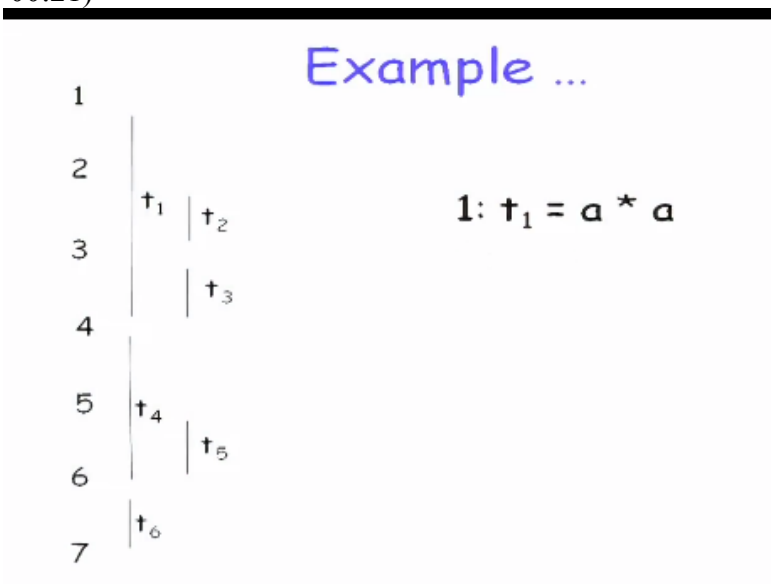
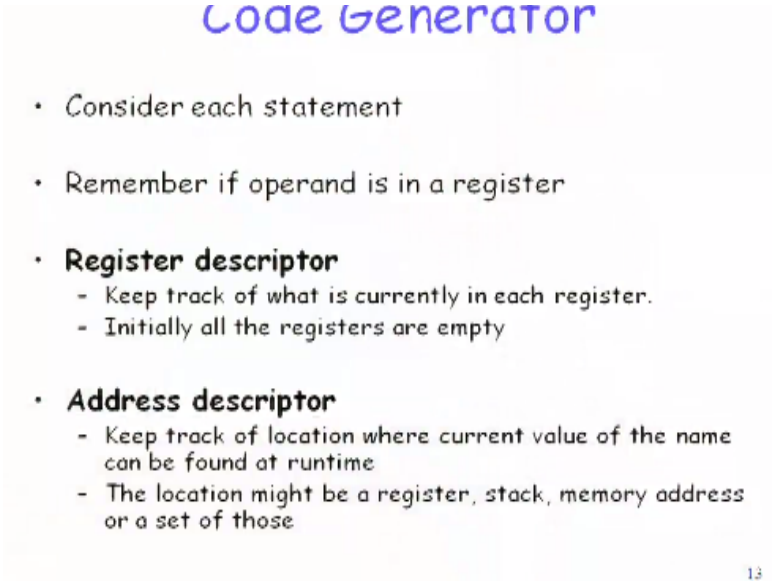
Indian Institute of Technology
Kanpur
NP – TEL
National Programme
On
Technology Enhanced Learning
Course Title
Compiler Design
Lecture – 29

by…
Prof. S. K. Aggarwal.
Dept. of Computer Science and Engineering.

So if you recall in the previous class.
(Refer Slide Time: 00:21)



Towards the end started.
(Refer Slide Time: 00:22)

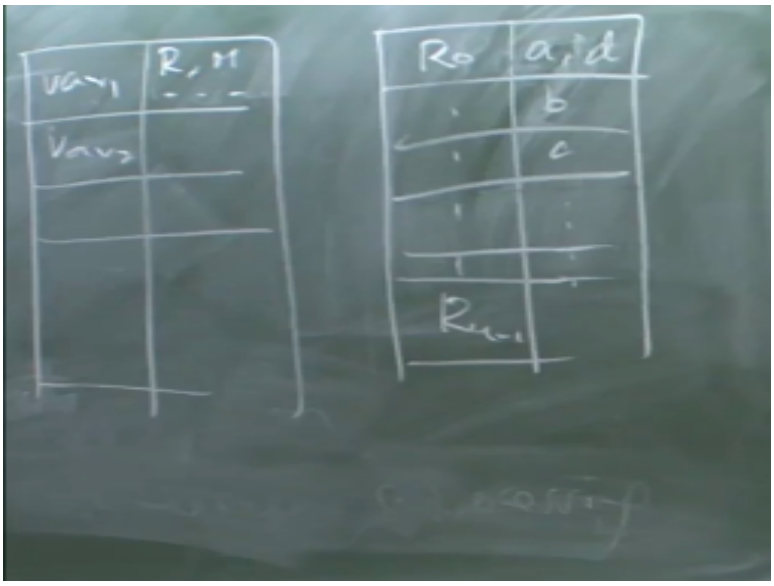


Looking at information about next level information and we are trying give the number of now actually look at this part this was just to bring this example of but we do not minimize number of country what we do is next to information at the time code generation group make sure when that I am using the liters when I am using tempers I use few so when I talked about port generation that exclusive information.

So let us look at what code generator does and what I am assuming that we have already have a models of the machine and free address code is not be converted we look at both kind Boolean expression so kind of machine that is invention okay, so what you want to do now is so what we want to do now is that for each statement in each statement is in terms of the form either text in designed by offset or is if X well of Y then for what to do is we want to remember if augmentation while register resources but will also remember more things so what we do is we create two kind of this little more one is what we called register descriptor.

So basically what I do is I engage tables and in this table.

(Refer Slide Time: 01:49)



I say that if I have registers R 0 Rn - 1 I want to remember what each of these resistors contains okay, and so I may have information by saying that this register contains but you are fake this register contains value of V difference and it contains C and so I also have a situation where I may say that a register may contain two variables so register may contain let us say A and B how that happen it is possible simultaneously contain value of two variables possible so I so not look at the value ,variables may have the same value.

So the reason this may happen suppose I say A is sign D is a signed A so what they do I will do is that I have a country which has you know that you know register the I just do a assignment ,assignment means this value is already in the register I do not want to change anything so register descriptor basically says keep track of what is currently in each of the registers and initially before the basic talk.

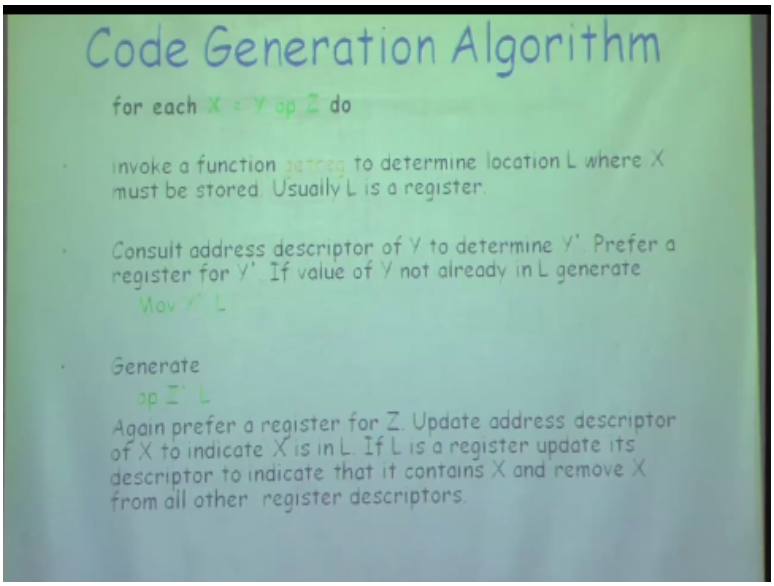
I assume that every distance registers okay and I am start using register entered in the basic law now I have another descriptor and this is called and this descriptor and what I do here is that for each of the variables. I want to now keep information and these variables are nothing but either they could be user variables or they are temporary variables and I want to keep information like that for each variable so if these are my variables.

I want to know places where they are available so this may be available for example in both inside memory are variable may also more than one place so example if I say I have a variable in when say register that it comes a distance variable so this is what my address descriptor is that I want to keep track of locations we have current value of and location might be registers that memory address or a set of intubation our variable may be available contently more than one

location and are register may keep value of many variables including so these are the two description.

I start input generation I say that initially where will variable be this register contain and start the port generation of a basic law all of them will be not going to be end of the register I will start loading them and all through the register and each register to begin with this going to have so these are the two description as I come to Boolean's island more information okay , but right now for arithmetical logical expressions sufficient information good okay .So what we do now is let us look at how we do code generation.

(Refer Slide Time: 05:11)



And what I am looking at is the simplest possible output in fact I mean this is most general output say X L designed of Y or I can have unitary operator I can say that Y I can have just assignment where I say X is actually sine Y this is the most complex decide or first concept we understand what I am trying to do be we want generate fast efficient code therefore as far as possible to you try to use registers now suppose. I say that I want to compute these values now to begin with everything is in memory.

And Y and Z are in EE and I want to apply this operation on that and store value it that is now as far as possible I would like to do the register to slot value of x that is the first thing second thing I will say is that suppose Y is or Z is already in one of the registers and that resistor has or that particular then I can use the same register and this information comes from the Mexicans information okay so what I do is that first I said let me get a register for doing this operation that is the first function.

I will go as far as possible you will try to use register each register is not available or register is not requirement a nice moves in memory location so first I do is I make a call to a function which I called every step can we look at this function in the next coil this function does it actually returns a location which says this is where you can compute and slot value of X okay, so what is get registered does we invoke a function called get register which determines some location.

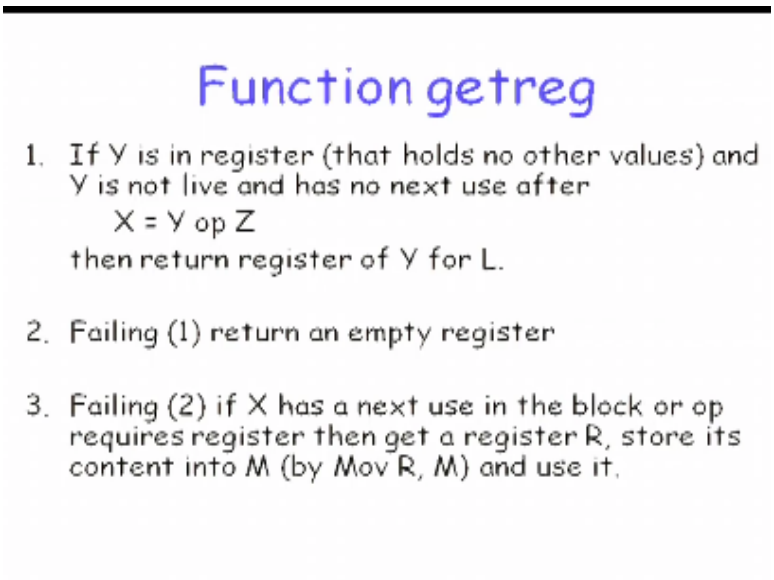
So this location need not be a register it could be anything but so do not confuse between this name get register and location. I will get ready step where X is going to be stored and usually L is going to be register now if this is in a register what do I do I will say that I can load now suppose this register did not contain anything what I can do is I can load suppose the instead did not contain anything I can load Yx the register and then I can apply the calculation on the register.

which said ʌ and then I just have to move it but suppose now if I checked that Y is already in that register I can do that checks and how about that so it is possible that get registers found that since Y does not have a future use I can return whatever register is useful by for keeping the value of X so remember this p1 is being assigned even + P okay, so I know that beyond this point p1 has no use so I can use the same register, so I need to check whether this register contains already Y or not in so now we look at and this descriptive of Y and we want to find out y prime now remember what is Y prime.

Y maybe simultaneously available in more than one location I want one of those locations which is fastest to aces and fastest this normally going to be the register so we determined Y prime and we prefer a resistor for Y prime d and if value of Y is already not in L then what do I do I generate one instruction which says move Y prime 2 L okay, it now remember what can happen here is that I am saying that look at why I look at Miss descriptor which says that Y is in the gist of zero but R zero can have a future use so get register with no effect on that the term register r2.

For example I can overwrite R 0 I need to protect it so what do I do I say that now you copy value from, R0 okay and then once I generate once I have give it to this instruction so move instruction is basically copy would say that from the location where Y stored and that when location is Y prime note this fashion will do it that is the location which has been returned to me and after that I generate one more operation which says now off and off is this off which is this operation of Z Prime and now what is this what is that prime Z prime again is the location from

there I can pick up value of Z at runtime and this again is expected to be the fastest location with values so it is possible the exact also from the address descriptor is available.
In multiple locations and already the register and what do I do I just apply this operation so suppose to begin with I say Y is it R 0 that is in R 1 and this returns a value saying that both are 0 and R 1 have a future I cannot feel it becomes R 3 for this so what do I do immediately I say first move Y to 2 R 3 and then I say and R 2 so these are the two instructions it is will get it and sometimes first instruction may not be generated if Y is already in a register which has then I generally the second instruction.
Okay now what do I do after this so I started doing code generation for this three in this form which says X is Y (x) object and I have these new instructions am I done do I need to do anything more code generation or any more bookkeeping I need to do it enough because now I have to change all my descriptors so now you say that I am using now L which means a register so now this contains value of X and then I have to change at this descriptors of X to say that X is now available.
This particular location so now again I am going to prefer a register for Z which was that prime and after this I update all my descriptors of X to indicate that X is in L and if only the register updated descriptor to say that now it contains accent remove X from all other locations okay I do bookkeeping after doing this for generation and now we say that if Y & Z they have no excuse and this is where this next use information is coming in and we say that there that going to be there will to be dead on exit from this block if we change the descriptor to indicate.
That they are no longer containing these register so that good register or can take advantage of this fact that this is the register and therefore this indicate become free okay , so I need to generate two instructions a then I need to do some keep this is where I use excuse it is one of the places very simple straightforward as far as simple arithmetic restrictions are concerned I can easily see now you can see that I assume the machine which has move and so on now you can see that if I am using certain addressing modes then I say that whatever that addressing mode then I will have to use addressing code for Y prime and Z prime.
Okay I will just using a register so that so you have complex addressing codes then I will say that whatever mode is which will help me fetching that U of π Prime and value of Z prime now let us move on to this function get read which I used here now how do you get rid of forget that is saying that is being invoked to say that give me a location where is this computation so what do I do now we say that if Y is it some register.
(Refer Slide Time: 12:53)



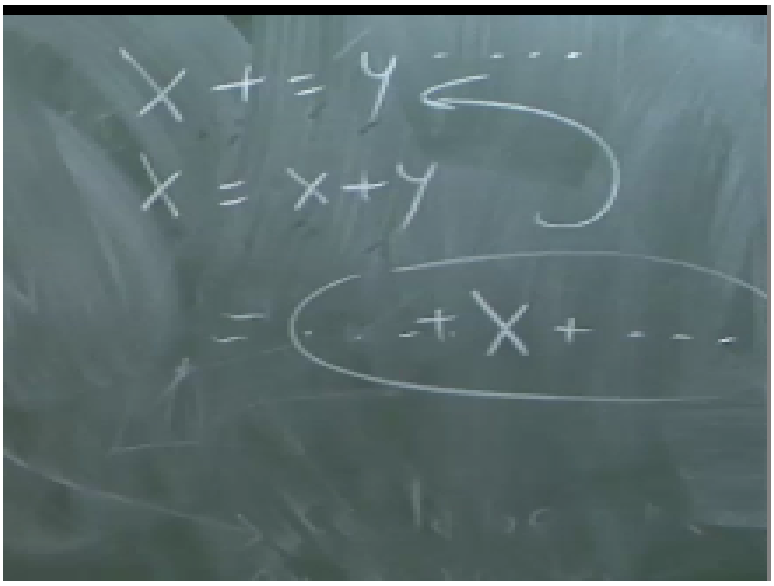
So what is Y now Y is the first operand here it is an obvious that is why is in some register. That does not hold any other value you can see that Y you may have some other value there this variable may not have a future use but this may have so I need to remember that so if Y is in some register and has no the value and Y is not like and has no next use after this particular location then what can I do I can say use the simplest if 0 and the oldest are 0 does not put any other variable and Y is known excuse that means I can free R 0 immediately.

After this particular instruction is the same that is therefore who in this computation and get registers going to return already support now it may so happen that this register either holds additional value or π may not be a register in that case this condition will so what do I do that is I said get a free register okay, and use that I was giving you the previous example I just saw a fresh register so to begin with no variable will be register so first time. I am going to use are you busy now suppose there is no activity step this can still feel right that I find all the registers are full.

I can find a new location I preferred a occupied so can actually register and ready so this is what is known as register filling I say this register does not have an immediate use what I can do is I can take value of this register loaded some country location start using this register and when that particular variable is required then I load it back into the register and continue from that point onwards and this is a technique which is known as register spilling.

So what we do is if X has an excuse in the law or operation is such that it requires a register then get a register and how do I get a resistance corresponding people memory location by this and use it okay, and at some point of time we have to well advantage of this is going to be that suppose this X has lot of users beyond this one and this particular variable which is containing R

is not going to use in the next two instructions then I may do one extra scope and expert good but then all the excess of variable X are going to be cost.
That is a advantage so now you can see that if X as an excuse in block or this operation sometimes we say that I can do this operation only on other is operational imagination needle is so register that is the filling is a standard technique which is used and as fast and you can see that the register filling is always going to have over it so you would like to minimize the register filling as far as possible you will see examples how to reduce the register filling but in this case.
We have to do already register now suppose this is not the part that either we say that this is the only time I am computing X beyond this point X is not going to have a use and therefore there is no point in their filling that means doing something into a memory location and holding it back and also the operation is such that it does not require a register okay then I will use directly memory location for X n so value okay, so only this memory location we use if all these conditions fit.
So as far as possible loser register but if there is no free in register and I use a register by being a freeing it and using it but if it is not required that there is no need to free register because the register to operations one for moving this content a that to memory and second time moving back into the register.
]So then you will say these two instructions optimizations you can see immediately possibly here because I say here if X is in a register that holds no the value. I can do the same thing for set and also you will see that I mean and this is something you can try by compiling you program for C and you will find that lot of actually optimization which goes on here and it is not possible that if I look at this non-trivial to find out.
(Refer Slide Time: 17:20)

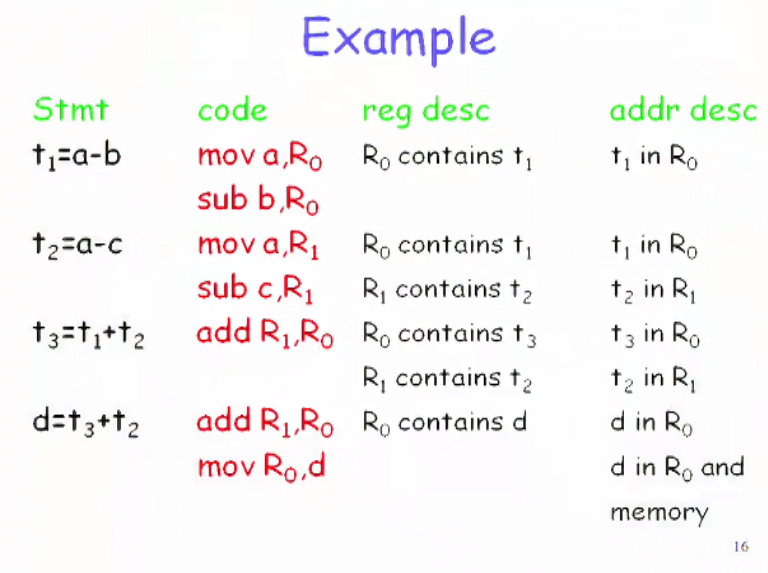


Whether X += Y and X is being assigned X +y they are = so many times you will find that when it comes to code generation compilers generate better code for this as compared to this because then they assume that the first offering from the right-hand side is same as the left-hand side and therefore you are saying that whatever is the variable here that is being used to store the value of the left-hand side okay, so normally I mean Y I mean she introduced these kind of operators.

I do not have too much computation at that level if operator if any user can help me in using these kind of operators than all this see in fact it becomes sometimes worse than this that suppose I have a long right-hand- inside where X occurs somewhere okay then whether in general this can be transformed into something like this do commutation become very possible and do a lot of analysis so all this reorganization unnecessary takes time fine and therefore if user knows that this is much faster.

We will find that so you will find that lot of when we get written up for arithmetic operators so this can take care of all possible effectively it can even take care of Boolean operators if I am just moving assignment so we will see or the conditionals are handled but at least as far as assignment called Boolean operations concern you can see ,I am just using off here I was specified what kind of so any binary operators unary operator any single assignment so how do we handle okay. So I have an example so let us go through this example so what we have is suppose.

(Refer Slide Time: 19:23)



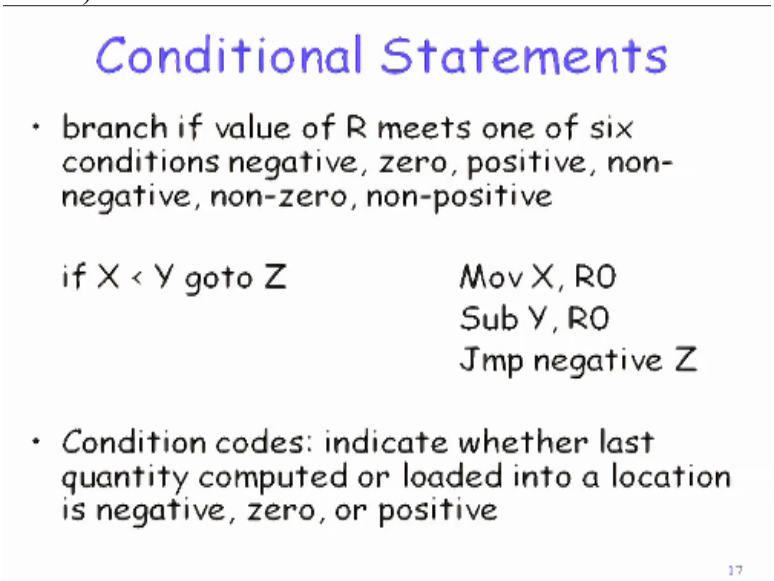
I am trying to , so I am looking at the block in which the first is instruction says T1 sin okay notation is same that whenever I used T1 so on temporary valuable required for paler whenever use other letters are user variables okay and we will assume that all user variables are going to and so what we do here is B for this we say that my get register routine is same that whenever I used this because nothing is it registered returns the register it says it can be used for computation which is R 0 so I now first say move into R 0 and then I subtract B from R0 and how you might descriptors change at end of this these two instructions.

This is says R0 contains T1 and this description okay then I have this instruction which says T 2 is assigned a - c okay, now what do I do now I can see that A is now what happened here was that I decided that whatever is this value .I am going to another again so now since this value is already in p1 we now say that reason we look at this port if I am doing this extra well and then I say subtract C from r1 so now at end of this we say that R0 contains t1 and r1 contains D 2 and D 1 is now 0 and T 2 is now normal.

So let me actually give you the full code then okay, so now it says P 3 is fine even t1+t2 now I find that both even and T 2 are in registers so and beyond this point D 1 has no use so this resistor that we use and therefore what we do is we straight away instead of doing a Boolean operation straight away to an addition and my descriptor stage like this and then I say at e3 and T 2 and T 2 is in r1 so now I can use because beyond this point you will find that neither of our 0 or R1 is going to have a use for what I do is I just move this addition and then store it into memory location D and my descriptors out say that r0 contains B and E's in r0.

And these holes in R0 and memory location to follow so this way division can retire but when it comes to conditional.

(Refer Slide Time: 21:53)



I have to be slightly differently and machines provide special hardware so what happens is that normally if you recall assembly language programming here have done will find that there is something for the conditional for descriptor or position for quits on the machine so whenever you execute an instruction on a machine simultaneously depending upon the value of the result some pics are going to be set so suppose the value is very high some overflow kit will be set or the value is zero some positive some negative between he said if value is positive.

Then some quality between seconds over and that happens automatically and advantage of doing that is that when it comes to conditionals. I can take care of many of the jumps I am looking at these condition coordinates I do not have to explicitly do the computations so what may happen is something like this that when I said I had branches so what will happen is something like this I say that if X is less than Y then go to Z now I for example if I look at first two instructions here this is mu X x r0 they subtract 20 from our zero.

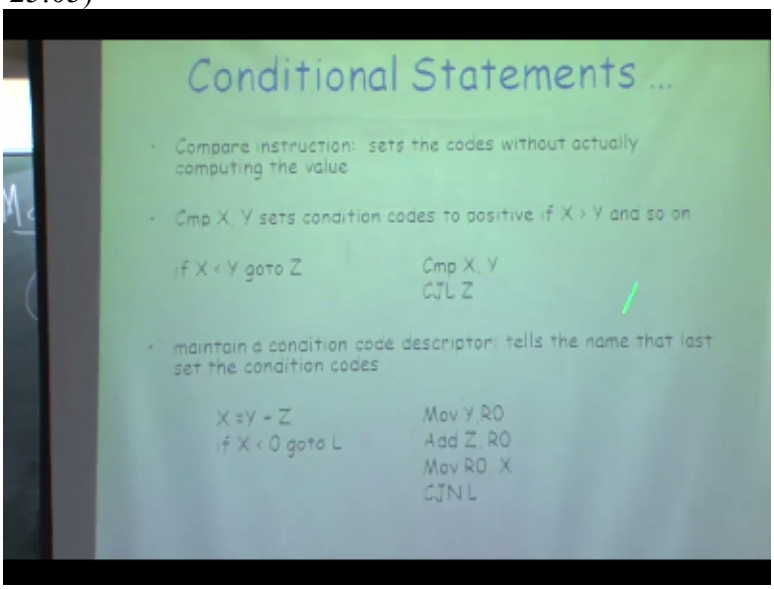
Now I do not have to know the value of R0 know I will compare when you are little bit better it is less than view all wrong simultaneously what will happen is that some condition what bit will be set which will say that if R0 is negative which is self and what do I do here then I just say that jump if that bit is negative on the location so basically what you do is that if one of these six conditions so it could be either negative zero or positive or non-negative on zero a non positive.

And many machines will have a condition both ways and simultaneously on each operations more than one condition for which they get effect we are going to set that bit and then we use this information here in the instruction and who this so I am you can see that I am not comparing are do with anything but the solution for information is being used because condition code has

been said because of this operation so what we are doing now is that we use condition codes and we indicate what was the last quantity which is computed and good even when I do a move for example so for example when I say move X x R 0 some conditions would be easy depending upon what is the value 1 what is the value of x value of x is positive then automatically move XR 0 will set the condition.

So this is loaded into location in and depending upon what is the computation or holding which happened but on these bits is going to say and I am just moving this but machines also provide a couple sessions where I do not take even how to do a movement for logical operation I can see a comparison operation and comparison operation also is going to set one of the condition codes so what can happen is something like this when I say compare x and y this sets some condition for quit to positive if X is greater than Y.

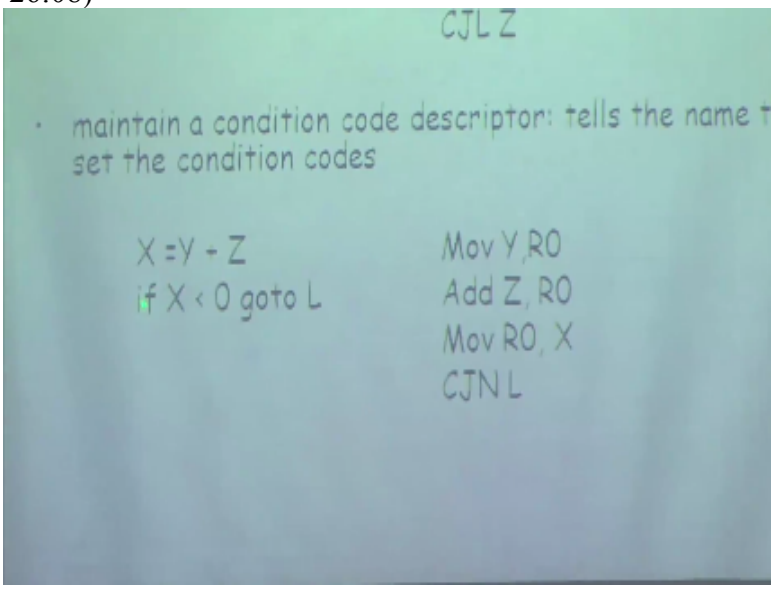
(Refer Slide Time: 25:05)



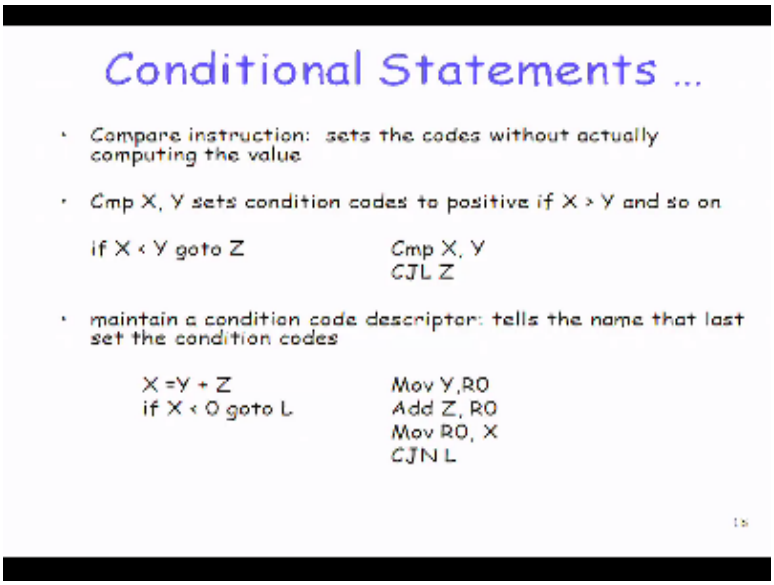
And similarly for other than way so what I may do is if I say if I want to do for generation for this I mean just say come here white and conditioned on less to Z okay. So this becomes now a conditional jump okay so this is so there are two kind of jumps one is unconditional jump where I just notice anything I say straight away change the program counter to this new way is nothing but changing the value of peace and condition. And the conditional jump is you say that change the program counter to Z only if this condition is met which says that it is less.

So I can use both these in but then I can also use to remember something known as conditional descriptor so I have these two descriptors now I can also keep one more descriptor and say that one of the last operation which still a bit or between that little bit so what's my happen here is something very interesting so look at this if I can remember suppose I say X design y plus net

and then I say if X is less than zero then jump it now because of this computation I do not even have to have a comparison efficient. What I can do is if I can remember.
(Refer Slide Time: 26:08)



What I can do is I generate three instructions for X being assigned by objects is the move by into R 0 at Z and then I say who are the requests now when this happens then I know that one of the condition code bits is going to be set because of X now it is possible that these instructions are not registered there is maybe may be there are few other instructions which are in between but they are not impacting the value of x I can still use if I can remember that the last time I loaded X this bit since that has not changed it was set 1 X.
So what can happen in that cases I can now just say that jump on negative and some little head so this operation research you can see that I can even say one comparison instruction I can remember this particular thing that the last operation was the one which really changed this particular but for that I need to do additional book keeping and that additional bookkeeping is in terms of a condition for district so for each of these condition for bits I can say that which bit was that because of what operation if I can remember that then I can further say certain instructions.
I mean again you have to see that whether machine provides this information and machine provides all these facilities are some of the earlier machines for example did not have enough condition for which they did I think a guest I remember is the one white condition code the distal of condition both bits that means I could set on create conditions and some of the recent machines had 16 conditions and 3230 this one so all kind of funny things will happen but if you can remember those things. Then you can here okay.
(Refer Slide Time: 28:11)



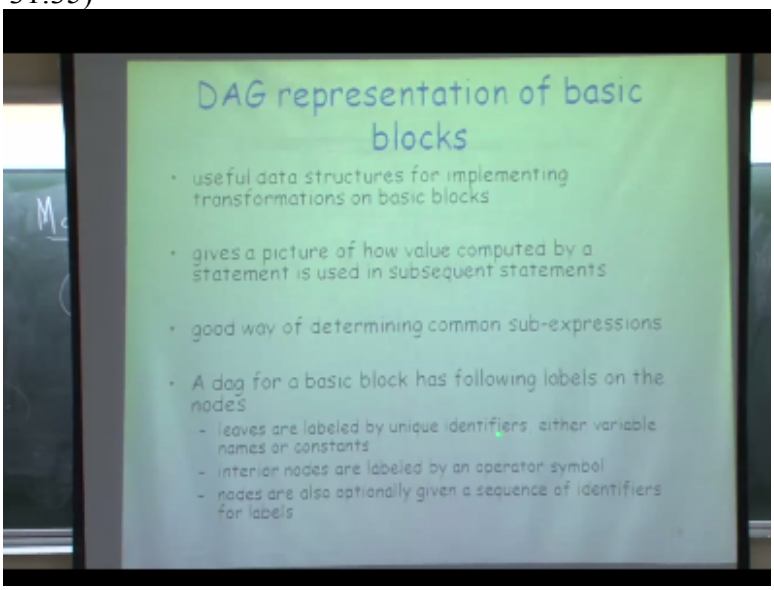
So you can see that how do I handle conditionals and how do I handle all that we have the sports and this as far as machine is concerned it is really nothing but correct code which is coming you are question in the last place if you have this adhesive the last variable so my conditional description will say that but that see this descriptor hair condition for description so my condition for descriptor suppose I say that X is signed by plus net and then.

I said compare say any suppose CMP suppose compare any is going to attack with it will activate my conditional is condition for descriptor is going to say which we was affected and if it was the same build which was impacted by X then my descriptor will change and I might be able to generate this instruction but suppose that which was not impacted which was set by X then I can use so that is why I am saying smart signified the profit that is the hardware see this descriptor condition.

For dispute say I have 16 bits I say that I keep this condition for description and say that for each build ends so this is big one with two and so on remember this is going to be here so the descriptive I can keep is only for all the bits and so let us move on it and let us also see that part where engine for products now so far I have been looking at here this port but we also remember that at one point of time we started looking at decks and how is that different from places for how is a be different from procedure code only the most employees on the same time so it a three or that only the name tempers.

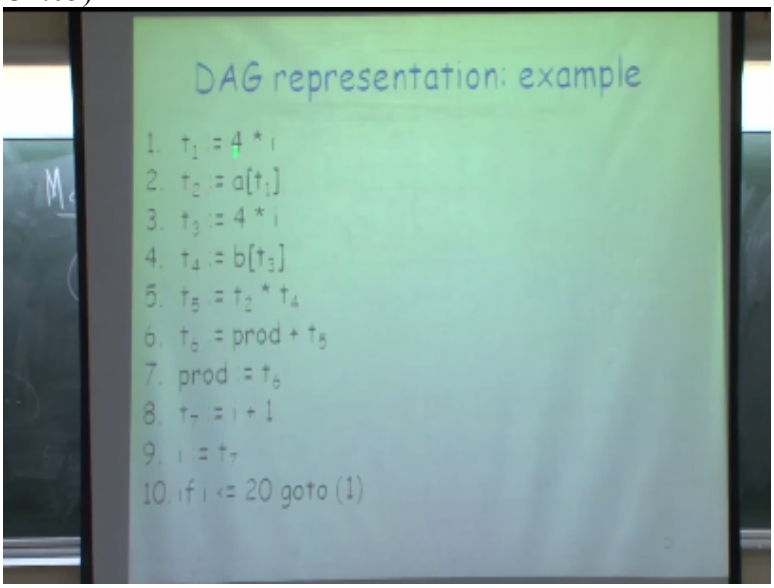
I have internal nodes and when I explicitly like to respond I'm going to so basically what will happen is that when I start moving port in addition for a three or that I have to remember now that what is the name I am using to be for each of those so what will happen is that this is useful data structure.

(Refer Slide Time: 31:35)



So we have already talked about properties of these decks but basically leaves are going to be labeled by the identifiers which are either variable or the constants interior nodes are going to be labeled by and operate symbol and nodes are also optionally given about a sequence of identifiers for lives because I have to jump somewhere so I need to have labels and this is the only information I have as far to your day is nothing other side again okay so let us look at this particular.

(Refer Slide Time: 32:05)



Piece of code okay so this is some time some representation of a tag where I am saying that I want to compute D 1 which is 4 star I D 2 which is area axis with this particular axis of phase of this basically same is AI then I recomputed is 4 star I and then I say P 4 is so this is AI then I'm

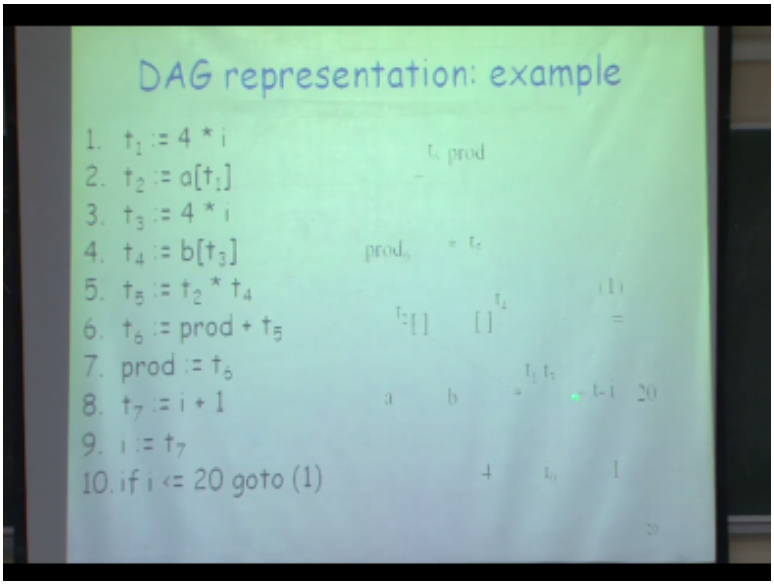
saying after doing this and then I and if it is now you can see that here for star I is being computed now since I am talking about that you can see that I am not recomputed but I use the same forms for this application and similarly.

When I say P 1 P 1 is 4 star I and therefore t3 also is false right so if I just compute this P 1 I can access a with t1 by the Nexus be with t1 and as far as these two instructions are concerned which says t6 assign prod + T 5 and then prod is sine T 6 since this is a single assignment I could have said right away t6 is being assigned t6 or prod is being assigned Plus D 5 and here I can straight away say I is in the sine Π plus 1 this is how it so happens that this is how I did it so we look at I see look at this basically and this is for them to un multiplied that you stored in T 1.

So this even actually now contains start for unevenness and then I have a and E is basically saying that I am going to get this address of it and I'm looking at index p1 and p2 is now an array operation which is saying a t1 so this is corresponding to the second instruction and then I say since t3 is a B = T1 therefore instead of deleting a I just know capturing the complication which is below this node and then I have equal the sine V T 3 now t3 is same as this so what happens here them so we have B here and t4 is nothing but B and the same node which were either the called out T 1 or T and the next one is v5 is sine T 2 multiplied by T 4 so 2 multiplied by T 4 is the sign if I wear this which are not visible and then.

We say T 6 is a sign thoughtless prod multiplier or + D 5 so I am looking at T file and I'm looking at Prada at in the pooh and that goes into T 6 and then I say prod is nothing but D 6 so therefore this node is also given a little punk rock and then I say D 7 is a sign I + 1 so T 7 will now take this value so this is taking I plus 1 which is going to be 7 and then I say either sign t 7 so I now gets the same level and now I have this conditional it says if Y is less than equal to 20 then I go to 1 so what I do here is I have this conditional which is testing for less than equal and is jumping on level 1 and it is testing this value.

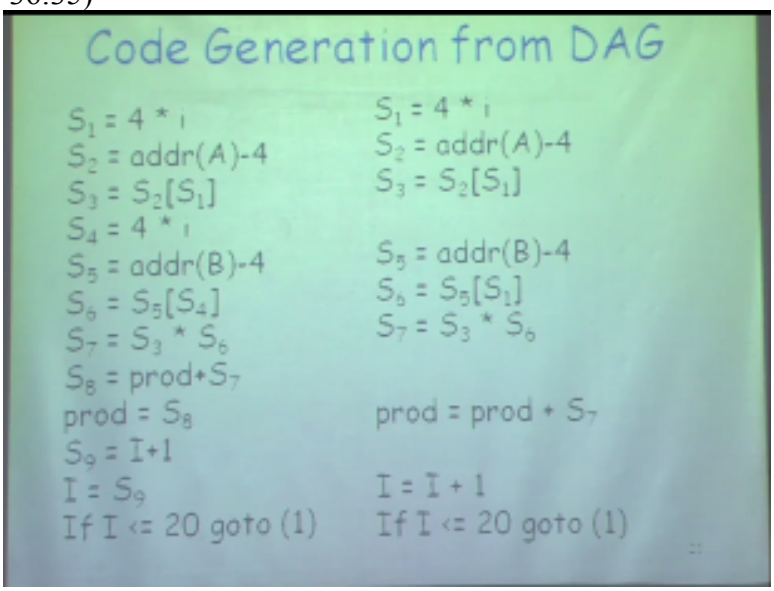
(Refer Slide Time: 35:40)



So this is how the DAG rules okay and then if I look at the poor condition and this is the state court for the two not really the day since so what happens here now here we are saying that as one s 1 is now some symbolic your source which is being a sign for it will I look at the base address okay and then I do this Eddie and I do this computation but if I do it for the dag what are the instructions.

I would say immediately you can see that this instruction will be eliminated and instead of s4 here I will use s1 but what happens to this part here this part I will say that part is generated so s8 will be eliminated because this is just the copy.

(Refer Slide Time: 36:35)



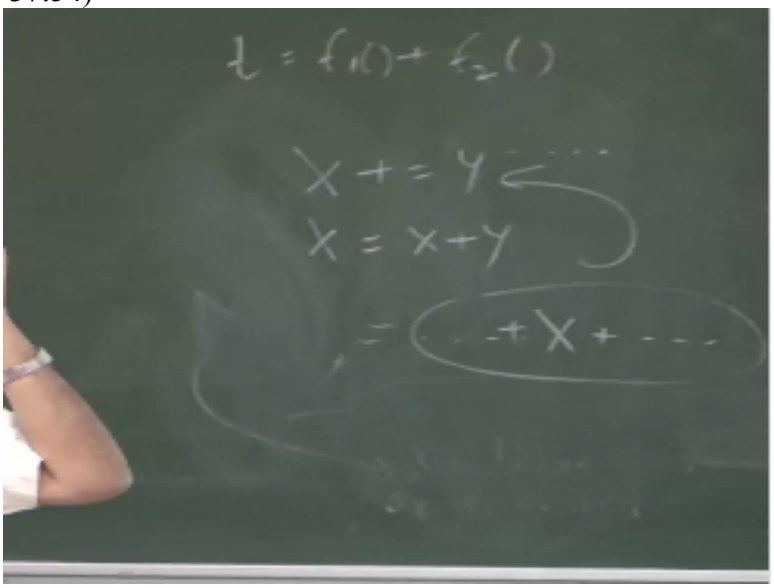
And I say probably the sign process 7 and similarly this part will say that I assign I plus 1 and this is conditional so I will be able to eliminate one of the instructions here so this is how the

whole word looks okay that this instruction gets eliminated and this instruction these two are will be 1 this two are will be one okay these and you can see that basically how we are doing for the addition and we shortly see how for peace.

If I know it for how to do it for t and you can see that I think for tag also because the way we keep it stacks we are keeping observables for this so we have seen how to do for code generation for clear this code but you cannot see how to pop please is that but basically since we are going to do that next I am showing you that basically you can handle the two things together so we see how to do this please all right.

So let us move on okay now let us come to something I just mentioned about register sphere and if you recall in the beginning I said one of the optimizations was that I wanted to change the order of execution so now when I say that is fine D plus P suppose we can see our expressions then which expression should be evaluated first supposed both our function box so suppose I write a port like this here.

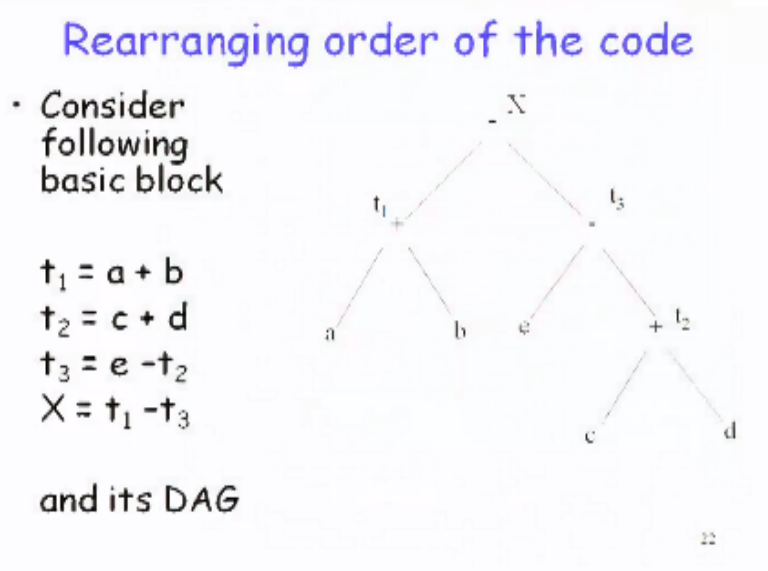
(Refer Slide Time: 37:54)



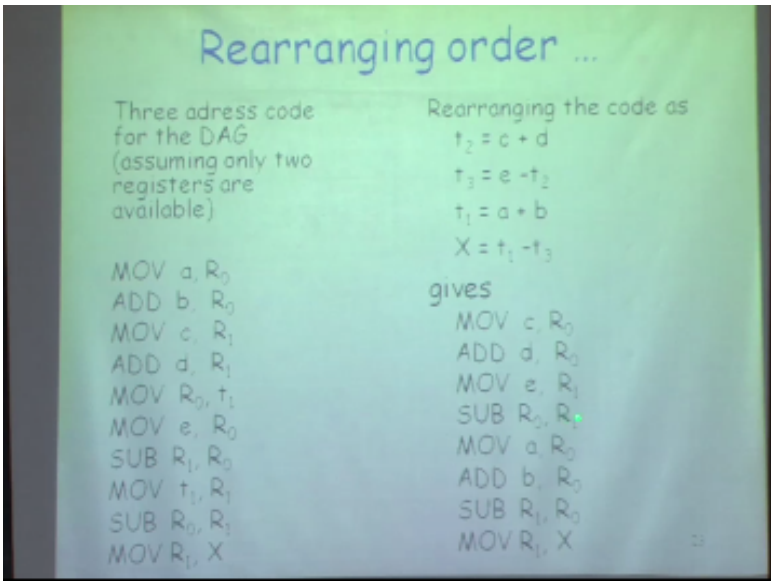
I say that p is a sign let us say x 1 plus 2 and both are function calls this clearly says do an evaluation from left to right but what is normally not specified is many times which function should be relative first okay and suppose these are complex expressions so but most of the time we do not really unless they are explicit side-effects and I am doing partial evaluation and side effects could be different yeah.

I do not care about my values right most compilers will try to do it in an order so that I can optimize my resources now how does changing the order of evaluation optimize my resources so look at the simple situation suppose I have some limited resources I have any sources now when

I start doing this computation this could be an arbitrary complex expression I start doing this computation suppose efficient.
Computation of this requires n reducers and I have only n business hey what happened here that I am going to use all the N registers for this competition but finally I will leave this value either in a register or register are if I require more register that I was changing okay and I be this store this particular value and I leave this pole so this particular argument now suppose I leave it in the register then I am left with one version once you are ready sir now suppose this also required and the business then this register can come only because of skill so what can happen is that if I change my order of execution so look at this code.
(Refer Slide Time: 39:45)



It now this code actually is interesting what I have is t1 is being assigned a plus B then the who is being assigned C plus D and E threes being assigned a minus B so you can see that this particular instruction depends on this but if you look at t1 and t1 is being used only in the fourth instruction okay now if I look at the tree corresponding to this again I mean it is are not visible sorry about this but stuff what is happening here is that I am on the left hand side I am saying that a and B are being computed and they have been stored.
In t1 and then on the right hand side e and D are being computed as C and d are being added that is being stored in t2 and then I am saying e minus t2 is being computed and that is being scored in t3 and then I say X is now being computed by saying T 3 minus T 1 minus t3 and the situation I have okay.
(Refer Slide Time: 40:33)



Now suppose that I have only I have 3 address code okay and I have only two registers available my left to right evaluation so what can happen is that and generate this kind of code of this is this was wanting to saying that I am moving into a register then I am adding B to this so this gives me now a plus B okay and then I say I move C to a registers a plus B is now available in register r0 so one resistor is now blocked.

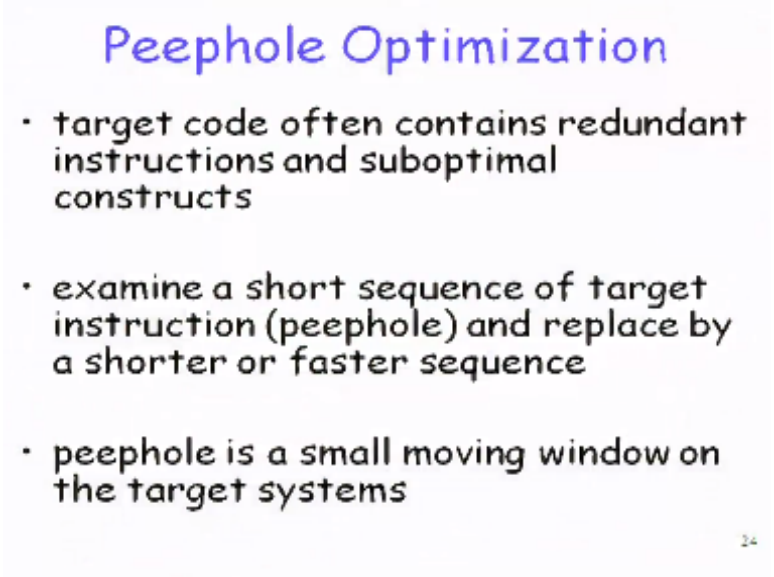
On the left hand side now I say move C to R one and then I say add now T to R 1 so I am out computing C plus B and beyond this now I will say move R 0 to D 1 now why I am doing this basically now I want to do this computation which says that I temporarily free this register R 0 and this basically is not moving the register spilling and then I am moving into a register which has been freed by this particular spilling and then I do this subtraction and after I have this done this subtraction and one of the registers has become free then I moved even back in to R 1 and then high subject R 0 from all.

So basically after this r1 is being stored in X and now if I change order of execution here suppose I say that I am first doing this computation now you can see that this is a larger expression which requires extra releases this is a smaller expression which requires one less resistant suppose I did this computation first okay then these two instructions are generating which is moving R 0 t1 and moving t 1 into R 1 these two instructions.

Would have been saved because this is basically if the discussing so what kind of code I generate okay this is now going to give me faster code because these two instructions are same and basically this is saying now compute C plus B E minus c plus D which is being computed and that is being stored in r1 and then I am left with r0 and I can use r0 for doing this computation.

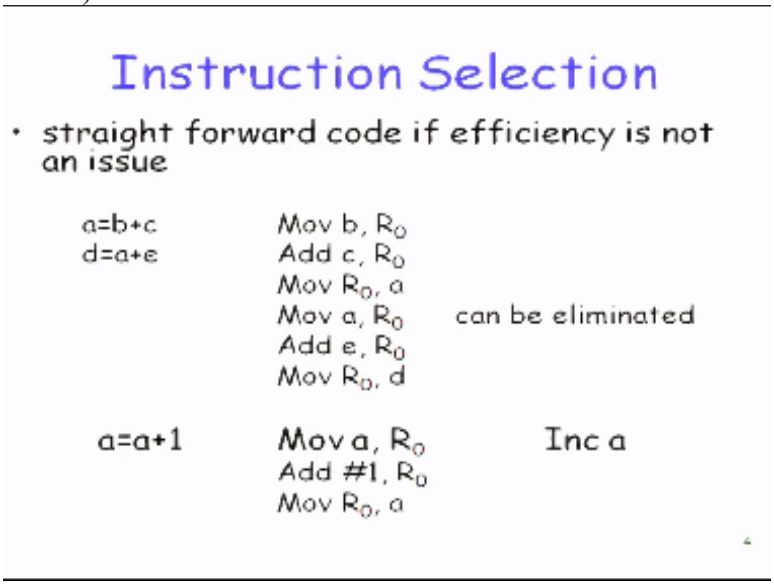
Which says p1 is a sign it plus B so a plus B is now computed now zero and then I do the subtraction so basically this is what is the register s feeling and this is what we will use the previews so sometimes changing the order of execution possible so normally there is one technique which is use this that when I'm trying to four the addition or piece I will use dynamic program and I say that I keep on commute computing the cost off.
So cost could be in terms of number of registers and say that I keep on computing the number of resources required for the left hand side and the right hand side and I determined which is the optimal way of doing it dynamic programming immediately makes it cost you your complexity computational complexity goes up.
So you can either do it in single traversal and incur this cost but many times and you want to get it optimized for it then use that programming that can find out that using computing this part into a register and then doing this computation is faster and you are not violated any code because there no side effects the computation is going to remain the same and therefore you can get now you can see that actually.
I mean so this is another thing that has to sort of register in your mind and I should completely when we take the posture for it now you may start computing for how much I am saying now any saving instructions know between yeah.
But what we have to remember is that if I come number of instructions I have common 4 + 2 6 + 8 + 2 10 instructions and when I say I am saving two I am saving to all of them which is 20% of this and if this is part of a loop which is easy to take large number of times then 30% saving people are doing nobody is saying that I had program can run on hundred percent faster or can double the speed or triple this no people are saying and percent 20% person that is a big game for people I mean imagine a situation where somebody is program is running for 20 hours if you say I can finish this in 16 hours that is it for the person and all large concentrations program back to Eden in those days .
I meant in terms of hours and it is like a new type of program say it or return and desert you can download and then you come back after see then check user normally this is what happens and that is there all this optimization stuff so even a small saving here 5% 10% ending is adverse amount that is the number of people on that nobody is looking at kind come on the speed of my program and that is something unless you have very special hardware is on single machine optimization unless you have some very specific program be killed through this it's so really need order is important but there is not the only thing we want to keep on to do more optimization.

(Refer Slide Time: 45:41)



And thus more optimization is what it was people so I just introduced this and then are you can take a break so basically what people optimization is something like this I look at where this for so let me go back to an example we had listed so what we want to do is I say that here are six instructions on okay.

(Refer Slide Time: 46:13)



Let me have a small window which I quality full name and move it over this sport and when I say move it over this port I say that suppose my port is size two okay then you say that ok move it over this code and said look at these two consecutive instructions and say that can I convert this into a possibility yes now then you move it over this part of the code ok you will see that

immediately notice that we are saying move r0 into a register and then you're saying all moving to a register.
And we know in ports have agree to this kind of calculates now when you say I am doing this kind of store and then I am loading this value second inspection can always be similarly if I say that my window is of size three yes then I can experiment with this board and say oh I know this template this is just adding a constant value to register and I can then just replace this by a single instruction okay.
So t4 optimization in general is a technique where I examine my code after code generation happen I know who any of the optimization products for them and once I have this code then I define code full of size two three four and normally port have experimented and already because it does this is like nominee if I have a t4 of size four to five then an odd number of optimization can be done by just making these codes.
So what you trying to do if you create soccer fan place and then move it over the code and whenever you find a calculate like this let us replace it by a faster instruction sequence so it is possible that I may take these two instructions replace this by one I will take these three instructions replaced are you and so on yes and with two kinds and group quality of your book okay so what we are doing the next classes we look at few transits of equal optimization and then we also look at code generation form for the addition of how to do for the addition for peace like we did for teens so this is today and last class in the last class we will discuss this things end of the classes.

Acknowledgment
Ministry of Human Resources & Development
Prof. Phalguni Gupta
Co-ordinator, NPTEL IIT Kanpur
Satyaki Roy
Co Co-ordinator, NPTEL IIT Kanpur
Camera
Ram Chandra
Dilip Tripathi
Padam Shukla
Manoj Shrivastava
Sanjay Mishtra
Editing
Ashish Singh
Badal Pradhan
Tapobrata Das
Shuubham Rawat
Shikha Gupta

Pradeep Kumar
K.K Mishra
Jai Singh
Sweety Kanaujia
Aradhana Singh
Sweta
Preeti Sachan
Ashutosh Gairola
Dilip Katiyar
Ashutosh Kumar
Light& Sound
Sharwan
Hari Ram

Production Crew

Bhadra Rao
Puneet Kumar Bajpai
Priyanka Singh

Office

Lalty Dutta
Ajay Kanaujia
Shivendra Kumar Tiwari
Saurabh Shukla

Direction

Sanjay Pal

Production Manager

Bharat Lals

an IIT Kanpur Production

@Copyright reserved