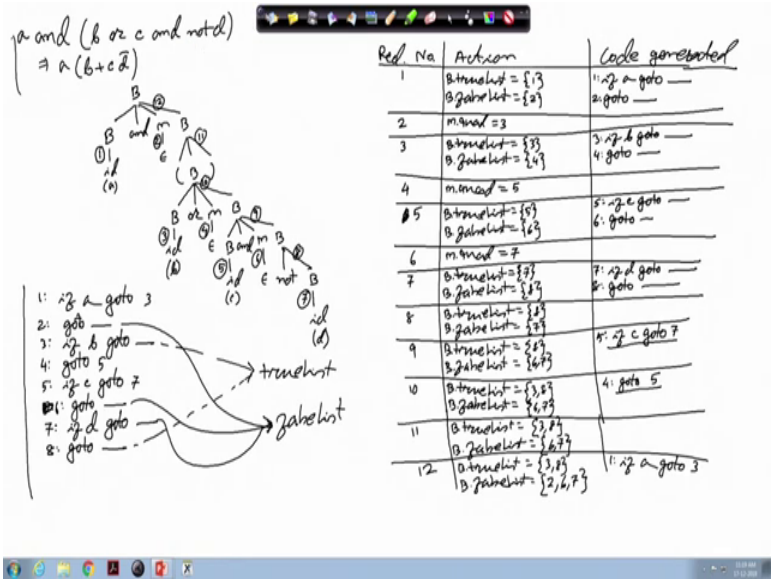
Compiler Design
Prof. Santanu Chattopadhyay
Department of E & EC Engineering
Indian Institute of Technology, Kharagpur

Lecture - 60
Intermediate Code Generation (Contd.)

Today we shall look into conversion of Boolean expression into 3 address code and apart from that we will take some more exercises for practice.

(Refer Slide Time: 00:27)



So, the example that we take for Boolean expression is like this. So, a and b or c and not of d that is basically a b or c d bar. So, corresponding to this Boolean expression, how can we generate the code that we will see. So, the grammar if we first thing is to draw the parts tree. So, the since we have got and at the top.

So, this will be b and m B then this B will give me i d which is a then this m will give me epsilon and then this B should give me within bracket b. Now the next this B will give me the or. So, B or m B. Now this B will give i d which is B this m will give epsilon now this B will give me the and function. So, it is b and m B this B will give i d, which is c this m will give epsilon this B will give not of B. This B will give not of B and this B will give me i d which is d.

So, this is the parts tree. So, if I number the reduction. So, this reduction will be done first then this one. Then it will do this one. Then this one, then this, then this. Then it will do this reduction then it will do this one. Then it will do this reduction and reduction then it will do the or reduction. So, this is number 10 and this is number 11.

Finally, this is number 12. Now you have to consult the semantic action for this Boolean expression and accordingly we have to generate the corresponding code. So, if I make some sort a table like reduction number in the corresponding action, and the code that is generated, code generated. Do it like this then at reduction number 1. So, if we look into the Boolean expression rule grammar then it says B first one is B 1 and m b 2. So, sorry reduction number, one is a reduction number, one is B producing i d.

So, for b producing i d the corresponding rule is this one. So, here it is not written, but in some other slide it will be there , but for b producing i d. So, it generates a code which is if a if i d goto this goto will be filled up at number 2 it will generate goto which will be filled up. And here it will produce this true list and false list. So, B dot true list is equal to 1. And B dot false list equal to 2 ok.

So, this will be done at reduction number 1. At reduction number 2, at reduction number 2 that is m producing epsilon the only action is m dot quad m dot quad is equal to next quad which is 3. That is done at reduction number 2. At reduction number 3 again we have got this b producing i d. So, 2 lines of code will be generated if we b goto to be filled up later and this goto to be filled up later. And action is that b dot true list is equal to 3 and b dot false list is equal to 4.

Now, reduction number 4 is m producing epsilon. So, it will do m dot quad equal to next quad that is equal to 5. Now reduction number 6, reduction number say sorry reduction number 5. So, reduction number 5 is again m producing i d. So, it will generate 2 lines of code if c goto unknown, this part will be filled up later and this is the else part. So, this is an accordingly it will generate true list and false list B dot true list equal to 5. And b dot false list is equal to 6. So, this is at reduction number 5. At reduction number 6 is m producing epsilon.

So, m dot quad m dot quad equal to next quad that is 7. So, this is done at reduction number 7 again B producing i d. So, it will generate lines of code if d goto and goto. Now at reduction number 7, this true list and false list will be generated. B dot true list

equal to 7 and B dot false list equal to 8. Now reduction number 8 naught of b. So, here the false list and true list they will get interchanged. So, B dot true list will be equal to 8 B dot true list equal to 8 and B dot false list equal to 7. At reduction 9, so, b and m b 2.

So, it will first of all. So, the and rule if we see B. And so, this is the B and. So, it first says back patch b 1 true list with m dot quad. So, that back patching will be done if m dot b dot true list. So, B dot true list is 5 f b dot true list is a this 5. So, 5 true list. So, that is a truly you have got 5 here. So, this location will get corrected now. So, that will be filled up with this m dot quad. So, m dot quad for this m producing epsilon 9 line number 6. So, it is 7.

So, this fifth location will get corrected. So, it will have a code correction, where this fifth line will be modified like if c goto 7. So, this correction is done. This a correction and then it will be having this B dot true list B dot true list will be made equal to what ever true list we have with a from the second B. So, this 8 true list that is equal to 8. And B dot false list this will be merger of this b 1 false list and b 2 false list that is 5 and 8 false list.

So, 5 false list is 6 and this is 7. So, 6 and 7 will get merged. So, you will get false list as this one. So, this is reduction number 9. At reduction number 10. So, this B or and for B or again there is a back patching. So, for B or. So, it is back patch B 1 false list with m quad. So, B 1 false list. So, this is reduction 3 a is reduction 3 reduction 3 false list is 4. So, that will be corrected. So, the location 4 will now get corrected with go 2 that code will be modified. So, it will become goto 5 because this 4 if this m producing epsilon is a force. So, corresponding m dot quad is 5. So, this 5 is filled up. So, this is again another back patching that is done and then this true list and false list.

So, B dot true list will be merger of the true list. So, B dot true list will be merger of this of 3 and this 9. So, this 2 true list will be merged. So, 3 true list is 3 and 9 true list is 8. So, 3 and 8 will get merged. So, create the true list. And B dot false list will be B 2 dot false list that is 6 and 7. So, this is done at reduction number 10. At reduction number 11 is within bracket B.

So, this true list and false list will get copied. So, B dot true list equal to 3 8 and B dot false list equal to 6 7. So, they are copied and at reduction number 12. So, this is again. And so, first of all there will be a back patching and then it will do something. So, and

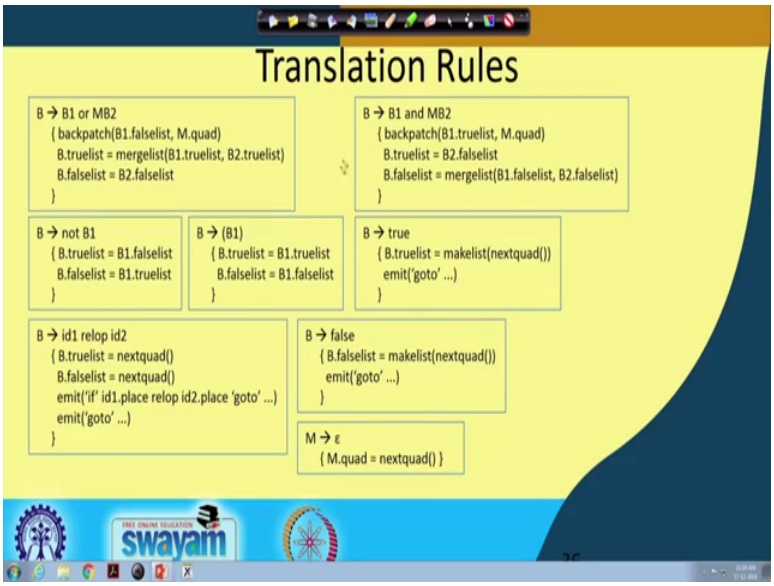
so, it will back patch B 1 true list with m quad. So, B 1 true list. So, B 1 true list it is 1 true list is 1. So, it will back patched with this 3 m quad is 3. So, this back patching will be done that is line number one will get corrected.

So, it will be collected like if a goto 3. And then B dot truly stand B dot false list will be created. B dot true list will be B 2 dot true list that is 3 and 8 and B dot false list will be merger of 2 false list. So, this is the merger of this falls list of one and falls list of 11. So, false list of one is 2 and false list of 11 is 6 and 7. So, they are merged. So, this is the this is the this completes the code generation part. So, you can just write down the final code that is generated. So, the overall code that is generated is like this. At a of set at index 1.

So, we have got if a goto 3 then 2 we have goto and this is not yet filled up , at 3 we have if b goto not yet filled up at 4 it is goto 5 ok, 4 it is goto 5 at 5 it is if c goto 7. At 8 we have, sorry at 6 we have goto at 7 will have if d goto. At 8 we have goto. And then if you consult this final true list and false list so, this one and 3 and 8. So, 3 and 8. So, they will constitute the final true final true list. So, this is 3 and 8 will give me the true list. And this 2 6 7 they give me the false list is 2 6 and 7 they together will give the false list for the overall Boolean expression fine.

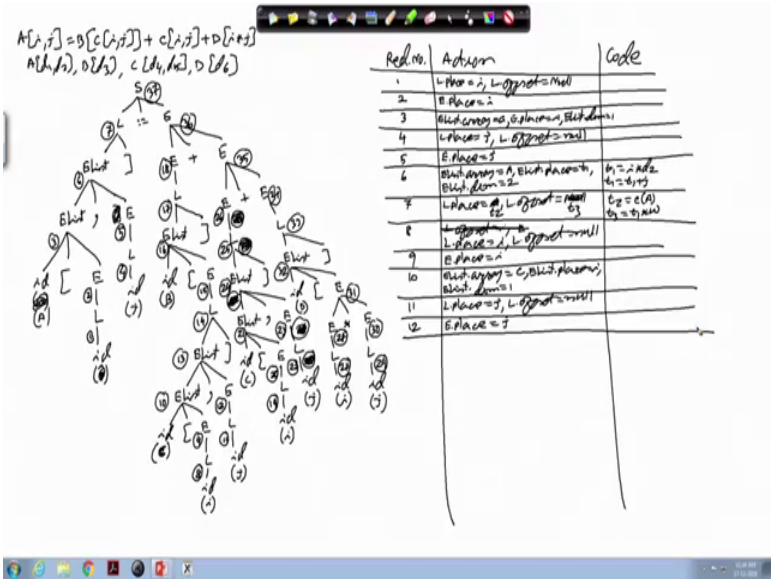
So, you see that. So, you know if you try to match. So, if a then you need to evaluate the second part. So, if a goto 3 now it is taking here if b. So, a and b both are true. So, it goes to the true list. Otherwise if a is false then it definitely false. So, it goes to the false list fine. Similarly, if b is false it comes to 5, and now it check c, if c is true then it goes to 7 here it checks if d then it goes to false list. So, if not of d it will come to line number 8. So, that will goto the true list. So, in this way this piece of code is correct for the original Boolean expression that we had and this 3 address code generation process. So, it has successfully been able to generate the corresponding code. So, next we will look into an array conversion.

(Refer Slide Time: 15:23)



So, this for the array conversion it is a slightly mm tricky in the sense that. So, it is very conversion is slightly tricky because the number of rules that we have is more. So, will try to do it. So, let us see how this can be done.

(Refer Slide Time: 15:48)



So, let us consider an array difference like this. Say A i j equal toB of C i j plus C i j plus D i into j. Suppose this is the overall Boolean expression and this array has a has got dimension d 1 d 2. B is of B is a single dimensional array. So, B is of dimension D 3. C is a 2 dimensional array. So, that is d 4 d 5 and again D is a one dimensional array. So,

that is say d 6. Now the step if you try to generate the corresponding parts tree. So, this is be s producing this is an assignment statement. So, we have got this left side assigned as some expression. Now this left side I have to generate this A i j for generating this A i j. So, this is E list bracket close. Then this E list generates E list comma E E list comma e. So, this E list generates i d bracket start E. This E generates L l generates i d. So, this i d is your i this i d is j sorry, sorry, this i d is a this name of the array a. So, this i d is I this i d is i. Now this E produces L produces i d. So, this is the j part.

Now, for the right hand side, I have got an expression. So, it is E plus E. So, from the first you have to generate this of. So,. So, then this E can give me thus left hand side. So, E producing L,L producing E list bracket close. Then this E list produces i d bracket start and E this i d is B. Now this E gives L. And this L will give that E list bracket close. So, this E E L will give me E list bracket close. This E list will give E list comma E. And then this E list will give i d bracket start E this i d is your c.

This i d is that C array C. Now this E it will give me L which will intern give i d that is your i. And this E will give L that will give i d which is j fine. So, this plus part we have done. Now thus that c i j plus d star i j. So, therefore, that part again I have to have this a E plus E. And from the first you have to generate this c i j. So, for this I have to go like L l producing E list bracket close E list giving E list comma E. Then this E list giving i d bracket start E this i d is c. Now this E gives me L gives me i d which is your i and this E will give L will give i d which is j fine.

Now, this plan E. So, this will again E, E producing L, L producing E list bracket close. Then this E list will give i d bracket start e. So, this i d is d now this E gives, E star E this will give E star E. So, this will give E producing L producing i d. So, that is your I and this E will give L producing i d that is your j. So, this is the full parts tree that will be produced. Now if I want 2 number it.

So, first reduction is this one. Then this, 1 2 3 ok. Then it will do a reduction of this one then this one 4 5 then this reduction 6 then this reduction 7. So, up to this much will be done then. It will be coming to this L producing i d. So, that will be 8, now this is 9. So, this is 10. Now this is a 11 this is 12 this is 13 this is 14 this is 15 this is 16 this is 17 18, now it will be doing this one, 19 then 20 21 22 23 24. Then it will come down here 25 ok. So, it will be. So, 25 26 oh sorry there is a. So, this is not 22. So, this will be a 22 this

will be 22. So, this will be 23. Ok after that this will be 24 yes. So, this is 24 this is 25 this is 26.

Now, this L producing i d. So, this will be 27 28 29 30 31 32 33 34 35 36, 37. So, this will be the whole parts tree with a with reduction numbers written. Now if we try to write down the code generated and the reduction numbers corresponding to that. So, it will be something like this. So, this is the reduction number. So, this is action. And the code that is generated ok code generated. Now at reduction 1 we have this one, L producing i d. So, accordingly it will have this L dot plus equal to i. And L dot of set equal to null this will be null.

And no code is generated. At reduction 2 at reduction 2 it will have this E producing l. So, E dot plus equal to L dot plus. So, E dot plus equal to i. Still no code generated at reduction number 3 it will. So, it reduction number 3. So, there is an array. So, with the E list. So, it will associate E list dot array as a, then E list dot place as i and E list dot dimension equal to 1 dimension equal to 1. So, this is reduction number 3. At reduction number 4 it is doing this L producing i d. So, again same thing L dot plus equal to j and L dot of set equal to null at reduction number 5 it will place that E dot place.

So, at reduction number 5 we have this one E producing l. So, E dot place will be equal to L dot place that is j that is reduction number 5. At reduction number 6. So, it will have E list dot array at reduction number 6. It will have E list dot array equal to A then E list dot place E list dot place will be a new temporary t 1 and E list dot d dimension will be equal to 2. And it will generate a code that t 1 equal to I into d 2 and t 1 equal to t 1 plus j. So, this 2 lines of code if you consult the corresponding array rules then these 2 lines of code will be generated.

Now, reduction number 7 it will do L dot place equal to i. And L dot offset equal to null. And it will have this t 2 sorry a at reduction number 7. So, L dot place will be equal to a new temporary t 2 and L dot offset will be equal to another new temporary t 3. And then t 2 is equal to the constant part of a and t 3 equal to t 1 multiplied by width. So, we assume that width of all elements are same all arrays. So, then at 8 it will have my a reduction number 8. So, L producing i d.

So, L dot offset equal to i and L dot at. So, L dot place equal to i and L dot offset equal to null. At reduction number 9, will have this E dot place equal to L dot place. So, this E dot

place is equal to i ok. So, that way it will do up to even read reduction number 9. At reduction number 10 at reduction number 10, it will have this that is this one. So, E list etcetera. So, this E list dot array will be equal to c because is the c array.

Then E list dot place equal to i E list dot place equal to i and E list dot dimension will be equal to 1. So, no code is generated here reduction number 11. So, L dot place equal to j. And L dot offset equal to null. At reduction number 12, it will do E dot place. So, at reduction number 12 is this one E dot place equal to L dot place. So, that is equal to j. So, this way it will continue. So, will continue with this example in the next class.