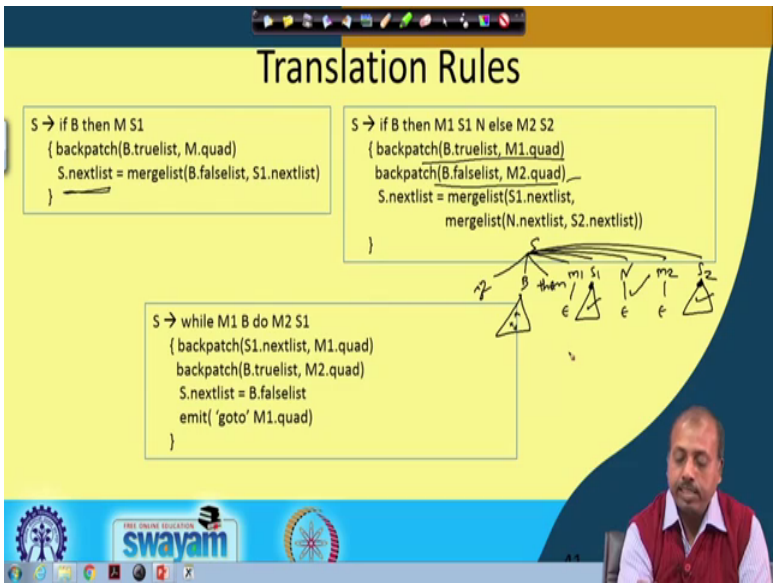
Compiler Design
Prof. Santanu Chattopadhyay
Department of E & Ec Engineering
Indian Institute of Technology, Kharagpur

Lecture - 57
Intermediate Code Generation (Contd.)

(Refer Slide Time: 00:20)



So, similar to the if then statement so we can now look into if then else statement. Only thing is that now the else part is there so, it becomes slightly more complex. So, if I look into the parse tree, then the parsetree is something like this S producing if then this Boolean expression B, then the key word then M1 S1 N M 2 N S 2.

So, by this time the B has already been reduced this S 1 has already been reduced M 1 M and N so, they have already been reduced to epsilon and this is also done. And by the process of this reduction we know that M 1 dot quad will have the start quadruple index for S 1 and M 2 dot quad will have the start index or start of set of S 2. So, they are known with that we are going to generate code for this one.

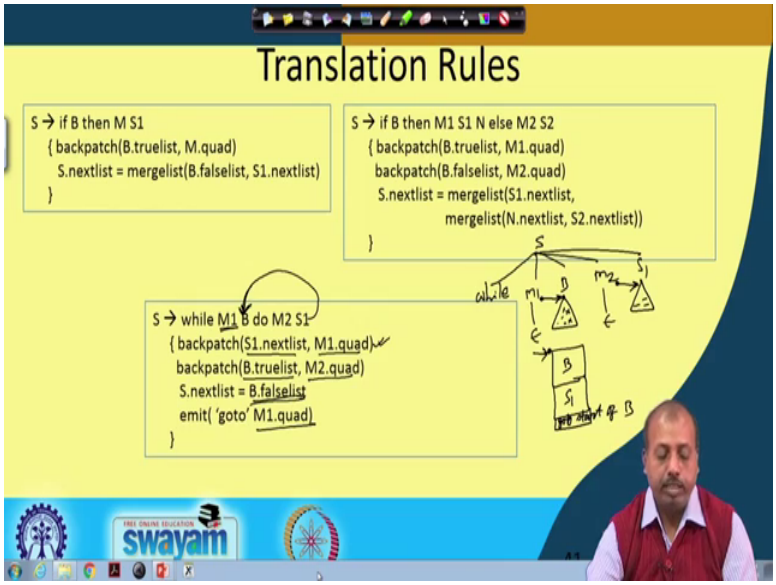
And now you can understand that is you know similar fashion so wherever this B was true so, all those places the go to target. So, I can backpatch them with this M 1 dot quad, because M 1 quad will have the start address of S 1. So, we backpatch B true list with M 1 quad then B false list can be backpatch with M 2 quad. So, in the previous if then part so, you see that we could not backpatch B false list because B false list target was not

known at that point. So, after this statement if I have the next statement then only I will be able to backpatch that. So, it was kept as the in as a part of the next list.

But in this if then else statement so, if this condition is false then the target is known. So, it has to go to S 2 and task S 2 start offset is available in M 2 dot quad. So, this B dot false list can be backpatch with M 2 dot quad. And what is the next list that way at which points I have to have to go to the next statement of S? So, that is defined by S 1s next list, S 2 next list and N next list.

So, all the three next list so, they need to be combined together and then only I will be getting the overall next list for this S. So, that is the code for this if then else type of statement the generating code for if then else type of statements so, we can have this type of actions. What about while loop? So, while loop is slightly because now I have to; I have to go back like once if the condition is true it comes into execution of S 1 and after S 1 is over; so, I have to go back to the evaluation of B to see whether the condition is still true or not.

(Refer Slide Time: 03:03)



So, how do you do that? So, you see that again the if we draw the tree the parse tree then it will be something like this while M 1 B M2 S 1 and M 1 and M 2 they produce epsilon. So, this B has already been done and this S 1 has already been done. So, this M dot quad actually points to the starting quadruple of B and this M 2 dot quad points to the

starting quadruple of S 1 ok. So, that is the situation with which I am going to generate code for of while.

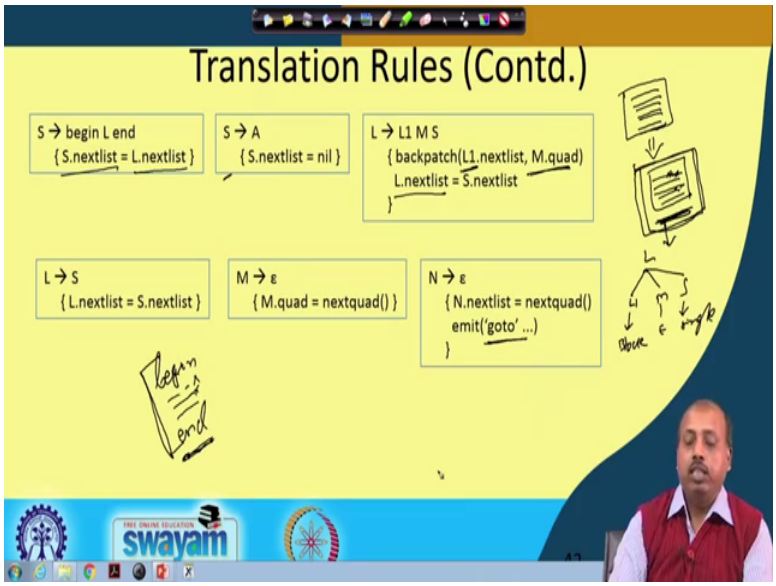
So, what do we do as the first statement with backpatch S 1s next list with M 1 dot quad because as soon as S 1 is over so, I have to recalculate B and B is started address start offset is available in M 1 quad so, that is what is done. So, this S 1 next list is made to S 1 next list is backpatch. So, wherever I had to I had to the next wherever this next was not defined. So, they are now backpatch with the start index of a start starting quadruple of set of S B; so, that is done.

Second thing that you have to do is wherever this B expression is true. So, all those places they should at that places I should go to execution of S 1. So, this Bs true list I have to backpatch with M 2 dot quad, because M 2 dot quad is having the start address of S 1, so, all those places can be corrected and S next list. So, for the S next place it so, it comes out of this S only when this B becomes false ok. So, all the points where B is false ok; so, all the points where B is false so, they are the points to be corrected once I know the next statement of this while.

So, this S next list will have B false list and then it will generate another code. So, go to M 1 dot quad. So, because the so far the code that is generated it has got the code for B and it has got the code for S 1. So, when this S 1 was looked into so, we did not know whether it is going to be part of the while statement. So, at the end I need to generate another goto statement to goto to where goto to this M 1 dot quad. So, M 1 dot quad is having the start starting quadruple index of B.

So, it will be the start of B goto the start of B, so, that is M 1 dot quad. So, it will be done that way. So, the while loop will be implemented in this fashion. So, this major statement that we have seen that this if then else statement if then statement then this while statement; so, they can be done in this fashion.

(Refer Slide Time: 06:16)



So, what about the block of statements like S producing begin L end ok. So, how are you going to do that? So, here this S dot next list is equal to L dot next list; so wherever whenever this next statement will come. So, I have got something like this begin some block of statements and then end and after that some statement will come in my program.

So, wherever this nexts are there this wherever this have to be defined; so they are to be there for this whole begin end block also. So, that is why S dot next is equal to L dot next list. So, if we have got a simple assignment statement like S producing A then there is no next as such. So, initialize this S dot next list to nil and then this one this one is interesting.

So, when we have got one statement followed by another. So, we have got a block so, we are breaking down a block of statements into a structure as if we have got a block of statement followed by a single statement that makes a bigger block.

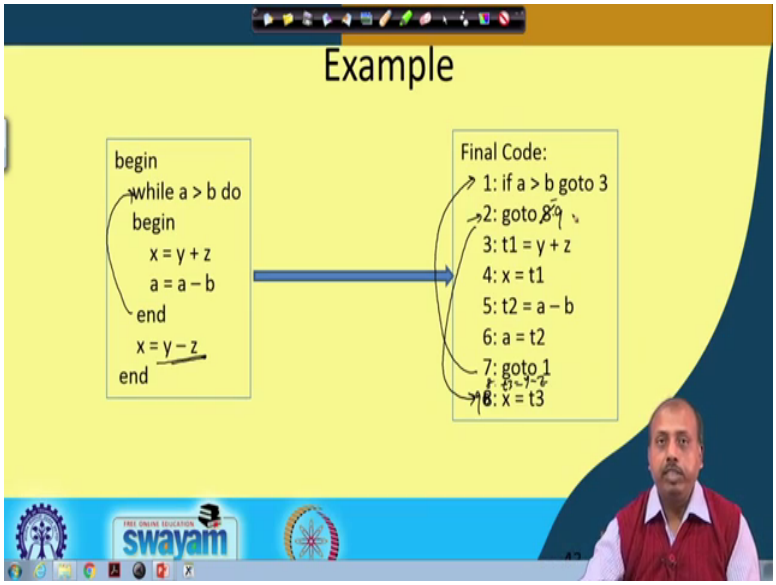
So, this is bigger block is made using a structure like this that we have got a smaller block containing one statement less than the bigger block and the last statement is coming as a new one. So, in that case so, this rule will be something like this L produces this L 1 M and S, where is L 1 is a block of statements and this S is a single statement and this M producing epsilon is there.

So, will be used to hold the starting index of this quadruple of S. Now this L1s next lists; so after L 1 is over after L 1 is over so, it has to go to execution of S. So, all the places where this L 1 next was necessary, so, those places can be backpatched with the start index of S that is M dot quad. And L's next list will be S next list because only the next part will come only from after this S has been executed. So, all this next so, they are pointing to this and for this big block overall block that next list is equal to S next list.

Then L producing S, so, L next list is S next list so, that is very simple. M producing epsilon, so that is M dot quad is next quad and this N producing epsilon so, this is a special case. So, N next list is next quad, but it also generate a goto statement, so, it will be using go to statements. So, we will see that this will be utilised for generating code for different types of instruction like this if then else statement we have seen.

So, this N dot quad so, that will be used for generating the go to after the then part; so, that is shown here. So, that part is shown here that N producing epsilon so, it is generating a goto statement at this point.

(Refer Slide Time: 09:40)

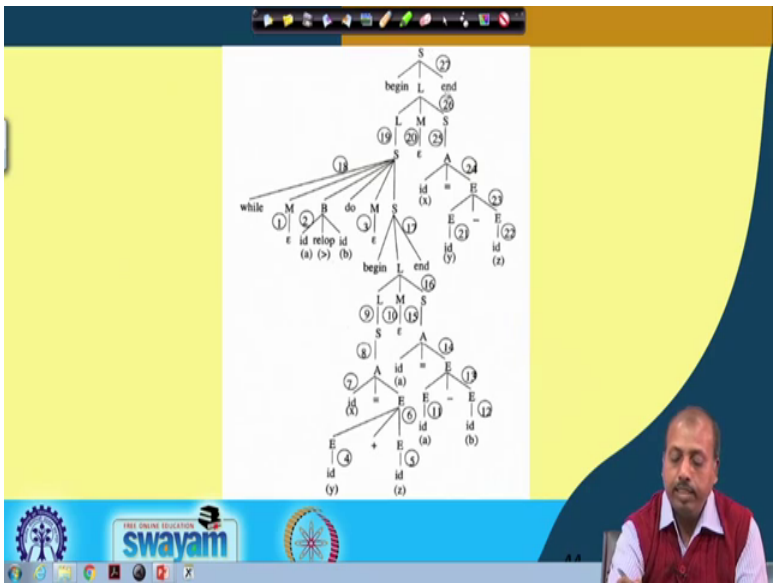


So, next we will be looking into some example. So, we have got a code fragment like this that begin while a greater than b do begin x equal to y plus z a equal to a minus b and x equal to y minus z and end.

So, that is the code fragment. So, the final code, so, we will see how this code is the been generated, but before that you just try to visualise like how this reduction can how this code will finally, look like. So, this while a greater than b. So, it will be generated like if a greater than v goto 3, at 3 it will be doing this x equal to y plus z. So, t 1 equal to y plus z then x equal to t 1 then for this a equal to a minus b; it will be t 2 equal to a minus b and a equal to t 2. And then it will be doing this statement like a equal to t 2 now it generates goto 1. So, from this point it is going back to this, so, that is the while loop.

So, this is a while loop being executed and if a greater than b is not true. So, it will not come to 3, but rather it will come to statement number 2 it will be doing goto 8. So, goto 8 is outside the loop so, this x equal to t 3. So, this, y minus z has to be there. So, there should be another statement here there should be 8 where it is t 3 equal to y minus z and this should be 9; this should be 9. So, we will see how this code is being generated by means of these three address code statements.

(Refer Slide Time: 11:30)



So, this is the parse tree ok, so, it is a bit complex. So, S producing begin L end then L producing L M S; so first here we have got a while statement.

So, that is the you have got first you have got a while statement here. So, that we are done begin end parts of this while statement in the block. So, that is while M B; while M B do M S that is a while statement M producing epsilon, B producing id relop id and then this S is another block of statement the body of the while loop.

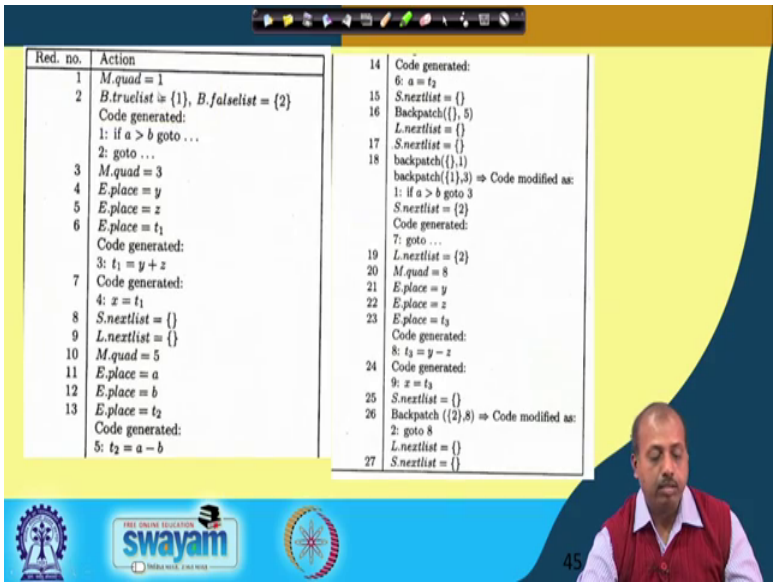
So, body of the while loop has got two statements x equal to y plus z and a equal to a minus b; accordingly I have to generate a block of statements ok. So, this is begin end L so, that is generated and now it is generating L M S. So, L will generate the first statement; so, this will generate the first statement this S will generate the second statement.

Now, how this first statement is generated? So, L produces S, S produces the assignment and that is id equal to E. So, you can see that the expression was something like this that x equal to y plus z. So, this x will be id and this y plus z will be an expression, so, this will be done. So, this assignment id equal to E and this id is x and then we have got this E producing E plus E. So, E plus E giving me ideas id plus id, so, E and E so, that is id producing id producing id.

So, that gives us x equal to y plus z. Now for this part the this other word a equal to a minus b for that part it generates this S producers assignment statement it produces id equal to E, this id is a and this is E minus E then this it id E gives id that is a and this E gives id that is b that is a minus b.

And then after the while loop, so, we have got another statement id equal to x equal to y minus z. So, for that it will be doing like this then it will be generating this parse tree S producing a, a producing id equal to E and E producing E minus E and then why that is E giving id is here so, y and z. So, this is the whole parse tree that has been generated; now how to generate the code for that?

(Refer Slide Time: 13:58)



So, this three address code is a produced is like this ok. So, the three address code produced is like this that this M dot quad so first reduction, so, you have to go by the reduction and then try to understand.

So, the first reduction that is made is this the reduction numbers if you see first reduction is this one, so, M M producing epsilon. So, M dot quad will be assigned as the next quad, so, that is 1. Then this is 2; so id relop id so, you remember that it is it will generate the code that if a greater than etcetera etcetera the true list false list etcetera.

So, how is it done? So, it will be generating a, so, it will generate these two lines of code if a greater than b goto and then line number quadruple 2 goto and it will have this b dot true list equal to 1 and b dot false list equal to 2. So, this is the code for the b part. Now deduction number 3 deduction; deduction number 3 is M producing epsilon. So, M dot quad will be equal to the next quad, so, M dot quad equal to 3. Now reduction number 4; so, reduction number 4 is your E producing id and then we have got this E producing id, so, E dot place equal to id dot place so, that was the action so for arithmetic expression.

So, E dot place E dot place is equal to y. Now at now at this point so, the reduction number 5. So, E producing id again E dot place equal to id dot place that is z. So, E dot place equal to z then we have got this reduction number 6.

So, reduction number 6 is this one E producing E plus E and when this E producing E plus E is coming then I have to get a new temporary variable at this point. So, if you remember the the code generation process for arithmetic expression. So, it was first generating E dot place equal to new temp is t 1 and then it will generate a code that t 1 equal to y plus z.

So, E 1 dot place plus E 2 dot place so, that is generated. Now reduction number 7, so, reduction number 7 gives us this one this assignment A producing id A producing id equal to E. So, for that it has to generate the code that id dot place equal to E dot place. So, id dot place is x that is equal to t 1 so, this code is generated. Now at reduction number 8 I have got S producing A, so, S producing A so, S dot next list equal to null. So, that was the action here, so, S dot next list equal to null. So, that was the action here so, S dot next list is equal to null; so that is done so, S dot next list is null.

Now, reduction number 9, so, L dot next list is S dot next list; so, that was the action. So, L dot next list equal to S dot next list. So, by that so, this is having L. So, this will also have L dot next list equal to that is also equal to; that is also equal to null. Now reduction number 10, so, reduction number 10 is M producing epsilon. So, M dot quad will get the next quad value that is 5, upto 4 we have generated code so, the next squad index is 5.

So, that is M dot quad equal to 5. Now reduction number 11, E producing id. So, this is E dot place is equal to id dot place so, that is done. Now E dot place equal to a similarly reduction 12 E dot place equal to b, reduction number 13 so, it is E minus so, it has to get a new temp and then generate a code.

So, this new temp is obtained like this that E dot place is equal to t 2 the new temporary and now it will generate a code at quadruple index 5 that t 2 equal to E 1 dot place minus E 2 dot place that is a minus b; so, that is reduction number 13. Now, reduction number 14; so, reduction number 14 is this one A assignment statement id equal to E.

So, that will be id dot place equal to it; it will generate this code that id dot place equal to E dot place, so, this a equal to t 2. So, this code is generated at quadruple index 6. Now then reduction number 15 S producing A so, S producing assignment. So, this so, that is a S dot next list equal to null.

So, that will be done, so, S next list is null. Now at reduction number 16; so, at reduction number 16 we have got L producing L M S. So, for that L producing L M S it will backpatch L 1 next list with M dot quad. So, it will backpatch L 1 next list that is reduction 9s next list with reduction 10s M dot quad. So, reduction 9s next list is null and reduction that is 5 so, that is the backpatch null with 5, so, nothing happens.

So, that is no backpatching is done and this L dot next list will be equal to null so, that is there. And then reduction number 17, so, at reduction number 17 so, it is begin L end. So, this particular block and then for the begin L end box. So, S dot next list equal to L dot next list. So, as a result this S dot next list becomes equal to null. Now reduction number 18, so this is the while loop ok. So, first so, while loop actions were like this that so backpatch S 1 next list M 1 quad.

So, see if we do that. So, S 1s next list so, this is 17th next list with backpatched with threes sorry so, reduction 1s M dot quad. So, 17th next list 7ths next list is null so, that has to be backpatched with 1 so, nothing happens ok. So, this is null nothing happens, but there is another backpatch at this point B true list with M 2 dot quad.

So, B true list is that is reduction number 2 with 3 quad ok. So, B true list is 1, so, that will be backpatched with these three, so, backpatch this list 1 with 3. And naturally the line number 1, so, this code will be modified if a greater than b go to 2. So, this this part which was unspecified previously now gets defined so, that is done.

Now after that this S next S next list equal to B false list. So, B false list was equal to 2 so, this S next list becomes equal to 2 and then it will generate a code sorry it will generate a code go to M 1 dot quad. So, it will generate this particular line of code this goto will be generated goto.

So, that is so, this S dot next list is null and then it will generate a goto part. So, this goto will be generated and then we can; so it should be goto 1 I think go to M 1 quad. So, M 1 quad is equal to 1. So, that way this goto 1 this statement that was here so, this is generated at this point. So, this 1 should be goto 1 not blank, this should be goto 1.

So, that while loop is done we have got N the statement. So, this while loop has been done, now we have got say this part. So, that is 17 is so, 18 is done now this 19.

So, L producing S; so, L producing S so, that is L dot next list is S dot next list. So, that way L dot next list is; L dot next list is equal to S dot next list that is equal to 2 that is done. Now reduction number 20, so, reduction number 20 is M producing epsilon. So, this M dot quad will be next quad that is equal to 8. Now reduction number 21, so, this E dot place will be equal to id dot place, so, E dot place will be equal to 21.

So, E dot place equal to y then at 22 E dot place will be made equal to z, now at 23 at 23 so, this reduction will be done. So, it will generate we will take a new temporary, it will take a new temporary and then it will be generating the code of E 1 dot place minus E 2 dot place.

So, this will be the thig that is new temporary t 3 is generated that is E dot place and then this t 3 equal to y minus z that is done. Now at 24, at 24 so, it will have this id equal to E. So, this id dot place equal to E dot place so, that code has to be generated. So, it generates that t 3 equal to y minus z. Now come to reduction number 25, this S producing A; so, this S producing A so, this will have this sorry sorry sorry this 24.

So, that is the id equal to; id equal to E. So, id id dot place will get E dot place that code has to be generated. So, this S equal to t 3 that is generated at quadruple index 9 and then at 25 so, S producing A so, dot next list equal to nil, so, that is done. Now at 26; so, this doing this L M S so, it will be back patching something this L M S the rule was like this backpatch L 1 next list with M quad.

So, L 1s next list, so, 19s next list will be backpatched with M quad. So, that way it will be back patching 2 with 8. So, the code will be modified as go to 8 ok, so, this code will be generated. And now if this L dot next list is null and finally, at 27 so, this S S S dot next list will be L dot next list and that is equal to null. So, we get the code in this fashion.

So, that matches with this one and then this go to 8 is generated at the if this if as part of this if then else this to 8 is generated. So, this is the final code that is generated and it is generated in this fashion.

So, this is the modified code of line number 2. So, it was go to blank here and that is backpatched with the value 8 at this point. So, this way it is a bit cumbersome but you see this is mechanical. So, you can just if you have the set of rules. So, you can always

do that and you can generate the corresponding set of code. Now only thing is that you have to remember the attributes properly and you have to apply those rules with patience so, but all these corrections that we are doing. So, ultimately for the grammar you see that the grammar modification and introduction of these rules. So, they are making it easy for generating the code.

So, we will be doing a good number of example. So, then it will be very clear like how this translation is going to going to be done automatically. So, first thing is that you have to draw a proper part tree then you have to identify the sequence in which the reductions will be done.

So, the first challenge is drawing this parts tree and once you have drawn the parse tree the next challenge is to identify the reduction order like looking into this parse tree. So, you have to start looking from the leaf nodes and then you see the which reduction from the left side can be done at the beginning.

So, that way you will get this reductions, like this M producing epsilon happens to be the first reduction then this happens to the second deduction. So, if if there is a mistake in this numbering then of course, you will not be able to do it by hand so, but this as far as the (Refer Time: 27:47) is concerned. So, it will not do any mistake because this S L are this L are (Refer Time: 27:51) so, they will always do the reductions in this way only.

But when you are doing an exercise by hand you have to be very careful that we do not; we do not get do a mistake in the numbering this reductions. And once the reductions have been numbered so, we have to look into the corresponding; you have to look into the corresponding a rules and the translation rules and then you have to apply those rules.

And you see that apart so, there are two parts one is at a different rules there may be some code generated portion and their maybe some portion which is corresponding to the actual some attributes. So, all these a so all these non terminals; so for this code generation purpose so, we have associated several attributes like true list, false list, next list, true false.

So, all those pointers then place of sets so, like that quad. So, like that there are many such attributes that we have associated with the non terminal. So, how to calculate those

attributes; so that is very critical like if again if there is a mistake in calculation of those offsets then also it will not give you the correct code generated.

So, that way you have to be a bit careful, but it is a mechanical process as far as automation is concerned there is no problem; as far as doing it manually is concerned, so, it is a combustion. Anyway so, next we will be looking into some other issues like function calls etcetera and how to generate the corresponding three address code in the next class.