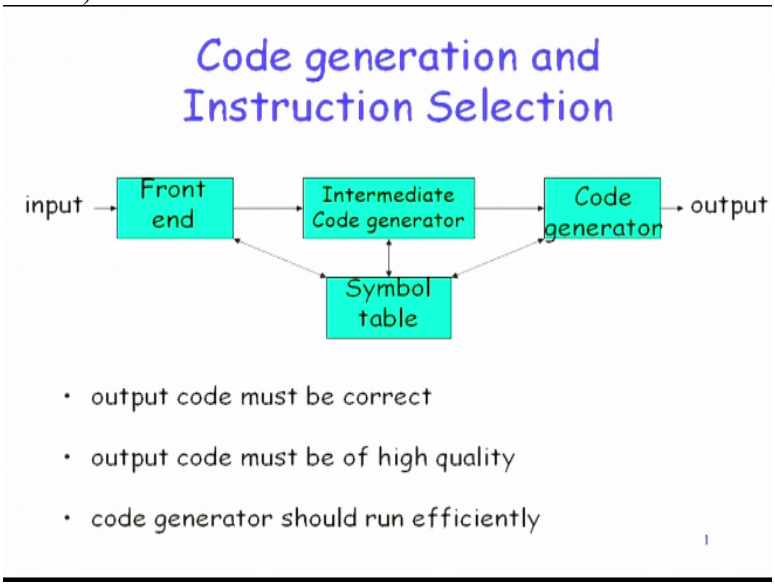
Indian Institute of Technology
Kanpur
NP-TEL
National Programme
on
Technology Enhanced Learning
Course Title
Compiler Design
Lecture-28

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

(Refer Slide Time: 00:14)



Good morning so let us begin with last state of compilation and machine code generation and what we are not going to look at next in the remaining classes how do we now convert and the only issue which is left now is that we have already written all the code generations and we know that the code is going to be close to whatever machine and you pick up which is close to almost risk architectures and we are just going to convert it into copied syntax and which is the register optimization.

What we will also simultaneously do is we will see that the variables which we want to minimize as far as possible and if in case it does not fit into the list then we will have to see how to skill the register and load it back what we are not going to do is look at register optimization problem that is something which goes really beyond and which is independent activity of graph based algorithm and so these are we will look at functional character and not worried too much about it as far as registers are concerned for the optimization.

\

So this is really what you see here as the specifications that we have this front end and intermediate code generator and the information on the symbol table and now we want to take information from the intermediate code generator which is a three address code and the information from the symbol table and the complete syntax of the machine and what are my requirements.

So obviously the output code must be correct and that is the biggest functional requirement we have and also the code must be of high quality and the code generator should run efficiently and these are the writeable functions and what are the inputs and outputs we have so input is an intermediate representation with the symbol table and the assumption we are going to make here is that the input has been validated by the front end because the input is not correct the language specifications and the intermediate code generator and so the process could have terminated right here .

So at this point of time and what about the target programs the target programs can either be an absolute machine language and this is fast for very small programs so we have already seen that already how do we generate machine code so we have seen that type and the numbers which are Pc values and we can now generate absolute machine code we can also generate relocate able machine code which is normally what we do.

Because the advantage with relocate able code I that you have multiple functions and that in case of absolute machine language code so absolute machine language code even the linker has been absorbed that is not something which is desirable will work only for small program and then we may have an assembly output which obviously will require the assembler and linker.

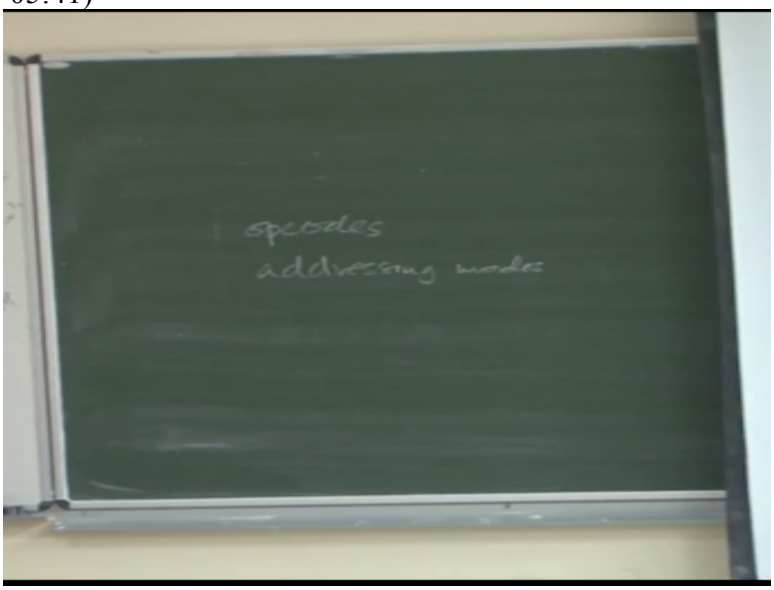
So normally most compilers either we generate this or we generate this one is really requires a compiler and I do not know how many of you have used programs like turbo C compiler how many any kind of turbo compiler most people do it at the level of schools not even in the graduating school and if you recall by installation it asks for two three options it tasks whether you want a tiny version you want a small version or you want a bit version or a large version and select the option tiny version .

What with installs is compiler which actually generates machine code it will not permit you to write multiple functions for multiple files and it just removes all that and compiler becomes really fast but then you can only have a small program so we are only dealing in this space and will not worry about getting the absolute machine code okay so what are the issues we face now

these are the issues I am flagging so that when we actually start looking at it will realize that these issues are important .

So first issue that comes in uniform now since you have done assembly language program if I say this is and assume you have a machine and you have operation codes and addressing node .

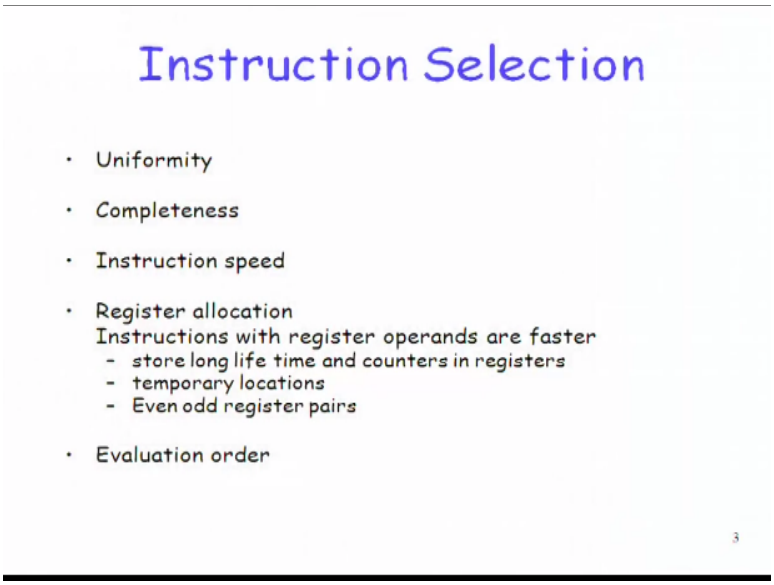
(Refer Slide Time: 05:41)



So for example if I say I want to do addition say so addition will work only with registers or will work with indirect register offset will work with index offset may not work with the register so normally we will find if you look at machine specifications there they will say that certain operation codes are not going to work with certain address nodes that I immediately brings an issue of poverty say if I pick up an operation okay.

What are the access and the addressing node is nothing but how can I store the variable this immediately says that be careful while choosing awkward because all addressing nodes may not this situation where I say that where I cannot find the machine operation code and cannot find an addressing node so this issue came up very early and machines did not have an operation code for doing addition and division .

(Refer Slide Time: 08:20)



And can I get into a situation where I say something cannot be computed is there a proof for that my experience shows that we will not get into problem at least from the search point of view now that still remains an issue that there is no data to be that compiler will always be able to generate the functional code okay and the instruction speed is another issue that comes up that I can write multiple programs for doing the same computation .

That means I can generate multiple ports at the target machine and which will do the same computation now which one is better sometimes you see that some programs on will be faster some program are going to consume less number of memory cycle and there are more issues like again also we may talk about the program which consumes less energy that is again optimization we will only look optimization with respect to resources .

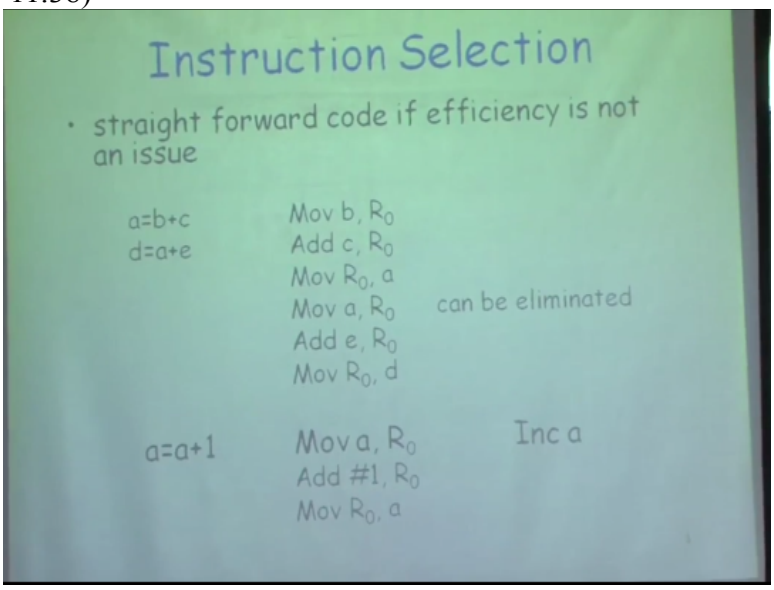
Then register allocation has become an for issue us then instructions in the register operand are going to be faster and the long life time and frequent variables are going to be in the register and they are also going to be in the temporary location and in some machines I will also have to worry about pairing of even and odd register almost every machine has an Intel processor you will find that if I take a long word .

We cannot say that take the register three and four and note this word I have to make certain way of register so even for this word so even when I have more registers available I may not be able to load certain variable so the evaluation order of instruction remains an issue that if I have a large expression being evaluated what is the evaluation it says that I have to compute from left to right .

And once you have converted that into it some kind of abstract syntax tree and optimization results you may find that if we not be beneficial to compute from left to right okay it may be

beneficial to say that first compute other side store the result somewhere and that may conclude fewer resources okay so again we will see an example of that as we keep on moving all these issues will be at some point of time based .So let us look at what code generation and at this point of time what we are looking at is concrete syntax of the machine .

(Refer Slide Time: 11:38)



So what we may have here is something that if is say that we have two assignment a is being assigned b + c and d is being assigned a+ e then I can generate very trivial code for this or what I need to do is that I for the first three address instruction I generate tree machine instruction and I am using a generic syntax and you will recognize as soon as you read it so this says move b to a register and I am not using addressing nodes of a b and c and I am assuming that then I just use that .

And then I say that when I come to the second tree address code and I am saying move register to the end and move the register to t okay and so again symbol names can be different to different machines ad this indicates that in which direction I am moving my data now you can say that I can do some simple optimization and I can do a optimization that a is already in the registers so I do not need to have a load again and I can immediately eliminate this instruction and reduce instruction from number of instructions and the construction reduction .

So if you count in percentage you will get some 16% on the screen actually what happens is this is a part of a loop and that loop iterates r number of times so one is not talking about over all program and if I have an instruction like this which says I want to increment a by 1 then I may have an instruction says move a, Ro an store Ro and many machines will also provide me single instruction and what happens in some machines is that if I increment a that a if it is in memory .

The increment is not allowed so normally what will happen is that if I have a loop going on and this is really the loop counter then the assumption is that first I must have loaded this and suppose this was already in the register and every time I say I want to increment the loop counter by 1 and if you want to optimize you can look at these parameters and you can find out the possible code .

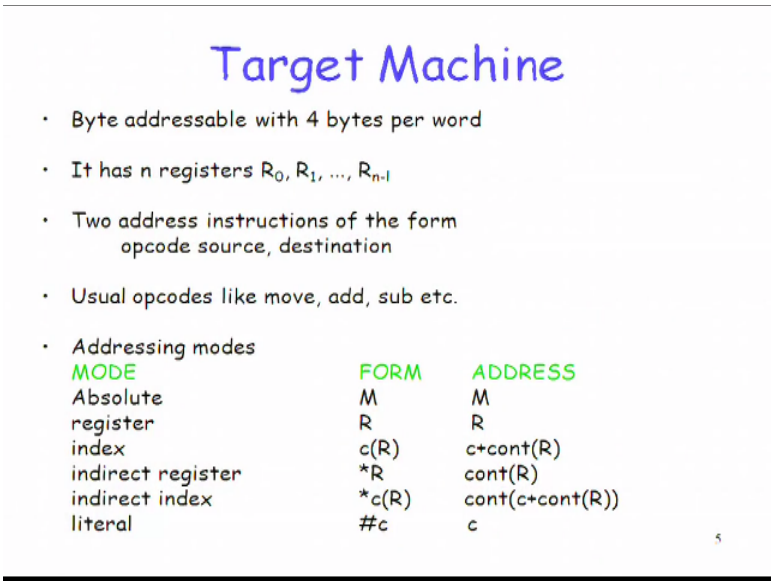
And you can find out that you will get into situations like so most of the compilers will take an approach that generate trivial code using a trivial algorithm which is functionally correct and we can optimize it by using a technique called e code optimization code can be equivalent ad so we will look at e code optimization will say that I have sequence of load and so you can eliminate the second load but the value always remain in the register and this way I can do straight forward generation.

And the efficiency is not a issue and you can recall the overall compiler structure I gave you and there was an optimization at the front end and back end and then there was a post code generation optimization. So at every step I talked about optimization and if you have three address code instruction.

So let us assume certain target machine so now what we are assuming for target machine is that we have byte addressable machine instructions with 4 bytes per word and depending on the machine I have register and two address instruction are of the form operation code source and destination and which will provide and then we have operation codes like move ,add and subtract and what are the addressing modes and I may have absolute and I may have value in the register and I may say we can use index variable .

When i say that the register has an address or I may have an indirect register and I have a indirect index register and I am say that and you will have machines if we say that this operand then this op code is different each to be stored in 4 bytes then this is going to be different so there am be instruction like so this whole instruction gain can be verified.

(Refer Slide Time: 16:36)



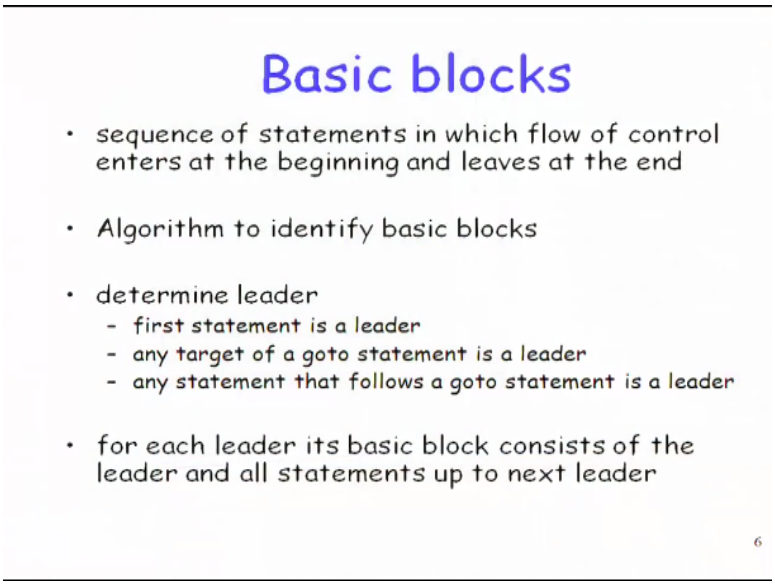
So most of the compilers will take an approach that first generate they will both using the tabular so you can see that we have machine and restrictions are on the form that I have offered source and destination which will provide you of one of the destinations this is we have now let me give you all uniformity you will find that if I have any stretch it will find their machines what is different but it needs to be stored in four bytes .

So there will be instruction like if you have control like instructions at immediate an immediate means that this whole instruction so what they did was something really tricky the fact so normally your operation code is going to be stored in four bytes but since as immediate we know that operand can only be fitted in three bits that the reduce size of the operation code.

And the part of the operand was encoded in that operation code byte that means in a single fetch we will be able to get both the operation code and one of the operands okay but it cannot so now and why this instruction is provided you can see that most of the time what is going to happen is that if you are implementing your loop down then you increment it by a small number you do not say we increment with every number so you may increment by one or two and normally this kind of instructions therefore will work much faster .

So architects normally will provide you different kind of instructions which will do the same thing on different data sizes and different address nodes actually take an instruction set this is my generic machine model and this is my machine model.

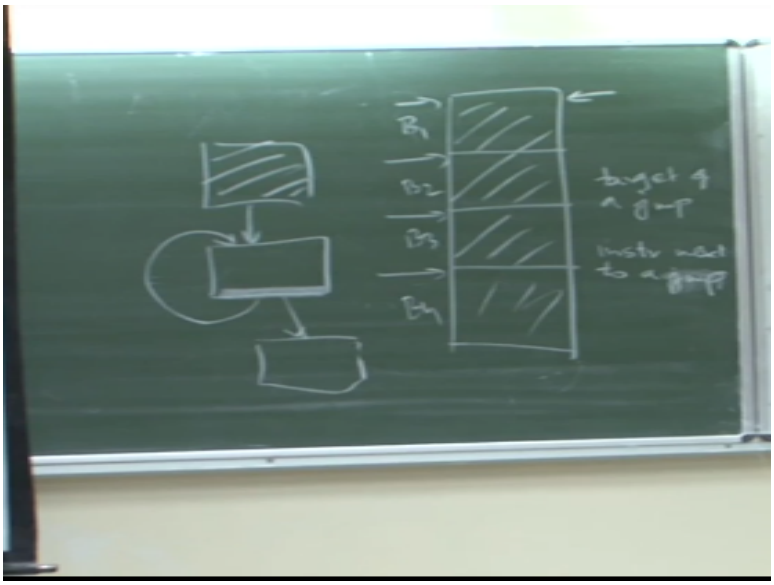
(Refer Slide Time: 20:00)



Now let us look at starting of the code generation and the first thing I want to do now is assuming that I have generated create the code I want to break this re address code what is basically known as basic block and now what is a basic block is a sequence of three address code which can have a single entry and single exit in the lectures step inside it jumped only to the first instruction.

And once I start executing the first this is only the and so in basic very simple just from the definition what I can do is I can start standing my period respond and mark certain instructions as leaders in what are the instructions which are the leaders first instruction obviously is going to be beginning of a basic block then I say that anything which is target of a job it has to be beginning of a basic law any instruction which follows the basic meaning of and then I say that starts these instructions from a leader .

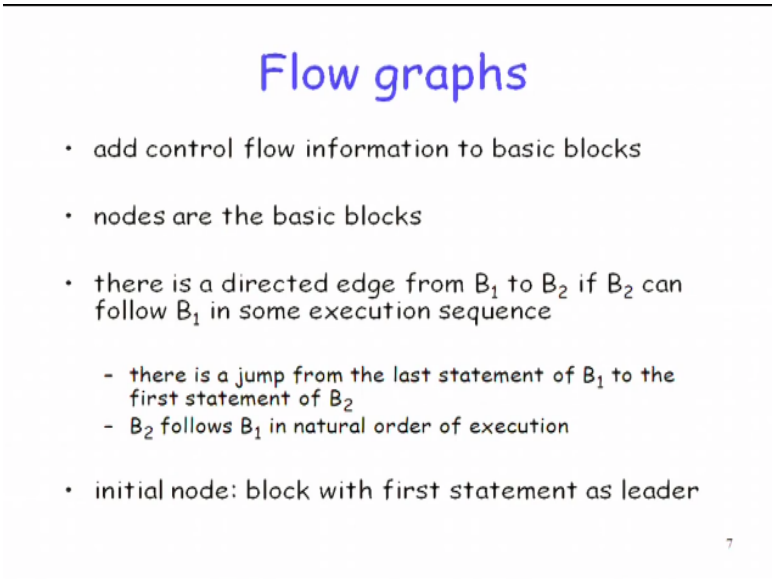
(Refer Slide Time: 23:07)



The instruction prior to the next leaders that is your basic block so basically what I am doing is I am taking my create code this already this definitely is a leader instruction no doubt about that that is the beginning of a basic block then I say that target f a jump has to be the beginning of a basic block cannot do anything about it and what is the third thing instruction next by instruction next will jump suppose as soon as there is a jump that must be the off instruction .

So next instruction must always be beginning of so once I marked all these leaders then what was it and these are the leaders I mark you in this body so first thing I do is I take this and once I have constructed a set of basic blocks then I say so what do I do now okay so just initially so first statement is a leader any target of a go to statement is a leader and any statement that follows a go-to statement is also a leader do that and each leader and its basic block consists of leader and all the statements of to the next people right now how do I add control flow .

(Refer Slide Time: 23:51)

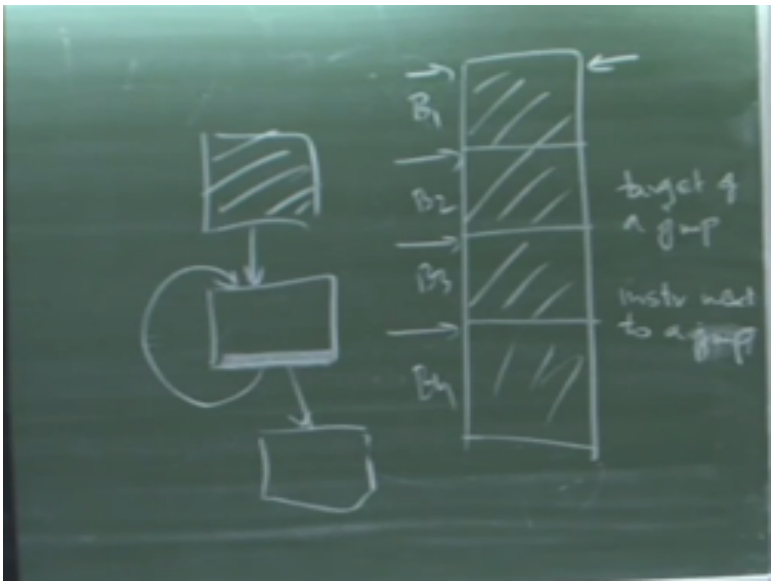


And I say that I am going to now actually convert this into a graph when I say nodes are going to be each of the basic blocks and how control flow go will say that if there is an explicit jump from block b1 so suppose I say these are my basic blocks b1 b2 and before if I say that there is an explicit jump from b1 to b2 then there is going to be an edge and if there is a natural flow of support to be and ask me to you will hurt your neck the way as you are tilting you can go back to your room and sleep there peacefully.

So basically what we are saying is that in the control flow graph in the flow graph and take all the basic blocks as nodes of the basic blocks and it is a directed edge from b1 to b2 if you and although even in some execution order or if there is a jump from the last statement we need an example for this so if I just say that look at the situation like this so what are the kind of graph I can get what may happen here is that the last statement might not be jump stations then I may have an edge from b 1 to b 2 because there is jump from the a statement off base across the beginning of this block.

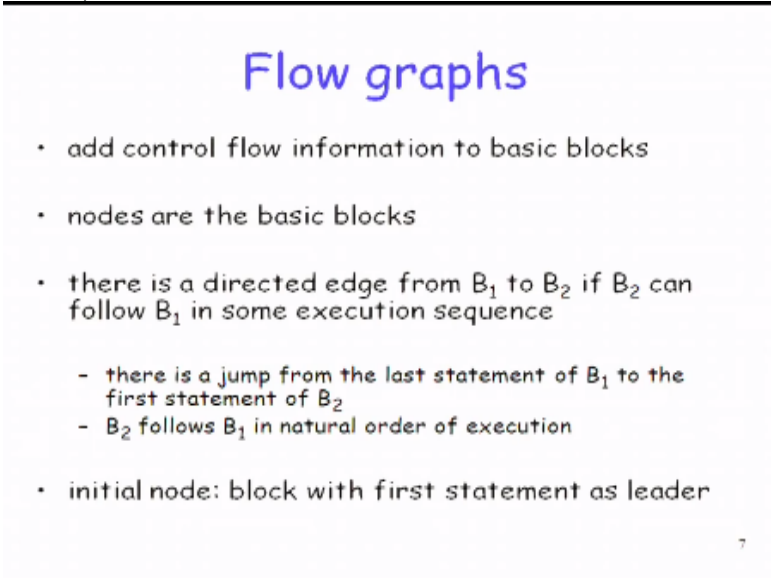
So this construction becomes a leader if I jump or conditioner here and all further code generation will be done off the program if I am given idea is that as far as basic flow is concerned I mean we have and as far as the last discussion.

(Refer Slide Time: 26:00)



It is happens to the jump destructions which is either a conditional or un conditional jump okay then I will able to generate a different kind of instructions okay because jump instructions we are the step instructions are going to be apt on to machine instructions are slightly different.

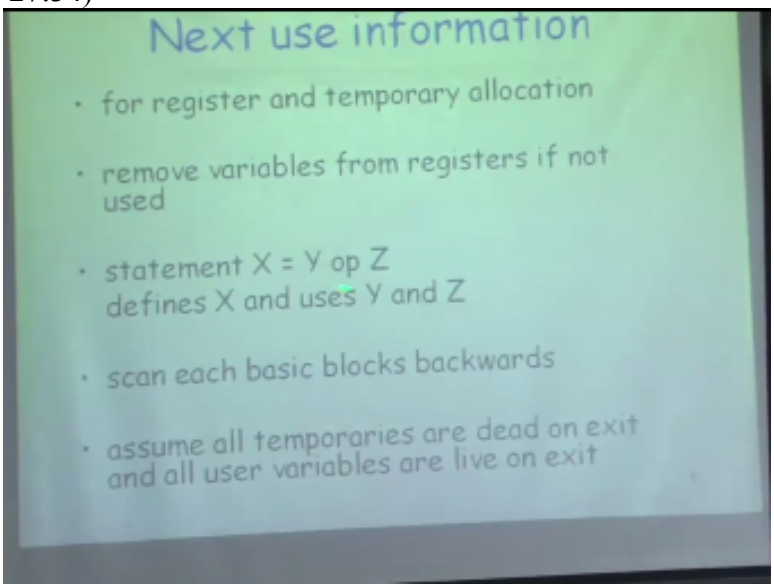
(Refer Slide Time: 26:21)



So this making free at the code converting with code with a basic law and control the graph as well as basic blocks any problems and any questions re there okay so there next move on and what we have to do on this we want to know when this type of register optimization okay.

Register optimization that saying that it is from the value is already it is register when I want to use it okay I want to use the same location and for that I compute what we know as next to information and next two information is going to be computed of basic then why I am computing this only the basic flaws it has toed not have the control of flow is simply equation and I can a

then I will able to generate a different kind of instructions or strictly sequential I can scan all the instructions and I know that I do not have a process of jump instruction okay.
So what we do first thing is we actually use some instruction and say that it is instruction I had okay and I say that this is I instructions it is defined variable X and using variables Y Z this okay this is the nomenclature value so basically we are saying that I want to now optimize this registers.
(Refer Slide Time: 27:34)



And I want to do loop variables of registers okay suppose I take some that we devalue so XYZ that could either be user defined variables of they would be temporary variables can be generated by the compiler and when does compiler get temporary variables so whenever I am generating clear this code you will find that compiler generates all that variables P 1 P 2 and so on finding the variables are all the user defined variables okay.
So compiler with the generate as ten hopefully I am going to put all these in as far as possible you should registers or I optimize that I minimize number of entries and then use fewer registers okay we see that earlier minimizing the query optimization translation I can keep all these attributes and active use only for the lifetime of this attribute only life time of attribute only and then if you recall one example we had in the life checking where I was trying to define declaration of an array.
I reduced all those activity equations into that P1, P2 and I said I can use only R 1 R 2 and R 3 and then we realize that some of them are copies of each other and I could reduce the whole thing to the registers okay so we have already done some kind of resource optimization now

what we want to do is to systematically look at each basic block look at how many temporaries it uses and minimize number of some things.

So that I can reduce number of registers are use I do not have to register spinning because if I use too many temporaries that I am not have sufficient number of registers so what do we do we say that this statement which is X inside Y of that is defining X and is using quiet this and you say that any variable register from to remove if it is not going to be yeah so what we do is systematically suppose I have this instruction and suppose X in a register okay.

And you on this point access no use so if I look at a sequence of instruction and I find X should not be used here I can immediately free all the resources which are bound with that particular exit now when I say three what that means is that the same resource can be non use for another purpose if that is what free means so scan each basic block backwards and you will see by this when we are doing this and assuming.

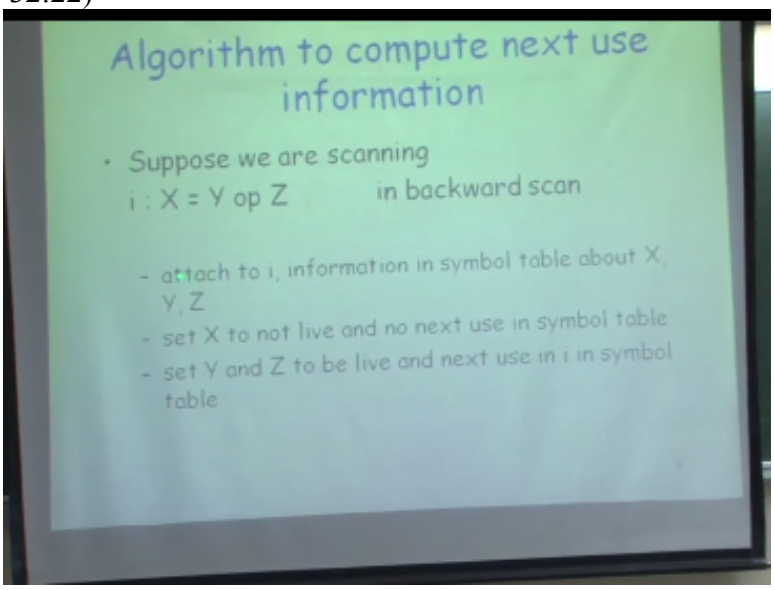
That all the statement is not networked with so register allocation and register optimization the I have going only for the basic that means when I entered basic block no variables one registers and when I exit basic block no way to do something to because now there is another technique which makes if you learn you learn in some of the course not is this where you say that I want to do optimization that was the basic box but optimization across the basic block requires a general technique called dataflow analysis.

Because I do not know which parts may be follows so we reach this particular basic form and to follow this path or sometimes I mean follow this path and I have to optimize it across on the block and idea of a basic block is that you have only one path so I would not have to worry about to analysis at all that is what makes it simple so what we are going to do is we're going to assume that all the templates.

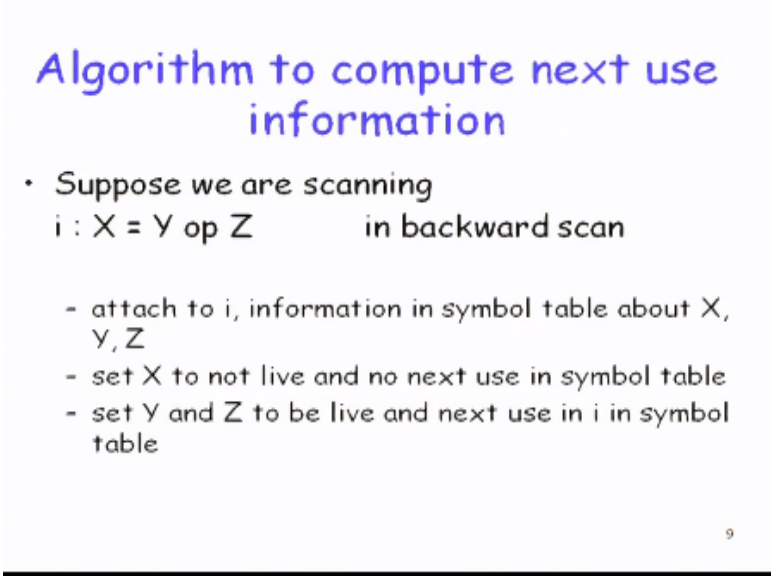
Because we are doing optimization only within the basic block that is all the temporaries are read on exit and all user critical and all the user variables live on exit user variables are have a live time which is the whole program but can templating have a life time and which is limited by the boundaries of the basic functionality and if we do dataflow analysis we can extend this and say that these templates are going to live even across the basic box so we do not worry about so our boundary conditions clear what we are trying to do here just know say something them okay.

So what we do here is that for each basic knowledge because consecutive okay it is a very simple all it says is that I can keep the statue of each variable available in this simple table and this will

say that whether the variable is live or death okay love or death means whether it has any variable because the future stack is like any variable which does not have future uses that okay.
(Refer Slide Time: 32:22)



What we do is that suppose I am scanning this instruction then what are you what would I do I will say that two I s I am going to attach information in the symbol table about variables X Y Z okay and I am not only worry about the user variables and then now I will say that beyond that point X is not life and has known exclusive symbol table and to say why is that could be like an excuse okay so first one security understand what is happening and then we can go to separate them I will give an example and then what concept we are able to see what I am doing thing so suppose I say that I am scanning this today and let us say I have at this point okay see only point which are this when I am at this suppose I have a this point what are this point mean this okay so this point right there X and Y I am going to use there.
(Refer Slide Time: 33:30)



So this is what I say that Y is that alive at this point and they have using I use their and this is me that X is not life and it has no let us use in the symbol table that whatever exits is be there whatever X I had prior to this and this moment we are going to be from this point onwards this is the exit agreement all prime axes are going to be now that this is one change so if I take any instructions.

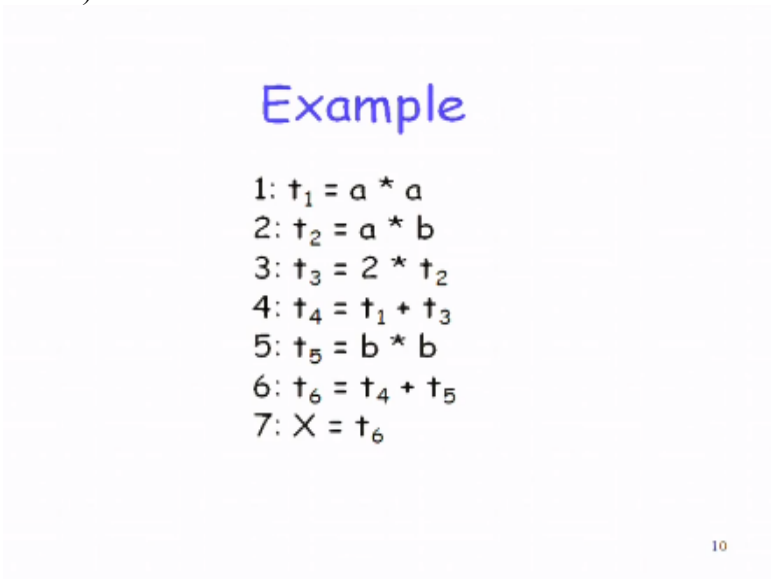
I say then let me find out status of each of the variables at this one it just this is second part this is section is basically at this point my information is that whatever definitions of my head prior to this are no longer valid because beyond this point the only definition of X I had but which is I know the value will be used in a basic form and at this point I am saying that y and z are still life and they have been used and obviously.

I will come to this point and what will I say what can I say about Y Z at this point or not so at this point after this instruction has been excluded what can I say about status of X, Y, Z values there so at this point is X life yes so exactly the point I am trying to behind the is why I am standing that was so if you just stand forward okay at this point you cannot say anything about X Y or Z unless you see the instructions.

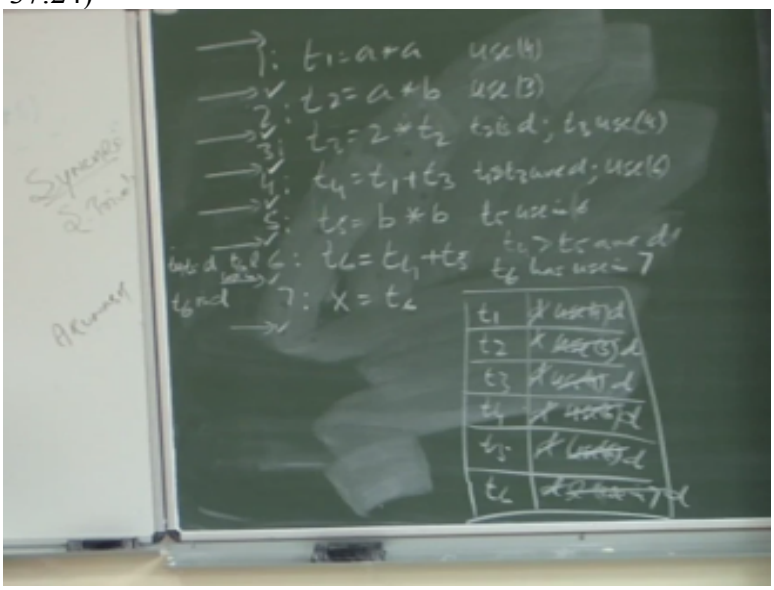
Which followed and standing backwards I will be able to see that information and that is the reason I actually stand like both backwards and not for the first time I will find out what is the use of available and then I find the most recent definition of that variable so I do is I start there for standing to this direction and some point of time I will say that I say that X is being used then I immediately said X to live and say this is the definition.

I am going to use in an inspection and then I will try to find out the most recent regulation of X and say this is the way the definition I am going to use either instruction.

And there I try to find out the most recent definition of X which was used here and that is the reason I do a backward not forward step because if what forward can I cannot say we determine anything about X Y and Z is that everything right so it is logic here why I am standing my code backwards and I am setting this type of status so what we can do is we take an example I take you through this algorithm and actually show you how do we optimize okay.
(Refer Slide Time: 36:09)



So let me kind of write a piece of code and okay that mean copy and this code and then I will go back to the previous volume and what is the example of so we were go with algorithm and the example remain in this point of you a star and t2 a star b and t3 2 star t2 and t4 t1 star t3 and t5 b star b and t6 t4 t5 and x t6 to a this is a code.
(Refer Slide Time: 37:24)



This write to be a symbol table and how to variables I have symbol table and all I have user table and how to reduce in that table I am only worried about the template available I do not know any optimization of the user visible so AB are user variables and t1 are AB are user variables and even a star template so why say here is that I have six and entries here corresponding to t1, t2, t3, t4, t5, t6 okay.

And then I will say that let me can reduce greater than this one okay now first thing I will said that is at this time all the variables are getting there is a greater than because I am not looking at live of t1, t6 beyond this point so that is the point what I will do this I said all the variables are okay look at the variables what I am doing here okay so I am doing here is that when I stay which is I say attached to y now.

All the body information is simple table about X Y or Z so now I am saying that I am connecting to seven information about pieces and information about t6 what are the information about t6 my single table or information about these single table attach to the user very cool so I do not even worry about this okay and once I have done this then I say that set X to not life and no x into single table and now.

What an X and X is the left hand side which is the user variable which is not actually variable so I do not change anything in the symbol table and this is that y an Z to be live and it says next is it is equal to I in the symbol table so on the right hand side I have t6 so what I do is that I said t6 is live then I say use in okay what I have done so what I have done is something very simple.

I started with my simple table we have all the variables are these this are the symbol table are the status here then I just follow these three steps.

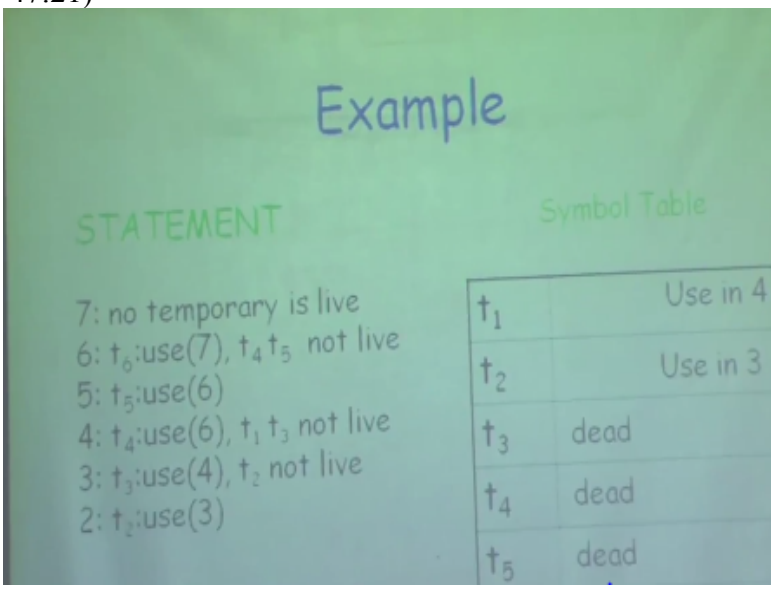
I said seven I am going to attempt information about these it is t6 is there that was the symbol table and this order is important and then I said that set X is not live so X is not a user variable I could do anything about it sorting and symbol table and then I say that set Y that is next to the symbol table so why so I have said this is not on this program it wants any backwards.

I have done to this and I have done to this formal of this okay this is so first this first back here because the first step is not clear then we get lost in the remaining class here the first step is clear to everyone okay now I say again I just follow this as I say that I am going to attach to six information about t4, t5, t6 okay so what are the information about t4,t5,t6 this is t4 and t5 are dead and t6 is live is use in and losing so this is t4 and t5 are dead and t6 has t7 these are the symbol table.

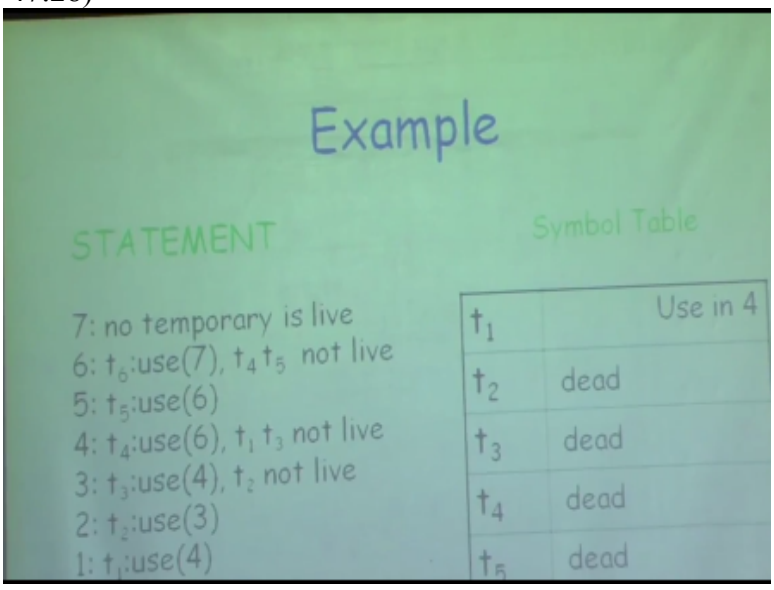
I have in it I am just popping the informations at the outset and then I say that X not life so what is my X here t6 so again I say that I will remove this I can say that t6 and then I say that said why

you got to be live on exclusion in symbol table so now I say that t4 t5 are life and this program on and they using t6 t4 and have moved in six right everyone is comfortable up to this one same thing I say that I am going to back okay I have done this program this point.
When I am come to this program point what will I do same thing okay I will handle the attach to find information about variable and the symbol table okay t5 is temporary variables it and then at this point then immediately I go and start taking information and what do I say that set X not life so I say why t5 this one the finally comes and there is nothing that I can go right inside Y and Z the variable are the symbol tables okay.
Says while and then and this happens to be do the variable Y just because this so I come to this combination and what is this combination of this is saying that attach status of even t1, t2, t3 and t4 to this statement so t1 and t3 are dead and one of the status of d4e for state has a new t6 is and using that Information that reduce that using t6 okay then how do I change my simple table at this time.
I say that look at the second one I say t4 is not life okay so T 4 is again to be reduce that and then t1 and t3 are live and losing four okay so t1 has t4 were we get the information before a symbol table so I am done with this one okay so now I come to this statement and I am saying time to attack statement of a each variable so p2 is the only variable combined with variable.
And I do not worry about A and B and this says t2 is implies of so I am not come to this t2 is assign 2star t okay now t3 is t2 star 4 this is the information and on and then I will come to this program point and what was change look at this point now why do I say even is what is my simple t2 is live and using t3 okay then I have done to this program point.
I attached the information of t2 this statement and now and t2 has assume this t4 okay and then what we have to do after this I could saying that t2 is and then t4 are the variables I do not worry about it okay.
And this statement I have attached the information about t1 and t4 and I can come to this point with this can see this program point and now it reduce I say t1 is dead a is okay that in my single table you can see that all the variable are there so I started with an assumption that this program all my computing are dead and when I enter again my computing are entered okay what is the point captured is actually something very interesting.
It has captured like that information that saying that say thatT1 has a losing before that is the last use that means lifetime of t1 is only up to this so this is where we have to define where to this is where it is used beyond this point given as this rules and this says that equals being defined here

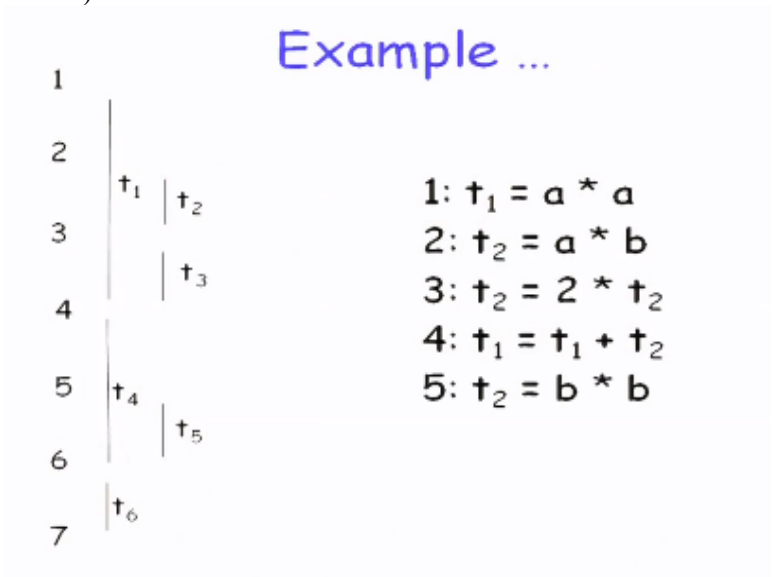
and was used here beyond this pointt2 have no use yeah similarly this says that d3 is defined here.
And has a loose in t4 for so the last t3 is here that is defining the lifetime and that in the incase P 4 is being defined here and is being used in used in t6 it so that is where the lifetime of t4 is and here I said if I was being defined here and it is even worse than 6 so lifetime of t5 is here and this is T 6 is being defined and be used in 7 so that is where the lifetime of t7 now if I say I want to find of certain resources with this okay so more that I am overlapping like times and I can find the same resources with this it Is all in this look at this now.
(Refer Slide Time: 47:21)



That if this reporter has identified attached now all this information so we have already gone through this yet so it skip this part and stake away we take to the situation okay.
(Refer Slide Time: 47:28)



The final situation final situation is something like this that t1 has a lifetime which starts at 1 and which ends t4 and then t2 who has a lifetime which starts here and ends here and t3 another lifetime which starts here and ends here t4 and each other has a lifetime it starts here and ends here which is go to 6 and t5 is live time is t 5 & 6 and t6 lifetime is routine successes okay.
How many resource I need six resources top of this so you can see that this is no overlapping this is non overlapping I mean actually to the resources so if I now do the register of a location I know that actually I just need to registers I do not really need six registers to do this composition and what is it that I actually have computed this the computation is saying X is assigned a squared plus 2 a B plus B Square X plus B whole square that is look at here.
Then I conclude it to be and then I multiply that by 2 so that is B and this is b square a square plus B plus B Square and that is what I was computed and now if you see I mean logically you can see that the as soon as I compute the first one okay I can store a square to the register and then I can do the occupation of saying that compute now to 2 star a star B heat that is register.
At the two and I am left with one more register where I compute B Square and then whatever was the first one at that will match so basically now I can re write the my goal board by saying that it let me reduce the number of register is so this point of time is and I say t1 one is a star a so t1, t4, 46 can be in the same live and so t2 is a stat t b and t2 is 2 stat t because beyond this point if you have to use so I can use we on this point.
P1 and p2 and moreover have been therefore can use the same variable and that I and say beyond this point t1 has no lose t1 can be assign t1 plus t2 okay
(Refer Slide Time: 49:38)



And then I can say t 2 is assign B Square and then I can say t 1 inside t 1 plus t 2 so this gives me because you can see so basically t 1 before the t4 and are all have t1and pieces t1, t2, and t3 and

t5 have been on to t2 okay so instead of resources I can go with only 4 resources and what I have done has just computed the life time information of each of each other variables okay.
So basically next to the information is saying although I compute my lifetime information and then actually we need my registers and we see that when we go for the operation if we will use this next we will use information and then losing the specimen information I generate code where I will be using the same register again it is registered as no use and therefore can be used for system.
So here what we went through what we went to our by doing next use information over a basic law I had computed lifetime of each of the variables and once I have lifetime of each other variables all I just do is find out which are the non overlapping lifetimes and use the same resource okay.
Questions however find the life time so they just think that even has t1 has using t4 this is giving me the last to so t1 is started from here and is being used here this is the live time that beyond this point it starts immediately after one and ends with testing of the course so this line is you piece so that is why I am scanning backwards that I know what is the last piece.
So suppose I had a use of t1 hear okay it then this I will have a testing that t1 is using inspection six on this so that is also so that is the information that we have common I would never have made attend it okay so that is I am scanning backward looking at the last comes and here you will find a simple activity and the instructions is here which use is t1 which is say that the same value you will find that.
When I am taking a symbol table t1 and t1 develop up there t1 said that live time of t1 is I will set down t1 is has use 6.5 and that information carry all the time okay right so let us break here today and the next class we continue with the class use this information of code generation and the if the next class now Tuesday and now we do not have a class in the week end we finish our 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