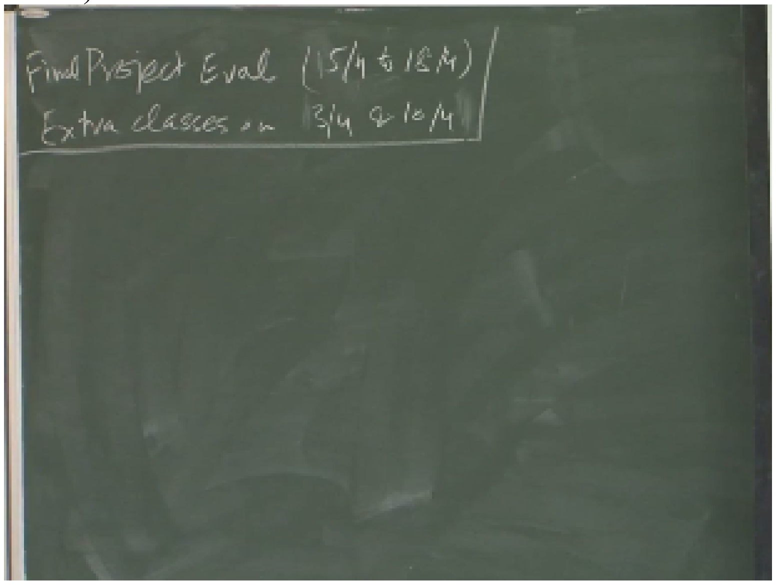
Indian Institute of Technology
Kanpur
NP – TEL
National Programme
On
Technology Enhanced Learning
<u>Course Title</u>

Compiler Design

Lecture – 24

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

(Refer Slide Time:00:20)

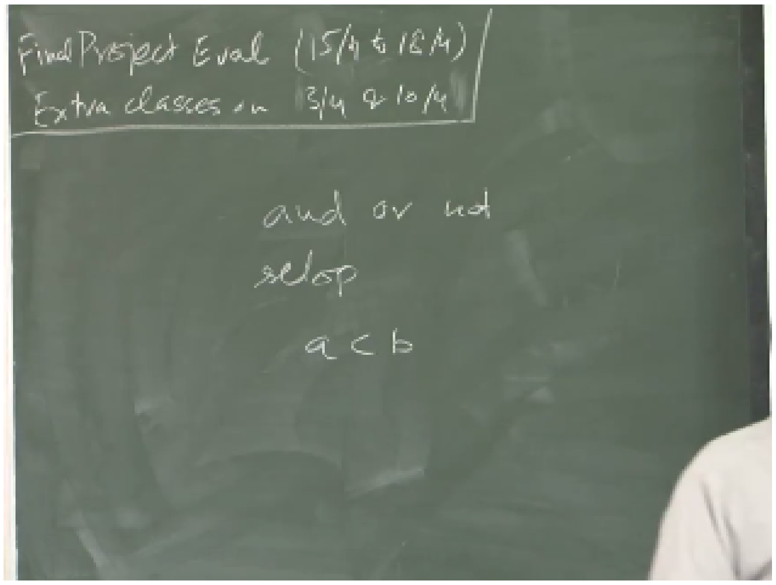


Final Project Eval (15/4 to 18/4)
Extra classes on 13/4 & 10/4

Good afternoon come back of allocation and just then talk our discussion code generation is a start today we know how the project starting 15 and we will do can we are popping can do coding this final exam and this two classes then we to have one are explanation okay.

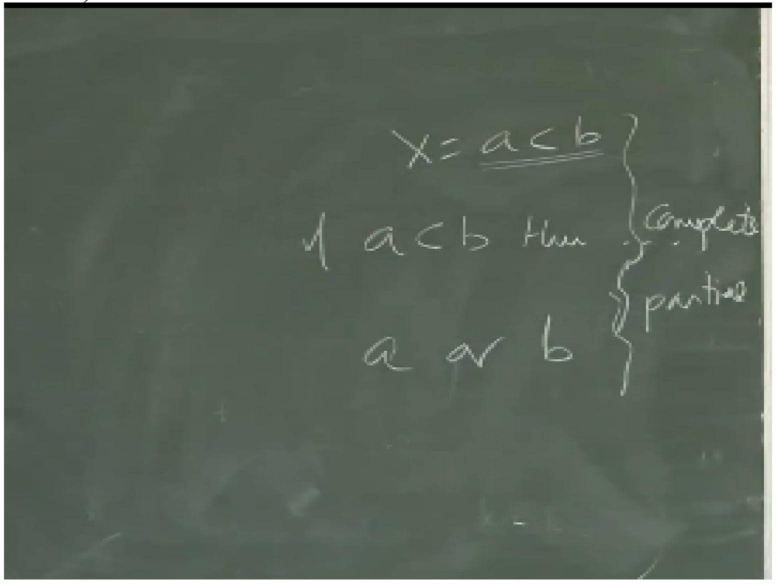
Any how the problem was coming for a stack so let us start looking at now Boolean expressions we are going to be how many we are going to be one that will involve some operators like and or and nor we have the relation operation okay.

(Refer Slide Time: 02:40)


Final Project Eval (15/4 to 18/4)
Extra classes on 3/4 & 10/4
and or not
relop
a < b

So typically that say a to b okay now when possible cogeneration of Boolean expression then this evaluation of co generation two ways of co generation one of the evaluate of Boolean expression and come to two general analysis and forms but now can evaluation normally what kind of used to do either I am going to a assign the variables and I have the equations like thing that is the point.

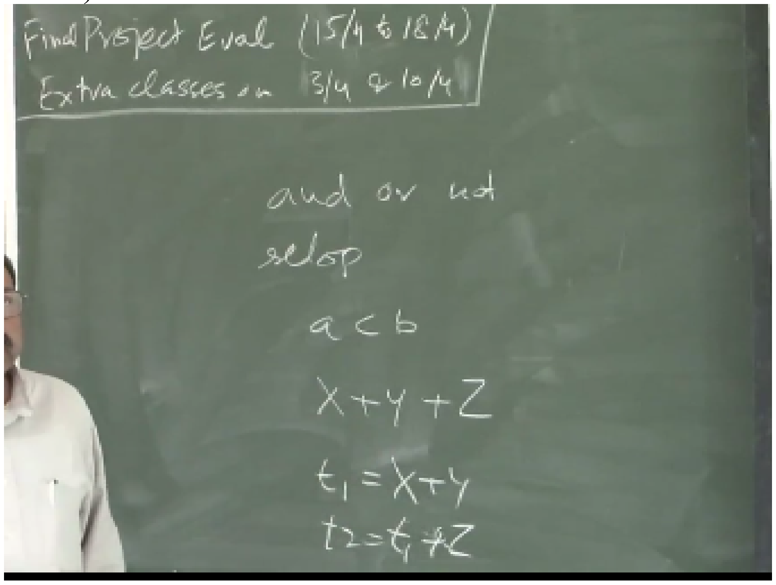
(Refer Slide Time: 03:13)



And X is the point a less then b and what kind problem to this cases and that time we will find out what is the value when I say that I want to evaluate the Boolean expressions but I do not have any vary kind of difference so I may have the situation in case a less than b change into the Boolean rule so this one of the another rule of Boolean expression so what we going to do are Boolean expressions how to be this kind of values.

And this kind of evaluation and second is important form is that we want to see that sometimes when we will either doing this type or this type if I can do a partial evaluation we have to see both evaluations okay now one way of doing this technique all my Boolean evaluation and Boolean operators and contain okay When I say that so let us go back to what we did for operators.

(Refer Slide Time: 05:00)


Final Project Eval (15/4 to 18/4)
Extra classes on 3/4 & 10/4
and or not
relop
a < b
X + y + Z
t1 = X + y
t2 = t1 + Z

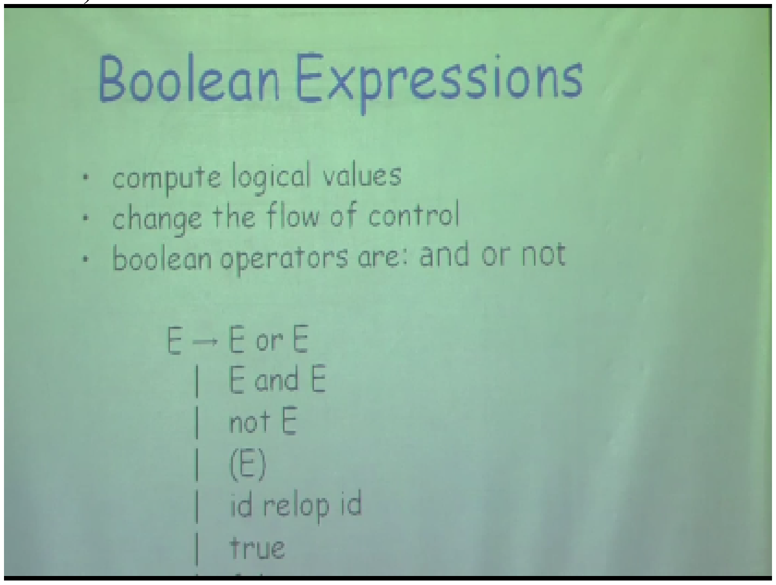
And I had X plus y plus z what are the kind of this okay t1 is t plus y and t2 is t1plusz okay and then t1 and t2 are the are the values f the expressions similarly now we can say that instead of these operator I have Boolean operators suppose I will say this is epsilon and only thing that will happen that t1 and t2 are the state values true or false okay for this discussion either this will take value true or false.

So this discussion the three I that take this 2 values true or false or we will say that discuses the value is 1 discuses the say that the value 0 or 1 now you can the language we can see we says any non linear value for this true value okay we the value of false value and but some languages put restrictions there we will say that either this keyword like true or false or there are only 0 or 1 and so on so really that not the fact.

Our discussion changes of that so this is really over all thing of all generation of Boolean expressions before I going to this okay of looking at how do we write grammar okay other in the general formats general questions some of this it not making start doing this expressions this concept is clear what we trying to achieve we now trying to two four generation for Boolean expressions that we involve these operators and linear operators.

And using for either kind of value either assigning a value or what changing into flow of control and we also the we have to handle both of the situation will be say that want to have complete evaluation of the Boolean expression of the form okay.

(Refer Slide Time: 06:56)



Boolean Expressions

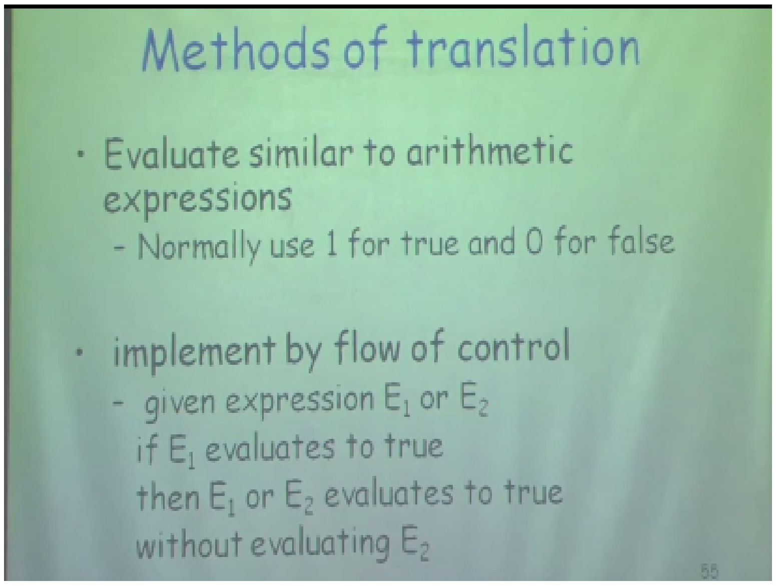
- compute logical values
- change the flow of control
- boolean operators are: and or not

```
E → E or E
  | E and E
  | not E
  | (E)
  | id relop id
  | true
```

So let us look at this we are trying to compute logical values and we have to change flow control and are the Boolean operators are and or and not and typically I have grammar rule which is have the forms says e goes for e or so this is the grammar for Boolean expressions to phase which says either or operator alternative operators this not okay that is an expression and for something say that or a unity of a is normal.

And then one I have Boolean expression or I may have something like saying that are you ID ever fight or I may have base versions being assigned to each which has true or false which are either you or so in this case I can just see that mistakes.

(Refer Slide Time: 07:41)

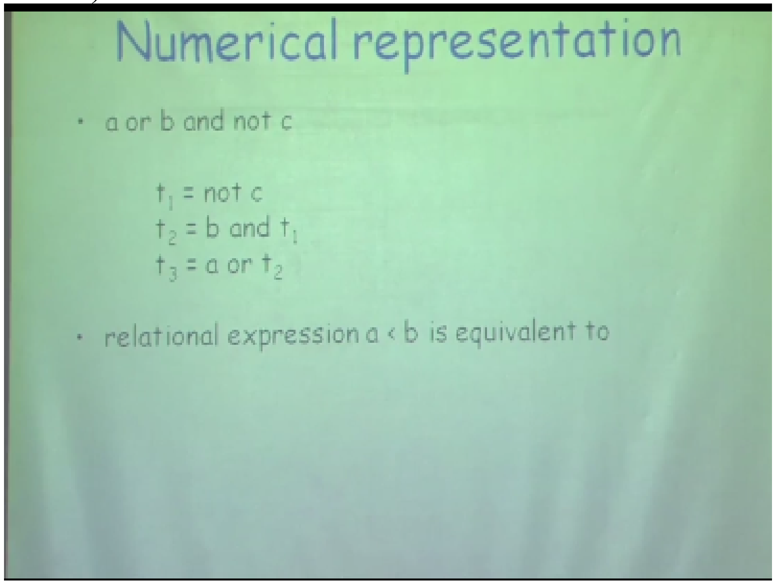

Methods of translation

- Evaluate similar to arithmetic expressions
  - Normally use 1 for true and 0 for false
- implement by flow of control
  - given expression E_1 or E_2
    if E_1 evaluates to true
    then E_1 or E_2 evaluates to true
    without evaluating E_2

55

So one of the matters of translation either I am going to do it similar to expression to give the c value use 0 or the implement so flow of control when I am saying that I will do only evaluate values not only this kind of expressions and have assignment or we have to change of flow control and to values okay.

(Refer Slide Time: 08:05)


Numerical representation

- a or b and not c

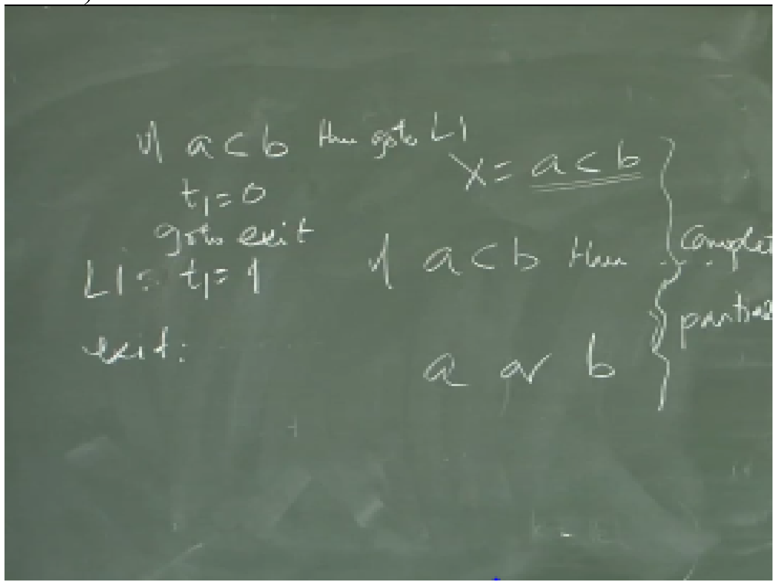
  t_1 = not c
  t_2 = b and t_1
  t_3 = a or t_2

- relational expression a < b is equivalent to

So let us look at the first numerical computation and start with an example and then go back so suppose I have an expression of this form which says they of the a or b and not c what are I am going to do that first I will say let me apply this unity operator and t 1 and then I say t2 is b and t1 and t3 is a or t so it finally we have this expression okay now I mean we know that grammar this is an associative so unity operator.

Is so similarly this is typical operator for this so similar if I want to say a less than B then what is the equivalent code for this so that was in general what about this of my grammar rules this was Boolean expressions then what we are code for this complete this one now I can do this and now have to check.

(Refer Slide Time: 09:50)



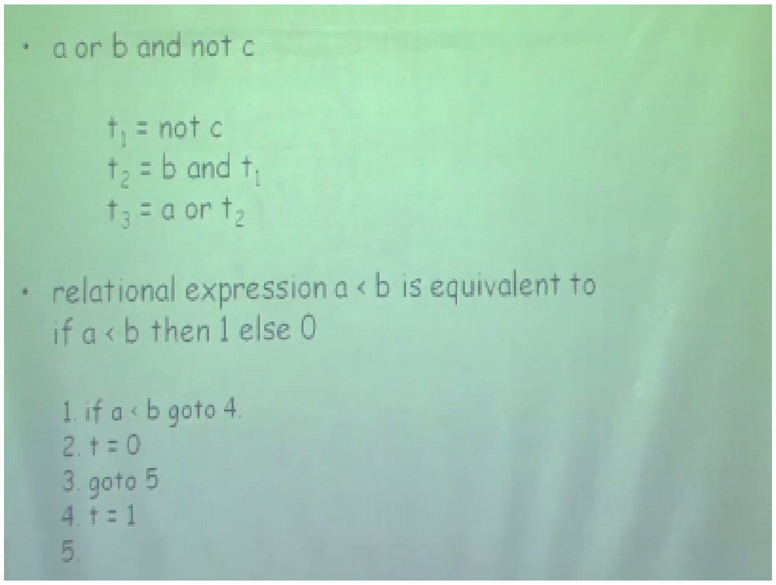
```
if a<b then goto L1
  t1=0
  goto exit
L1: t1=1
exit:
```

$X = \underline{\underline{a<b}}$
if a<b then
a or b
complet
partial

If we so all I need to do is and now have to check if a is less than B then what happens then what should be the value of this one right so then I can say just create this code I can have that statement then one can just say that l1 say that t1 assign 1 okay and this is false then I can just form 2 and into and then I say t 1 is assign 0 and then I need to do something after this assignment yeah I need to make sure.

That I jump over this statement so that this does not get it once again so if I just say now go to exit and my exit is here then have I achieved that what you are saying is this is equal to saying it is a less than B then value is 1 otherwise value is 0 and if I translate this is how to translate that if a is less than then go to 4 and 4 is really nothing like absolutely.

(Refer Slide Time: 11:13)



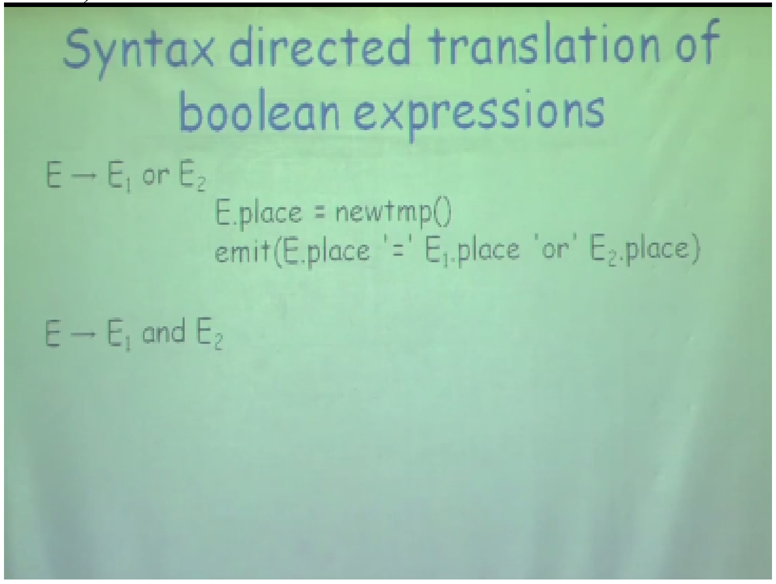
- a or b and not c

  t_1 = not c
  t_2 = b and t_1
  t_3 = a or t_2

- relational expression a < b is equivalent to if a < b then 1 else 0

  1. if a < b goto 4.
  2. t = 0
  3. goto 5
  4. t = 1
  5.

But saying that this is a label which is relevant to or which is at a relative distance for the statement so now you can see that that is not matter and what is the address of this it always wants to jump to whatever is the current address plus C right so I guess this for by saying that if a less than B then go to current address plus 3 then.
I say B is assigned 0 then I say go to current address plus 2 and then I say is assigned 1 okay so this is exactly what we are going to do now taking the grammar and writing the rule for generations now we already know how can we generate these template there more levels which are required as far as this is concerned but here we require labels but labels are going to say up that's an offset so I know how to generate templating against it equities for that.
(Refer Slide Time: 12:23)



Syntax directed translation of boolean expressions

$E \rightarrow E_1$ or E_2

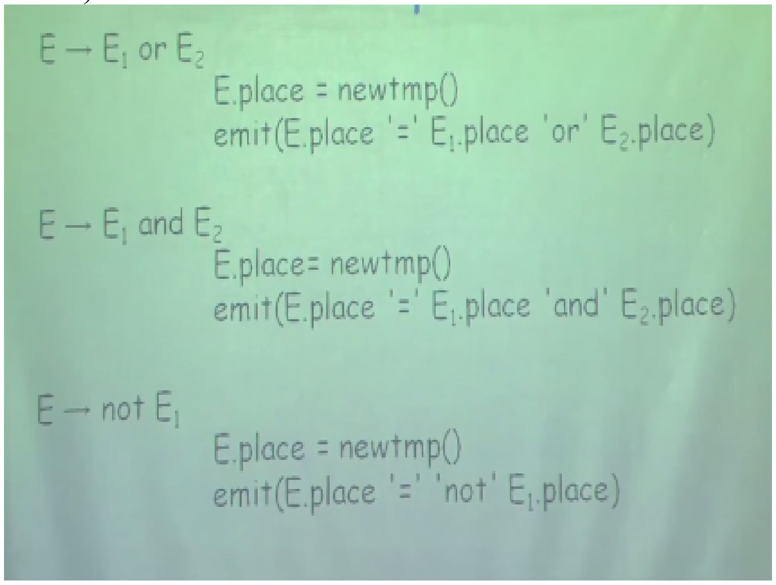
  E.place = newtmp()
  emit(E.place '=' E_1.place 'or' E_2.place)

$E \rightarrow E_1$ and E_2

So when I say that I have this Boolean expression e1 or e2 what would I do here okay again I will use the same synthesize that we would say that E one value is available in the place e 2 value is available in the place and e value which has to be created it has to be now created okay so this is first thing I will do that I say he places location and then I am going to emit the code sequence if you say that key place to the sign.

Even place off and E quad I am not writing explicitly but basically that is going to be even port followed by e2 poor followed by this particular instruction so if I can connecting this between be great so both the slides are used right either I can accumulate in a variable which is the synthesized exactly to power in keep on any P right okay what happens in this case what is the kind of four sequence are generate in this case you can here for this except with the same one except then this operator we have to case five and operator okay so there is a kind of code sequence again about dot e dot d e one what is the type of sequence I get for this.

(Refer Slide Time: 13:48)



$E \rightarrow E_1$ or E_2

E.place = newtmp()

emit(E.place '=' E_1.place 'or' E_2.place)

$E \rightarrow E_1$ and E_2

E.place= newtmp()

emit(E.place '=' E_1.place 'and' E_2.place)

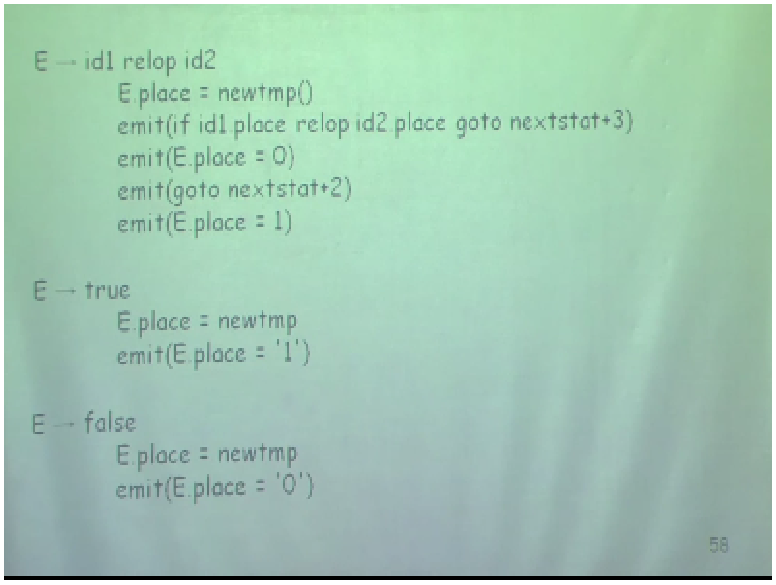
$E \rightarrow$ not E_1

E.place = newtmp()

emit(E.place '=' 'not' E_1.place)

So again I need a new temp for e so this will be new and popular little bit here I see phase you can see the computing now is instead of value of an expression after computing 0 and so on and I am still doing partial evaluation this is giving to write okay so let us move on and if I have graphical expressions I do not have to verifying okay so previous case I want to do this expressions and anything and now.

(Refer Slide Time: 14:35)



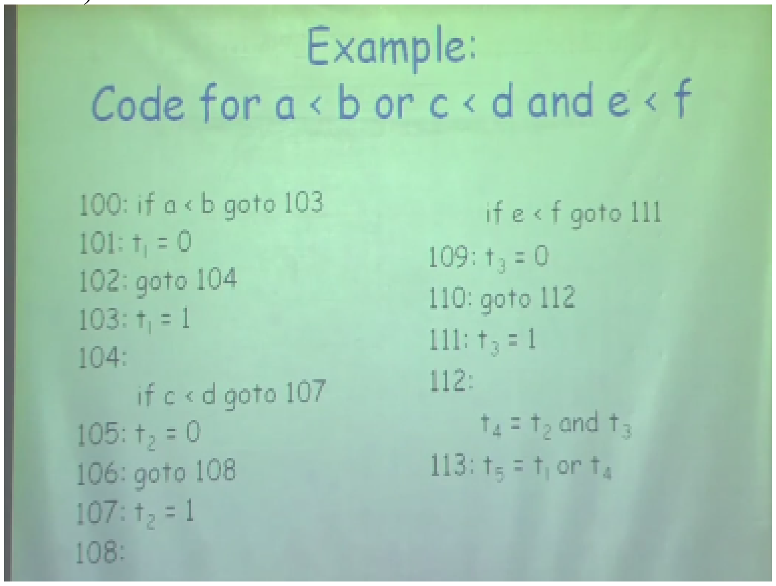
```
E → id1 relop id2
        E.place = newtmp()
        emit(if id1.place relop id2.place goto nextstat+3)
        emit(E.place = 0)
        emit(goto nextstat+2)
        emit(E.place = 1)

E → true
        E.place = newtmp
        emit(E.place = '1')

E → false
        E.place = newtmp
        emit(E.place = '0')
```

58

If I have I do want them of ID1 relpo id2 then what is the course equal sign is it exactly what we just discussed here we say that this is now going to be in do them and then I just say that I'm first going to emit if I need one place and offer you to place.

Then I am going to go to whatever is my location plus C and then I am going to say that if this is assign 0 and then I say go to whatever is the current location plus 2 and then he places one and then that is it and if this is true then one of the core sequence I generate for this so I just need to say he places a new tank which is going to be assigned one exhibition which either has all these Boolean operators of relational operation I can find out the value of that Boolean expressions going to be o right okay.

(Refer Slide Time: 15:50)



Example:
Code for a < b or c < d and e < f

```
100: if a < b goto 103          if e < f goto 111
101: t1 = 0                     109: t3 = 0
102: goto 104                   110: goto 112
103: t1 = 1                     111: t3 = 1
104:                            112:
     if c < d goto 107               t4 = t2 and t3
105: t2 = 0                     113: t5 = t1 or t4
106: goto 108
107: t2 = 1
108:
```

So let us take this example and let us try to generate code for this particular Boolean expression to find out a less than b or c less than b and e less than f exactly same thing that the weight and this is that you have an idea top ID here you have an idea about by details which gets the expressions and then this is e and E and then this also gets to use E and then this is e off this is how the parse t for this particular Boolean expressions.

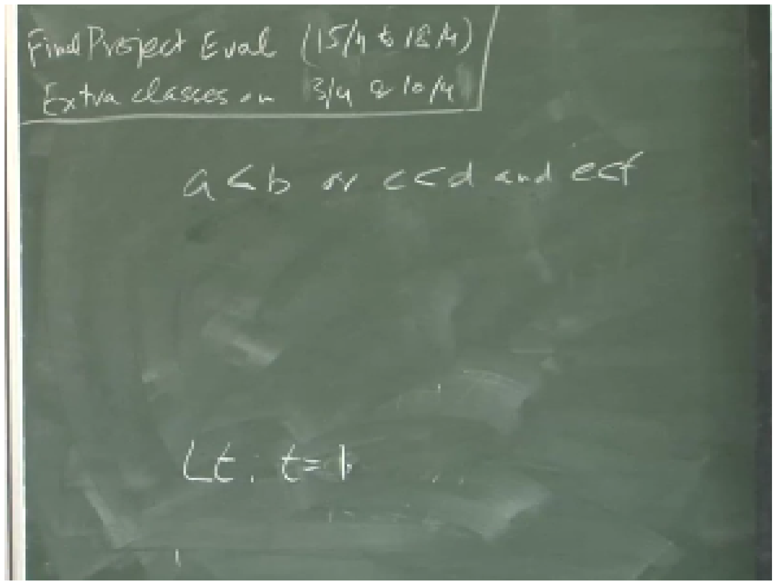
Going to case to so first time you say that let us generate code form a less than B and then slow the value of that available for T1 so I can just five code like this and say that if a less than B then go 103 so on assuming that my code generation starts from location okay and therefore with an offset of T I can jump there is it even is one or if it falls through then P 1 is 0 and then I jump it so at end of these four instructions P 1 is going to contain value 0 or 1 depending upon the rule true or false okay.

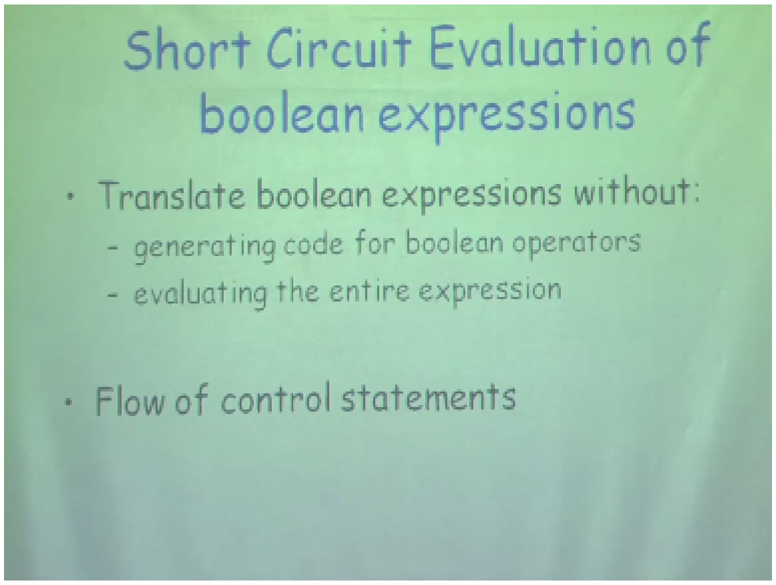
Similarly I can now write code sequence corresponding to C less than D and that is going to give me so assuming that this particular instruction is star from address 102 I will have an offset of 3 and so on and to that value of t2 of 1 okay.

Similarly I think like port for E less than F so you can see that these are exactly the same he must the only thing that is changing up that addresses I start with 100 here I'm starting with hundred four and values are going to be stored in T 1 T 2 T 3 now once I know that this is T 1 T 2 and T 3 I do now.

By visited says that first time we have added this particular Boolean expression and then this so that is what exactly do I say now people is t1, t2, t3, t4 and t5 that the form possible can okay so if clear this any confusion on this any complex equation anything and now going to get little more complex so I will using only single attributes assign was exactly same okay now I am coming to short evaluation okay so what we want to do is shot circuit evaluation the Boolean expression and so I am code to the before I saying that sure okay.

(Refer Slide Time: 19:30)


Final Project Eval (15/4 to 18/4)
Extra classes on 3/4 & 10/4
a<b or c<d and e<f
Lt: t=1
A less than B or c then B and who now is I do not want to now say that this will have value of T 1 this will have value of C 2 and this will give the value 3 and but I would say suppose I evaluate this and this evaluates to true then I know irrespective of what is C less than D and T less than an expression now if I know that for sure then not only I will jump over this part of the code.
I will straight away go to a location saying that if this is true then I should be able to reach some label which will be saying that D is sine Y and if this is false then what do I do then I just jump to this any evaluate right that is only thing I will do okay so this exactly what we will do In short circuit values are so now for argument say and then just to make a simple expressions that behind to that.
So and so on I am not the dealing with syntisization and behind that and behind that where my jump after finding their little thing an expression is so that is that you will have and we are now going to translate.
(Refer Slide Time: 20:49)

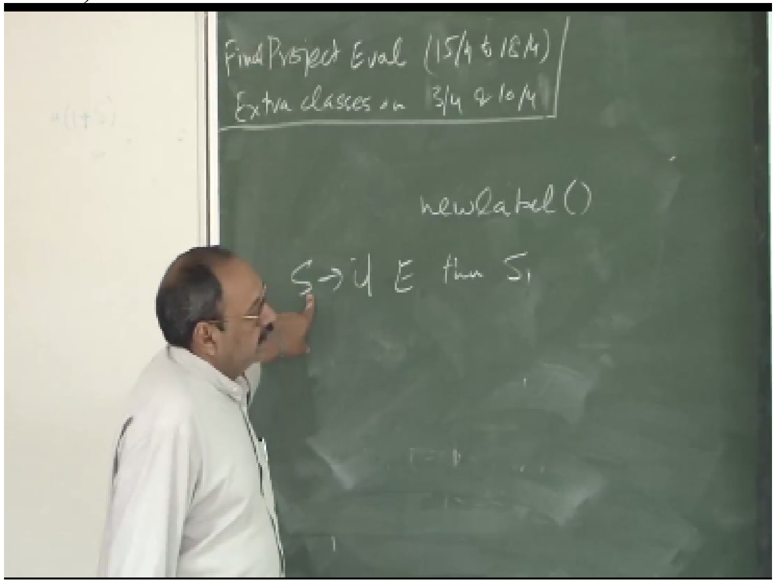


Short Circuit Evaluation of boolean expressions

- Translate boolean expressions without:
  - generating code for boolean operators
  - evaluating the entire expression
- Flow of control statements

Boolean expressions without generating code for Boolean expressions and without evaluate the entire expressions and some languages allowing to like some languages want to do so first we have look at flow of control statement before getting into the Boolean expression jumping to certain locations okay so remember that I also introduced a function called new level which was then it enables for me like.

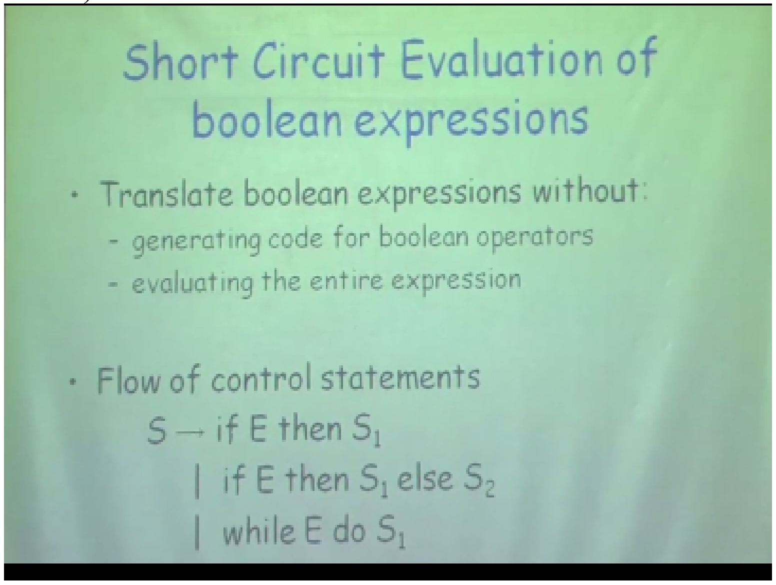
(Refer Slide Time: 21:23)



So I have this function and then if I have this then s and this end of statement then what did I go then I set that angle we generate labeled with this evaluates to true where do I go then I was actually directing this I was falling through and if this was value to false then I was generating a label which was the beginning of the next statement okay now I remember there was compensation boxing saying is it possible that in this particular scheme right.

So we had a situation where suppose this statement was followed by a y statement where I was generating a little for beginning of y and I was generating a label for next of x okay now it's slowly we will see that I am trying to address that situation also let us assume that we have an attitude which says that when I start parsing this I know where the control will go as next is available okay.

(Refer Slide Time: 22:28)



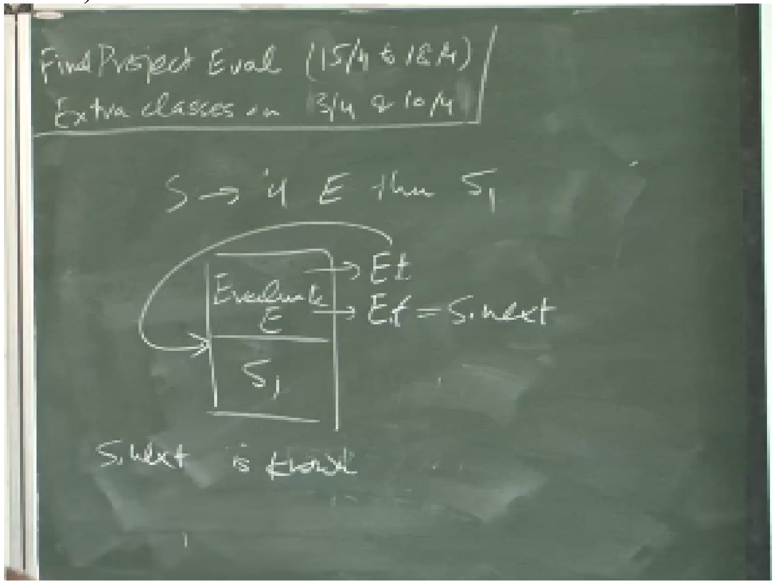
Short Circuit Evaluation of boolean expressions

- Translate boolean expressions without:
  - generating code for boolean operators
  - evaluating the entire expression
- Flow of control statements

$$S \rightarrow \text{if } E \text{ then } S_1 \;|\; \text{if } E \text{ then } S_1 \text{ else } S_2 \;|\; \text{while } E \text{ do } S_1$$

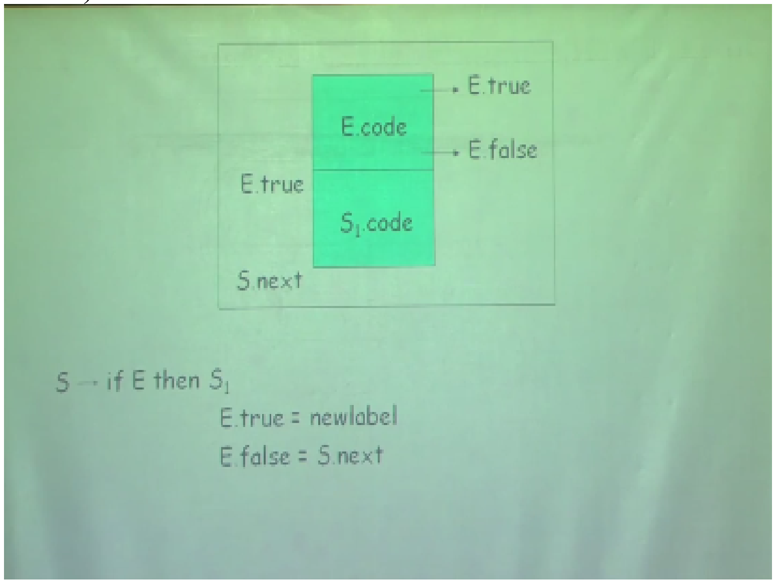
Now once I say that X next to e is available to meet that means inherited then if I look at this so let us look at the scheme I try to understand what happened okay.

(Refer Slide Time: 22:40)



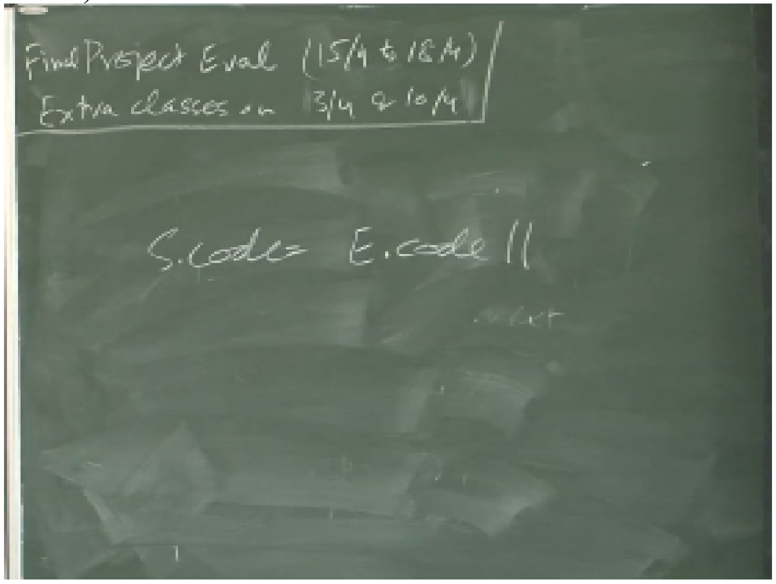
So now suppose I say if E than S1 okay what will happen here now if I evaluate this and I go be a statement this evaluates to and now if I real like this okay evaluate 2 where to control it so

where will control so I have two exits from this I have e true and I have this is right I don't have this variables and control the evaluation okay.
So this label has to be generated but where will the control go if it evaluates to false here right so now I can state let me say that this is okay so the very I stopped writing schema for this is so let us look at this scheme.
(Refer Slide Time: 24:09)



$S \rightarrow$ if E then S_1
E.true = newlabel
E.false = S.next

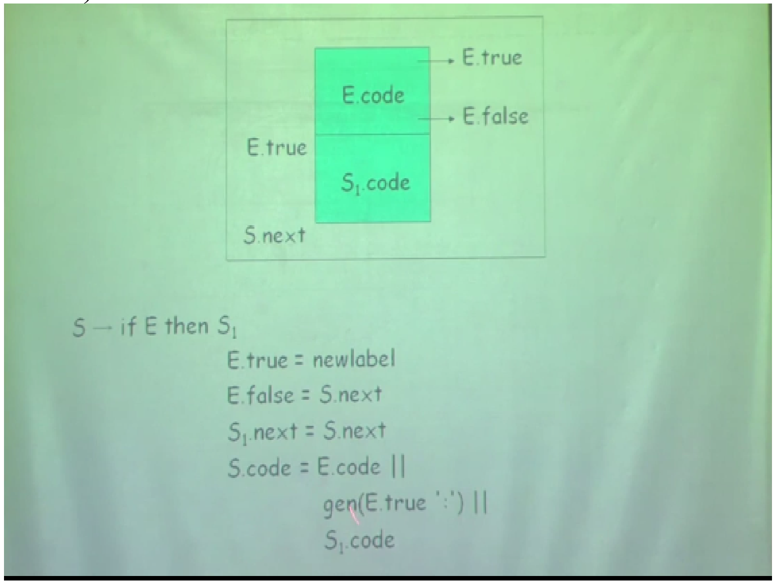
Now that we have port for e which is giving me two exits either on to or false true that to comes here and the false control comes here okay so how is my old loop okay now let us say that e to is a new level because this labor I did not have signing new level function but e pauses s next which is already initializing and which is ending that will get it and now how am I four of s you forget I like my s for test.
(Refer Slide Time: 25:09)



Final Project Eval (15/4 & 16/4)
Extra classes on 3/4 & 10/4

S.code = E.code ||
next

So if you just understand this things will be following this so obviously posting is going to be good right I have to E equal yes and then what do I do you only want to change flow of the control that is not assign to the value that was I thing of slightly different and check and handle for that and that is why I have looking that ,So after E code what word so let me just show you the code and then sort of arguing whether this is correct and the code is as simple as this.
(Refer Slide Time: 26:15)



$S \rightarrow$ if E then S_1

E.true = newlabel

E.false = S.next

S_1.next = S.next

S.code = E.code ||

gen(E.true ':') ||

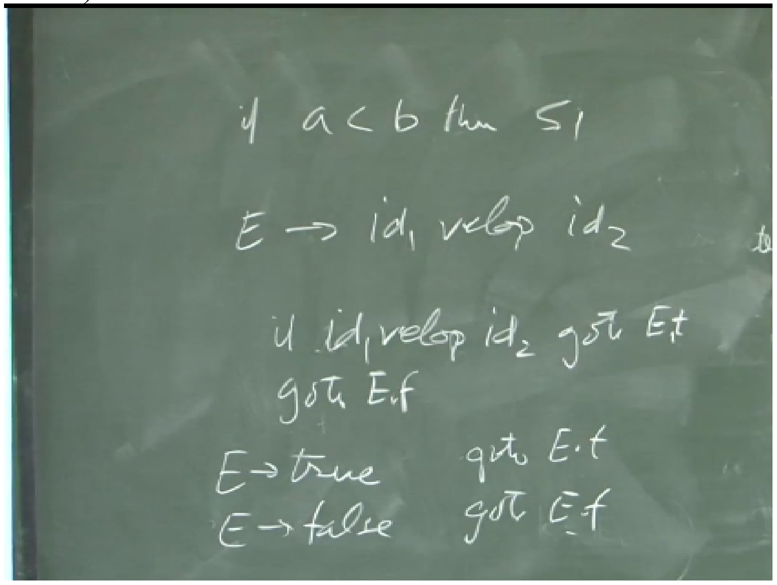
S_1.code

Now I say this is e code equal followed by this label followed by s1 whole that is the code does it make sense what is happening here .So how is change of control flow taking place and one of those that I also have initialized the variable which does not appear which is s 1 next so this initialization was simple because all I am saying is this exit jump in here and everything from there but this initialization and this piece of code seems to be slightly tricky .

So let us first come to this initialization anyone any guesses see s1 may again have a Boolean expression I do not know s1 is just a statement which will be unlevel when I start parsing that right so s1 inside suppose I have Boolean expression inside s1 then I need to know what is the location where I should you jump after finishing s1 so since I am now using inherited attributes I am saying I know s next so obviously if I know s next then s1 next also has to be initialized okay so that is what is being initialized here so otherwise this question would have come out how would I do next at some point of time you will find this s next a larger state.

If i initialized it there then I must initialize it here and now what happens in this code when I say e code okay what is e code here the jump statement will be part of the e code so e code is now not saying that make a or b and you store value to some variable T and then find out whether this

value is true or false e code will be just saying that if this time is to jump to this location this value is positive and so on .
So all the jump statements are actually part of the e code and then I just need to generate this label and I need to generate cod e for s1 so I need to generate at this level so I need to generate s next when you could have a line which will say gen x next no because again this label must have been generated where this whole s was being appended in some statement this s must have occurred on the right hand side like here I just get this as x1 as the initialization.
(Refer Slide Time: 30:05)



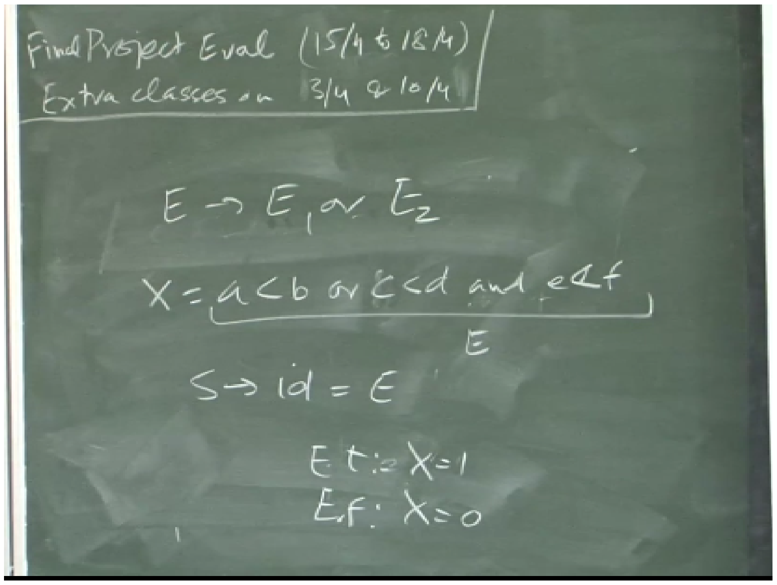
if $a < b$ then S_1

$E \rightarrow id_1$ relop id_2

if id_1 relop id_2 goto E.t
goto E.f

$E \rightarrow$ true   goto E.t
$E \rightarrow$ false   goto E.f

So let us suppose I write something like this if so this is my grammar rule which is parsing my Boolean expression now earlier I was generating four instructions for this saying that you first evaluate this and if this value is true and so on now I need to say is if id okay and if e is getting reduced by this rule then what is the rule I need to write e is v true go to e.t and false how can you create a Boolean expression .
(Refer Slide Time: 31:58)



Final Project Eval (15/4 to 18/4)
Extra classes on 3/4 & 10/4

$E \rightarrow E_1 \text{ or } E_2$

$X = \underbrace{a<b \text{ or } c<d \text{ and } e<f}_{E}$

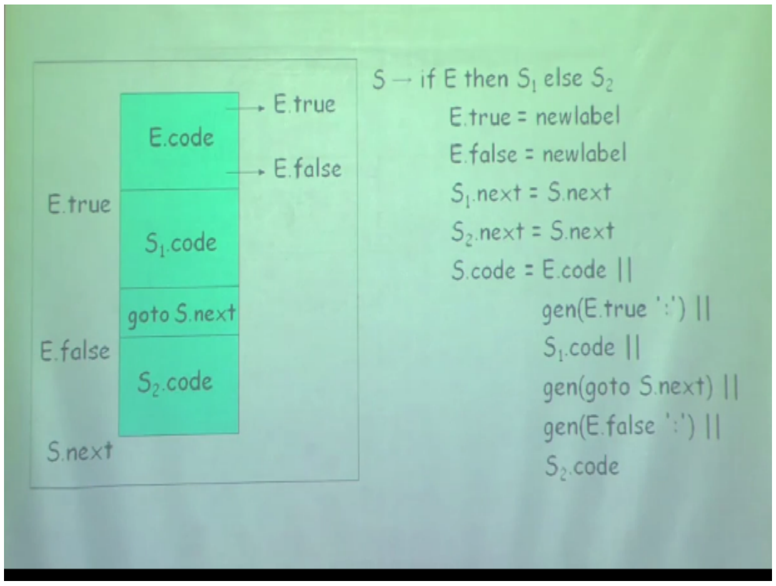
$S \rightarrow id = E$

$E.t: X=1$
$E.f: X=0$

Why do we waste values or you need relational operation and once you have done this and what will happen if I say if then what is the kind of code I write for this so what is the schema for doing this e 1 or e 2 so first I will evaluate e1 and if this is true then what do I do then I will jump to e 2 right and if this evaluates to false then what do I do then I need to get it a label which will take me here and then I need to evaluate so I do not have any variable being assigned 0 or 1 I am only jumping around from one level to another till finally I reach the target .

So now in light of this does it make sense say that what is the kind of code will now look if I am doing short circuit evaluation and you can see that short circuit evaluation always is going to be much shorter as compared so let me go to the next part and then if it is still not clear we will come back to the base case. So the next one is now if e goes to s this is really what in am looking at the statement .

(Refer Slide Time: 33:10)



$S \rightarrow$ if E then S_1 else S_2

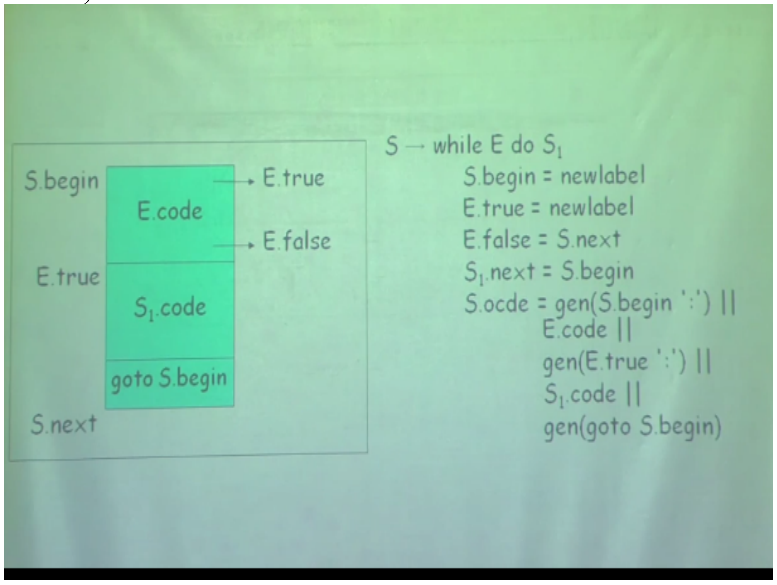
```
E.true = newlabel
E.false = newlabel
S1.next = S.next
S2.next = S.next
S.code = E.code ||
        gen(E.true ':') ||
        S1.code ||
        gen(goto S.next) ||
        gen(E.false ':') ||
        S2.code
```

So this is how the schema looks and I have the e code which will have two exits either true or false and if this is true then s 1 code will get executed but after s 1 I need to jump to s next and if this evaluates to false then I will jump to s2 okay and now what are the things I need to do I need to first generate code for e and how do i initialize these labels so what is e to be assigned how will I initialize each group e 2 is jumping here right which is in the middle of the schema.
So that has to be from a new label it not available to me I just have to generate this what about e false also has to be then initialized to the new label okay what about s1 next they will control go from s 1 after it finishes control must go here and from s2 also control must go there okay so s 1 next and s 2 next both are being initialized to s next so this is the initialization of all the labels and how will my code look now.
So after I have initialized so I have this label being initialized this is inherited and s 1 and s next connects up then initialized to whatever I have inherited and then how will my s code look s code will be first e code now I do not have to worry about jumps because that is going to be parser and get it in the code itself and then I say I need to generate this label then I need to give code for s1 then I need to generate this statement and then I need to generate this label this level .
We have these an actual assignment so what we are saying is something like this okay so look at it this way that this is still e and this statement is going from what I need to say is that do exactly the same thing what we are doing here and then I have two levels which will say e true and E false and e to and say x assign 1 and say x assign 0 who is responsible for that assignment so suppose instead of saying a less than b and I have these as function calls .
So suppose I have 2 functions f1 and f2 which return Boolean values we change values and suppose I say that I am going to evaluate this first okay and this has a side effect which impacts

this value so if language permits this facility so look at it this way its language perfect the facility then make sure one is not misusing it is job of the programmer but if suppose the programmer decides to write something like this yes.

Then compiler has no control over it worst which can happen is that many times it is not clear we start defining that when I say if a less than b is it the b I fetch first or a I fetch first so if I say that I am going to fetch this first evaluate this first some compilers will do that in some compilers will do that because languages do not define a is less than b which function should be evaluated first so two different compilers can give two different values okay.So let us go to the next schema which is white now you have to write the right-hand side code for this.

(Refer Slide Time: 39:21)



$S \rightarrow$ while E do S_1

S.begin = newlabel

E.true = newlabel

E.false = S.next

S_1.next = S.begin

S.ocde = gen(S.begin ':') ||

E.code ||

gen(E.true ':') ||

S_1.code ||

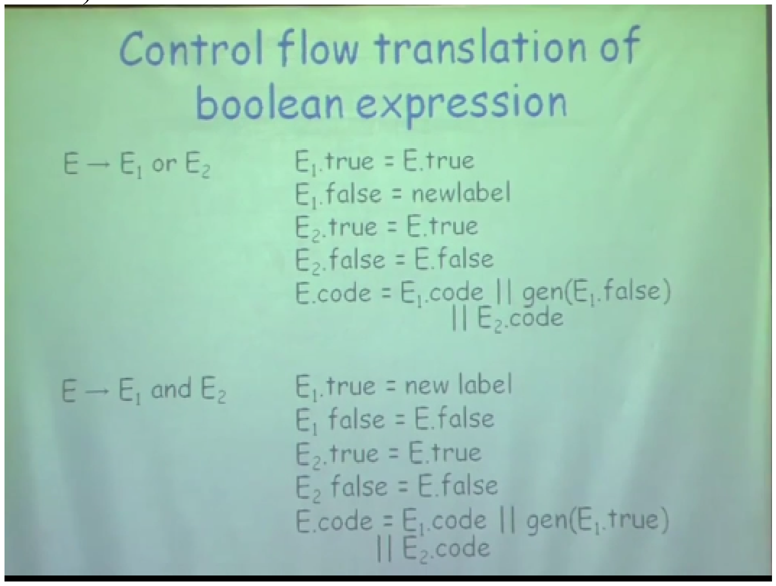
gen(goto S.begin)

Let us take one minute to write it down write the complete code so you can see that I need to initialize both equally false okay and what else I need to initialize s begin right so I speaking s to be now initialize by a function called new label and this is what will happen code is going to be e code followed by this one remember that since we are dealing with inherited attribute always whenever you have a statement.

We do not know what the statement contains we must initialize this video but it is possible that I will never lose s 1 next I did not have a jump statement but we need to have a variable so we do not know and s o code is now going to be that so I need to first generate this label then I need to generate this code so I do not have to worry about generating jump true or false starting here but I need to do this jump so scheme actually tells you systematically on what that will use you are capturing the semantics of the statement .

So this is what we are saying that has never existed that either hopefully then I can get this labels and either for s1 and then I generate and so clear now. so let us look at now going back to the question which was asked earlier.

(Refer Slide Time: 42:11)



Control flow translation of boolean expression

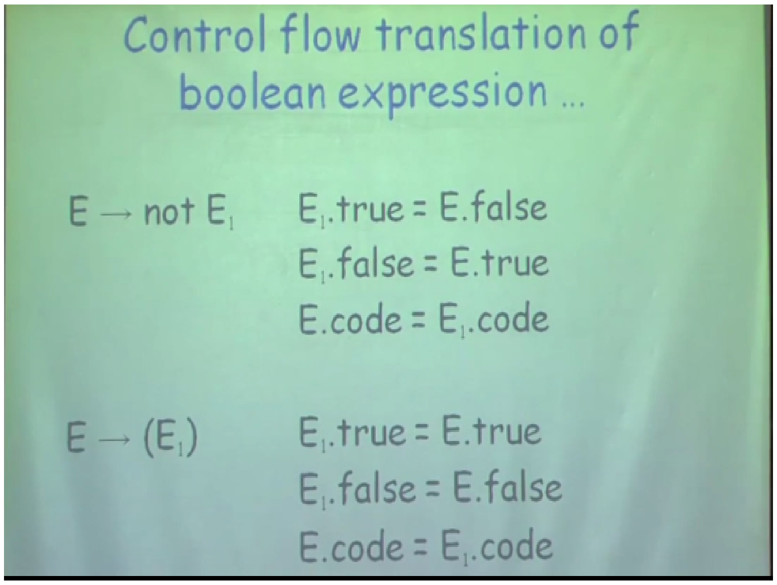
$E \rightarrow E_1$ or E_2	E_1.true = E.true E_1.false = newlabel E_2.true = E.true E_2.false = E.false E.code = E_1.code || gen(E_1.false) || E_2.code
$E \rightarrow E_1$ and E_2	E_1.true = new label E_1 false = E.false E_2.true = E.true E_2 false = E.false E.code = E_1.code || gen(E_1.true) || E_2.code

The support for e now if I say that I want to generate this code a now I know that if I am inheriting this I would get true and false it so remember that true and false before I came here so if I go back there true and false have already been initialized either by an inherited attribute or by the function so when I start now generating this code I know that what is e true and e false these are just some target locations so what will be e1 true e1 true is going to be e group control goes and what will be e1 false .

Where will e 1 false go e 2 is not initialized yet it has to go to the beginning of e that has to be a new label right so e1 false will jump to this location which is being a copy and what is e 2 true is obviously true and what is e 2 false it is e false very good and what is the code for e so just try what e code will be so e1 code followed by I need to generate this label which is e 1 false followed by code that is how my total code is going to look .

How can I do the same thing for and operation here instead of all what will happen in case of and if I do write evaluation then what will happen if I have and here then we will e2 jump if I say e1 is evaluating true where should control go both beginning of e so that will be a new label right and what will be e1 false and e1 false is going to be e1 false and if e2 evaluates to true then the whole thing is true because control would have come only this was true and my code is same except that this level .

(Refer Slide Time: 45:12)



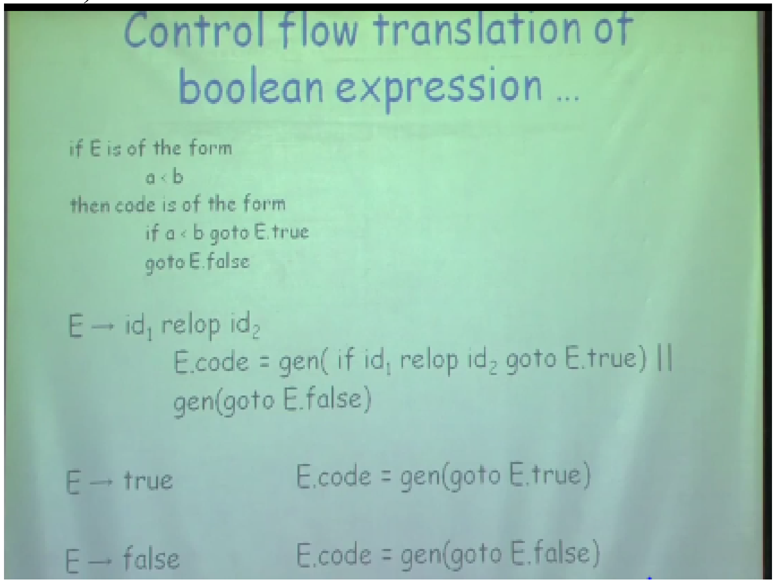
Control flow translation of boolean expression …

$E \rightarrow \text{not } E_1$ $E_1.\text{true} = E.\text{false}$
$E_1.\text{false} = E.\text{true}$
$E.\text{code} = E_1.\text{code}$

$E \rightarrow (E_1)$ $E_1.\text{true} = E.\text{true}$
$E_1.\text{false} = E.\text{false}$
$E.\text{code} = E_1.\text{code}$

Okay what about this now in this case I do not have to do anything I just need to switch labels right so if I know labels of e1 okay then I say that e1 true is where e false is jumping and e 1 false so I just need to copy the labels and e code is nothing but e1 code and not anything else and then the expressions and then say two labels are copy true and false labels and four remains against okay.

(Refer Slide Time: 45:53)



Control flow translation of boolean expression …

if E is of the form
  a < b
then code is of the form
  if a < b goto E.true
  goto E.false

$E \rightarrow id_1 \text{ relop } id_2$
  $E.\text{code} = \text{gen}(\text{if } id_1 \text{ relop } id_2 \text{ goto } E.\text{true}) \mid\mid \text{gen}(\text{goto } E.\text{false})$

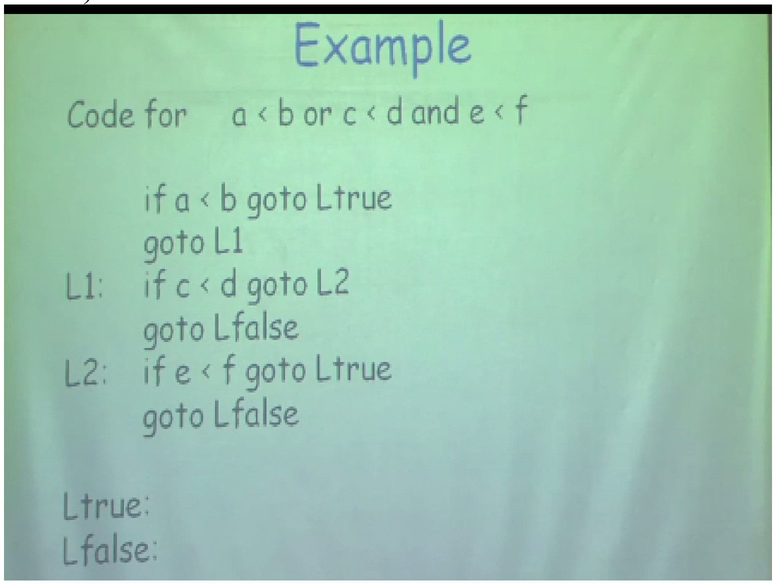
$E \rightarrow \text{true}$ $E.\text{code} = \text{gen}(\text{goto } E.\text{true})$

$E \rightarrow \text{false}$ $E.\text{code} = \text{gen}(\text{goto } E.\text{false})$

And what about this we are discuses these code form a < b then what is the code form the code Form is going to be that is a less than b goes to E true otherwise go to form to E false okay this is the four sequence expressions so when I now say id1 repo id2 what are the code sequence is generate I say if id1 there are elope id2 that go to 2 are by go E false and this is true and the evaluation as well as Boolean expression any more for the cases final far to design and changing slope rather case I will be find that okay.

If you have this kind of is specially design then finally jump to these any two false okay so there is example okay.

(Refer Slide Time: 47:10)



Example

Code for a < b or c < d and e < f

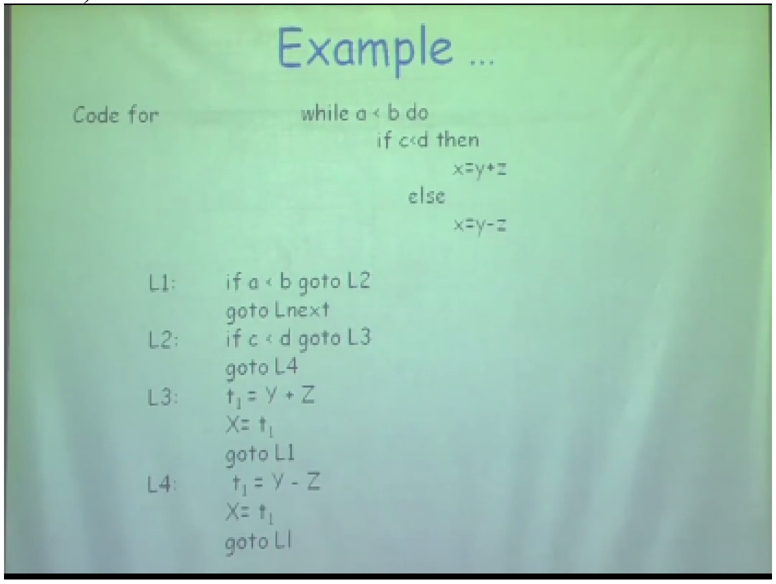
```
        if a < b goto Ltrue
        goto L1
L1:     if c < d goto L2
        goto Lfalse
L2:     if e < f goto Ltrue
        goto Lfalse

Ltrue:
Lfalse:
```

If I was write short circuit evaluation code was it so have only instructions would I have for each such phase and indicate code instructions as well should I have each surface I generate to that instructions as well then I had one for this and one for this total of code instructions I will say that A < b and jump to L true is that the whole thing is 2 okay and if otherwise I will just say go to L1 but what we can L1 here L1 will be this point of time I am not will be the point of time I will not able to organized or I am saying that for A <b.

This is the to sequences identify that is not possible that I have detent to jump here and we will see that the optimization where to try to do determine that okay this point of time we also form that content so if now say that c less than b what is I generate that is c listen b then go to l2 because is the saying to begging of this and this is false that I am not sure that also working I also jumping okay and what will happen my L2 here L2 is true is labeled of e and f and I am saying is and jumping to 2 and now I know that working is true if it is false get I know and this is end false okay.

And now you can see that L2, L4 are two labels depending upon the weather I have a assignment here okay I have an assignment here by say next which is assign this fallowing to do this I will say X is being initialized 1 and X be initialized okay otherwise I will execute the two statement here if I say this is the Boolean condition if the condition is that S1 then I have S1 here otherwise s2 here everyone is confident about Boolean expressions yeah so now you can see that we started

with controls book we did not talk about we an expression at that point of life I treated white Boolean expressions as black boxes but a cube or something.
But I was able to begin and so I know how we did it for our I know how we can be poor for it is I know how to get it perform Boolean expressions all the control flow statements and PCOS business you can see that if you do not have procedures in your language yeah that you can actually generate convert law code which will work we address both of course I am trying to say tip with your body.
Because you are in our division at least we should be able to generate code then you do not have functions to procedures we see how to code it functions and procedures but at least if you have a single function then you are able to type checking and code functions step create this performance and then made this for on the spin port or whatever is the target machine I'm not even worried about optimization here.
So even if they are redundant jumps like saying go to L 1 and L 1 is just the next statement that I can just go on optimizer okay so let us move forward I have on more example here.
(Refer Slide Time: 50:55)



Example ...

Code for

```
while a < b do
    if c<d then
        x=y+z
    else
        x=y-z
```

```
L1:   if a < b goto L2
      goto Lnext
L2:   if c < d goto L3
      goto L4
L3:   t1 = Y + Z
      X= t1
      goto L1
L4:   t1 = Y - Z
      X= t1
      goto L1
```

Just please take this point and then we break there so if I have a statement like this which says less than B then I have this if step statement inside this and you see less than B they are doing this assignment otherwise I am doing this assignment so you can see by partial evaluation of Boolean expression if I say this is true then I am jumping go to 2 to otherwise to exit but in L 2 I have this Boolean expression.
Where I say that either X is going to get this value P 1 or Y is X is this value and in both cases you can see that after I finish I am jumping to the beginning of the otherwise get just know the

past before this and what will doing to do and more class and teach in class okay we will see the next class.

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