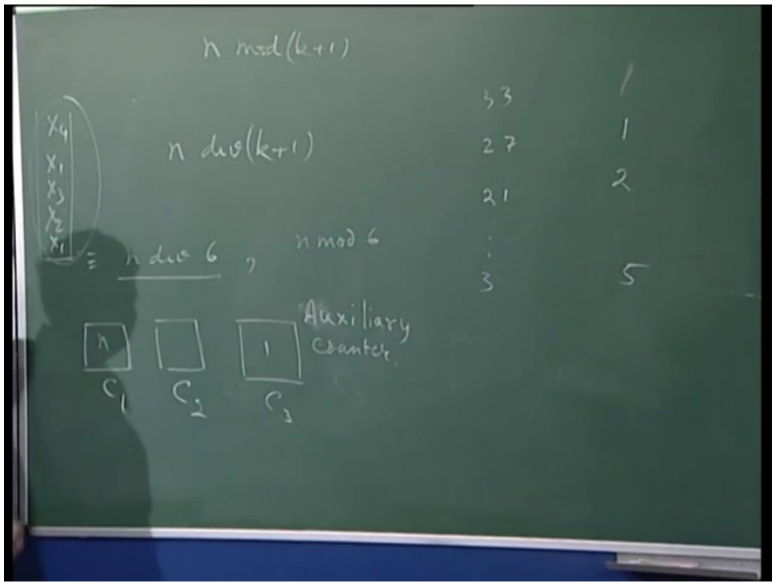
So here there are five symbols now imagine the number 5, 4, 1, 3, 2, 1 so what is what I am doing so I am each stack symbol I am thinking of the stack symbol as some Xi and so let us say there are X1 through Xk these are the stack symbols so there are k stack symbols in general 1, X1, X2 after all then the contents of a stack is a sequence finite sequence of with symbols from this set, right? And now I can think of this as 1, this as 2 and this as k in a k plus 1 a representation so these are the digits this is digit 1, digit 2, digit k of a k plus 1 array representation of numbers and then I from top to bottom we write the kre representation number in left to right fashion.

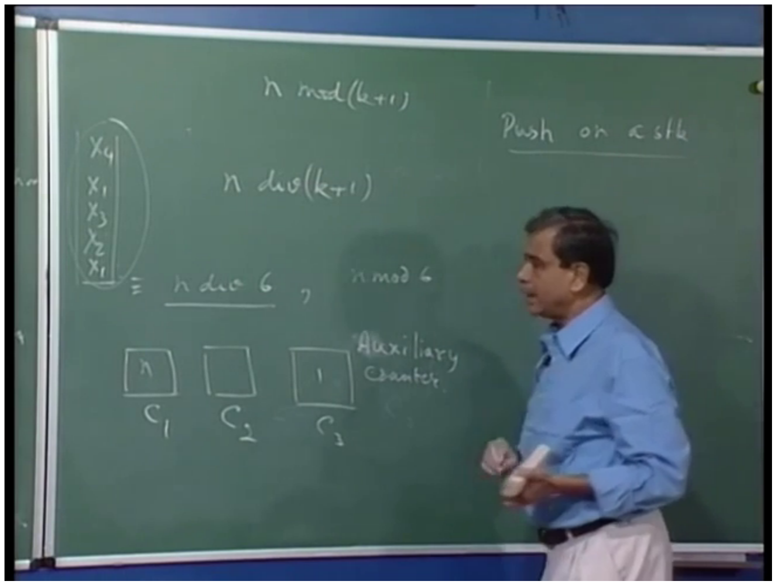
So this example is clear that suppose X5, X4, X1, X3, X2, X1 that is how the stack appeared at some point correspondent corresponding number that I am talking of in this case the base is 6 because you know let us say there are these 5 symbols can appear so we have some number and as I said that we write the number in this form, well here actually I should have said that this when I write like this I would like to see it as the least significant digit and this as the most significant digit, okay.

I am this is of course that is how we read the stack I am writing like this but imagine the most significant digit of the corresponding number which this codes is the number which is at the bottom of the stack here and so on. So now think of so therefore this is a definite number and what is that number? In this case when the representation is 6 array so we can say this is going to be this is the 0th first, second, third, fourth, fifth, right? So I can think of this as 6 to the power 5 1 into 6 to the power 5 plus 2 into 6 to the power 4 plus 3 into 6 to the power 3 plus 1 into 6 to the power 2 plus 4 into 6 to the power 1 plus 5, right?

So this is a very simple idea the idea is that think of the string which is there on the stack at any time as a representation of a certain number and this number the most significant digit is appearing at the bottom and the least significant digit is appearing at the top, alright? So now this is a particular number, right? And numbers are things which we can and this is of course as you can see this is going to be a positive number always representation of a stack contents in this manner is going to be a positive number or rather I should say non-negative number and what we can do is to say that this number is what is stored in a counter empty the stack empty means the number is 0 you can put it that way.

(Refer Slide Time: 27:55)





So now you see suppose you wanted to pop this number or rather this is a stack so you wanted to pop this symbol so then once you pop this symbol this stack would have had X1, X2, X3, X1, X4 once you have popped the top symbol the stack will look like this but you can easily see what is now the number that corresponds to this contents of the stack, right? So the top of the stack symbol was represented by the last or the least significant digit, so essentially what we are doing is if you want the number corresponding to this you are supposing this original number is n all you are doing is n div k, right? Or k plus 1 we are saying our arity is in this in general it is a k plus 1 array representation.

Now in this particular case the number that I will get here since it is a 6 array representation is n div 6, right? So this is equivalent to the number n div 6. So obtaining the resultant stack after popping the top symbol is equivalently if we think in terms of counters is obtaining the making the integer division by a certain a fixed number 6 in this case. Now how can you carry out this in terms of a counter? Remember that suppose this was your stack 1 and therefore this number n was stored in counter C1, counter C2 we would store the number corresponding to the stack contents of the second stack and recall that we are considering three counter machine so we had a pair counter.

Now what I will do so this is the spare counter or auxiliary counter I will use this to find out the result and that is fairly simple, is it? What we are doing supposing we had n here so we start decrementing n every time we decrement 6 here we add 1 to this starting of course with the value

0, is it? So what will happen so suppose the number was let us take a simplistic case 33 so first time you reduced this by 6 so how, what is that 6? Because that is a fix thing that is the arity which you are using for representing numbers which is so therefore you know and 6 is a constant so far as one particular stack machine is concerned because that the number of different symbols in stacks will be a constant and that is the constant which is going to define your arity of number representation, right?

So decrement 6 times this is possible 6 or whatever that constant thing is now that since it is a constant we can do this decrement by means of a finite number of steps that case of course you want 6 so 6 steps that will require of the counter machine 6 decrements here and once you complete all 6 without this number becoming 0 that is the numbers result you could fully take out 6 from here then you will add 1 here, right? So at that time you saw basically you subtracted 6 so what would you get? You will get 27 having decremented 6 from here.

So once this value became 27 this value became 1 then next time another round of taking out 6 from here you will get 21 here it will be incremented by 1 so this value this counter value C3 will be and so on, right? So at some point of time you will have three left and this counter value will be 5 and now when you was trying to decrement 6 from here you have done 1, you have done 2, you have done 3 and then immediately we will see this particular counter has become 0.

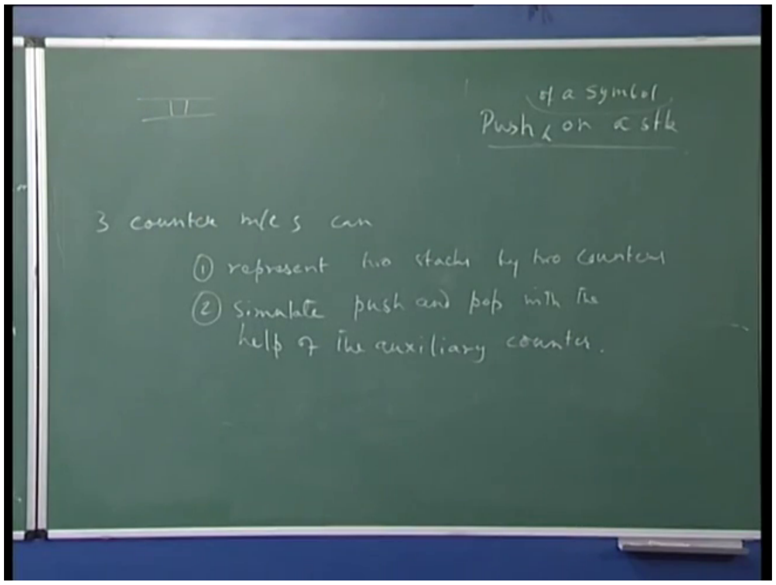
So therefore you know also in this process what is n mod a plus 1 in general or in our particular example what is n mod 6. So this would give you what is the top stack symbol which you have just popped and this will give you the value of the counter that would correspond to the stack which results after popping of the top symbol. So what I am trying to say and this is now fairly clear that popping operation in a popping operation of a stack we get something the top stack symbol to do whatever we wish to do with it and then the stack becomes less by one symbol.

So both these operations we can do by in this simple manner now of course you after getting the right value in C3 here we will copy back C3 contents to C1 because that would have been what is desired because C1 is representing stack 1, right? So any particular stack operation whether it is C1 or C2 so long it is pop operation popping of the stack we can do it by this manner it is way easy to see that pushing or adding a symbol push on a stack also can be similarly carried out if you wanted to push let us say 4 on this, right?

This was the number and you wanted to add 4 to it that means correspondingly the number the symbol x4 is pushed on top of the stack in the corresponding number would have been this now this has become the this 4 has become the less significant digit, how can we get the resultant number? The resultant number would have been simply n into 6 plus 4, right? Because remember this side is the least significant digit and this side is the most significant digit.

And can we carry out such an arithmetic operation using the counters only using the counter operations, yes that also we can, why?

(Refer Slide Time: 37:19)



Push of a symbol on a stk

3 counter m/cs can

(1) represent two stacks by two counters

(2) Simulate push and pop with the help of the auxiliary counter.

Because let us say I have a counter which is currently contains n and I would like from here finally the counter to contain let us say 6n plus 4. Now remember we will do this using the auxiliary counter and what will be the strategy here we decrement 1 from here and add 6 to this auxiliary counter every time I decrement 1 here I add 6. So if I do so when this becomes 0 this as 6n as contains auxiliary counter will have 6n. And now this is a definite constant that you are going to add this because of whatever reason because you know in some step x4 was added on top of the stack.

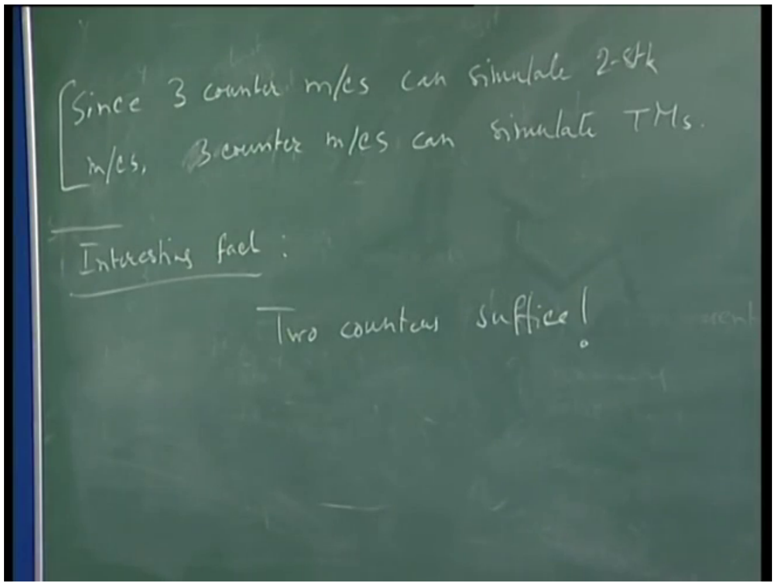
So now you add 4 to this that means increment 4 times you will get 6n plus 4 so now the auxiliary counter contains the correct value to represent the new stack here stack contents here so now you copy back the contents of here to here by decrementing 1, incrementing 1, incrementing 1, incrementing 1 remember it had already gone 0 and so on. Therefore if you just perform this operation in decrement this increment this till this becomes 0 we will transfer the contents of auxiliary counter to the counter of that correspondent to the stack whose operation push operation we were simulating.

And therefore my point is that both push and pop we just saw that push on a stack we can simulate by counter machines if I just be using 2 counters we can simulate the push of a stack what not exactly push of any string but what we had shown was push of a symbol 1 symbol but that is general enough because you can always even if your PDA wanted to push a string which is whose length is more than 1 that is it contain more than 1 symbol this you can of course think of as pushing separately all these symbols in the string.

So this is general enough, therefore what I have just told you to summarize it will mean 3 counter machines can one, represent two stacks by two counters, second what is the other thing it can do that can simulate push and pop with the help of the auxiliary counter. Now what is left for simulating a two stack machine, see a two stack machine of course read symbols from input but that is direct, is it? Because your three counter machine will also have an input tape and as it reads an input here the same input can be read the same input symbol can be read from the three counter machine corresponding input tape.

The stack machine changes states with the state of the stack machine we can keep track of in the finite control of the three counter machine. So therefore it is not difficult to conclude from the discussion that three counter machines can step by step simulate the operations of a two stack machine, right?

(Refer Slide Time: 42:52)



Since 3 counter m/cs can simulate 2-stk m/cs, 3 counter m/cs can simulate TMs.

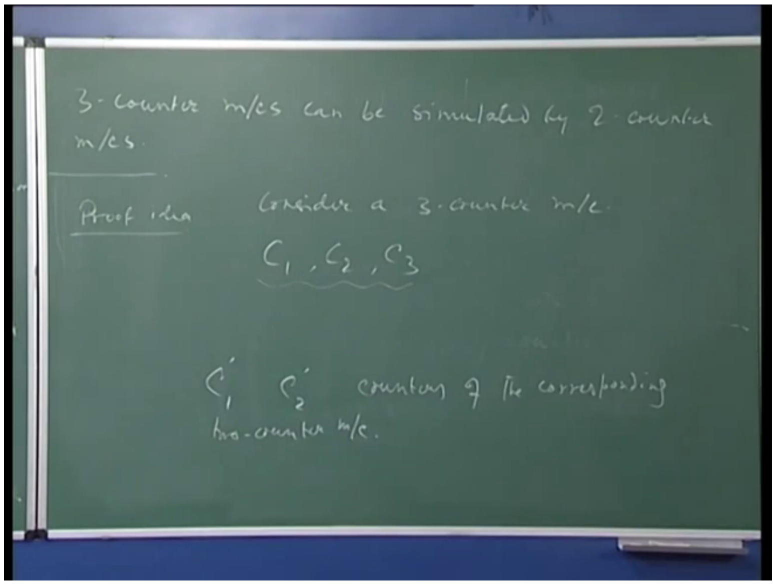
Interesting fact :

Two counters suffice!

So therefore our conclusion is since three counter machines can simulate two stack machines and since two stack machines can simulate Turing machines we conclude that such machines that is three stack machines three counter machines can simulate Turing machine, okay.

You see it is fairly strong restriction instead of a tape of a Turing machine the memory is now simulated memory component of an algorithm is being simulated by just three counters you know the interesting thing is that actually we can do only with two counters.

(Refer Slide Time: 44:35)

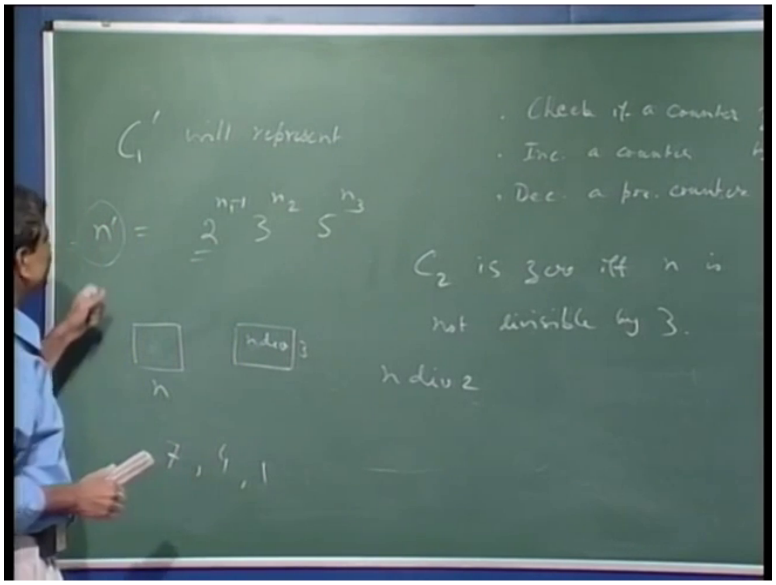


3-counter m/cs can be simulated by 2-counter m/cs.

Proof idea Consider a 3-counter m/c.

c_1, c_2, c_3

c_1' c_2' counters of the corresponding two-counter m/c.



Now let us prove this statement the way we will prove this is that we will show that three counter machines can be simulated by two counter machines, proof idea consider a specific three counter machine. So it is let us say it is three counters C1, C2 and C3 these are the three counters. Now our two counter machine what is it going to do? That it will represent all these three counters in one single counter, so let us say C1 dash and C2 dash are the counters of the corresponding two counter machine and here what we will do is first of all let us say that we will say that C1 dash will represent or will have all the information of all these three stacks.

Now recall that these three stacks at any given time it contains three numbers n1, n2 and n3, right? So essentially keeping track of three counters is keeping track of a three tuple of numbers n1, n2, n3 anyone of which could be 0 but none is negative. So that is a very simple way of there is a simple way of representing a three tuple of numbers by means of one number that is well known and this is called Gödel numbering so in consider this number 2 to the power n1 3 to the power n2 and 5 to the power n3 notice that these are 2, 3 and 5 are prime numbers and this therefore is one definite number n and I claim this n uniquely represents n1, n2, n3, right?

So the number of 2’s in this number n the number of factor basically two factors is n1 number of 3’s as factors is n2, number of 5’s as factors of n will be n3. Now counter machines will can perform certain operations and those operations are check if a counter 0, right? These are the operations increment a counter and decrement positive counter value by 1 these are all by 1, alright? Now let us say we want to check if C2 is 0 the counter value C2 is 0 that means what when C2 is 0 that means n2 is 0 the number n has no 3 as a factor in such a case, right?

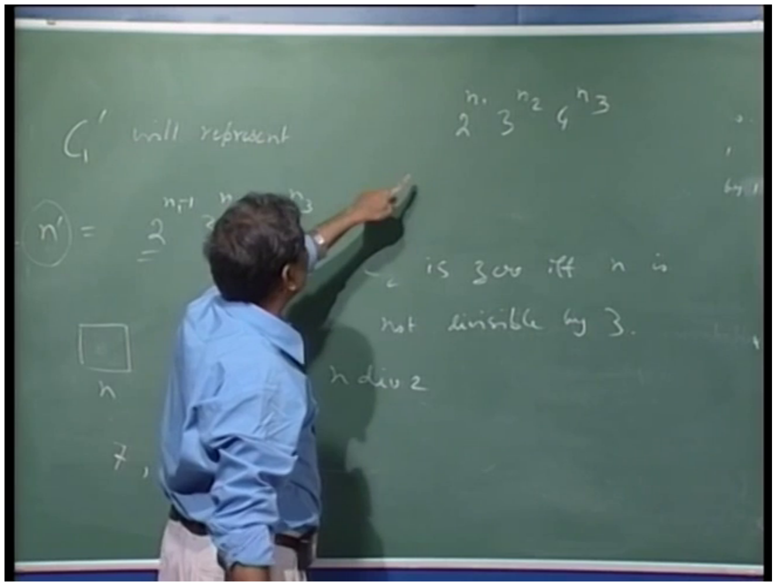
C2 is 0 if and only if n is not divisible by 3, so can I check if the number is divisible by 3 or not? Yes, by the simple use of the auxiliary counter remember I have now two counters in this two counter machines one representing the values n1, n2, n3 in one go with such a number so I have this number n and I want to check if this number n is divisible by 3, what do I do? I can keep taking out keep decrementing in this counter in groups of three if I am when I reach 0 by then if I have completely taken out another bunch of three then I know this number is divisible by 3, right?

So for example 7 if I first decrement it by make three decrements I will get 4 then I make three decrements I will get 1 and now this 1 I cannot as I try to decrement 3 times I will find that I will not be successful and the result will be the counter value will become 0 before the you know we have managed to decrement 3 times. So I know the number is not divisible by 3, right? And since it is not divisible by 3 therefore C2 is definitely 0 actually we were not losing out anything because as we were decrementing 3 at a time you can keep putting the number of times you have been it has been possible for you to decrement three times I mean 1, 1, 1 and you know basically the result of n div 3 can be here and on hand you will know how many the remainder will be there we can actually restore back this number so that is also nothing is getting lost in doing this checking.

Increment a counter by 1, so let us see we wanted to increment this counter C3 that means just multiplying the number by 5, right? So increment I can do, decrement a positive counter by 1 is dividing the same thing essentially if I supposing I wanted to decrement this counter provided it is n1 is non-zero that is possible all I need to do is to divide the number get the number n div 2 because when you decrement n1 you will have 2 your n1 minus 1 and the old value old this new n prime is just old value of n divided by 2 and we have already seen how we can divide by a fixed constant remember these are fixed constants 2, 3 and 5, right?

You should note that this you know situation will not be possible if instead of prime numbers we had put something else, for example instead of 5 had you put 4 and for the representation, right?

(Refer Slide Time: 53:18)


C_1' will represent
$2^{n_1} 3^{n_2} 5^{n_3}$
$n' = 2^{n_1+1} \ldots$
is zero iff n is not divisible by 3.
n div 2

So your representation would have been 2n 1, 3n 2, 4n 3, right? And now we said that divisibility by 2 will tell me whether n1 is 0 or not, so suppose n1 is 0 and yet the number will be divisible by 2. So that is why you need prime numbers whose you know powers are used to keep track of the contents.

Now to summarize what we have said is that the operation of three counters can be simulated by just two counters all the three counters of the original three counter machine their contents are represented in this manner in a composite manner in 1 counter and the auxiliary counter is used to do operations on the three counters to simulate the operations of the three counters we can do checking by 0 increment decrement.

So therefore the just with two counters we can simulate all the three counters therefore the a two counter machine we can design which will be doing same thing as any given three counter machine and therefore we say two counters will suffice in conclusion Turing machines can be simulated by counter machine with only two counter please remember that if you cut down the number of counters by 1 if you had 1 counter machine then we will have a very restricted form of a DPDA which is of course cannot in general simulate Turing machines.