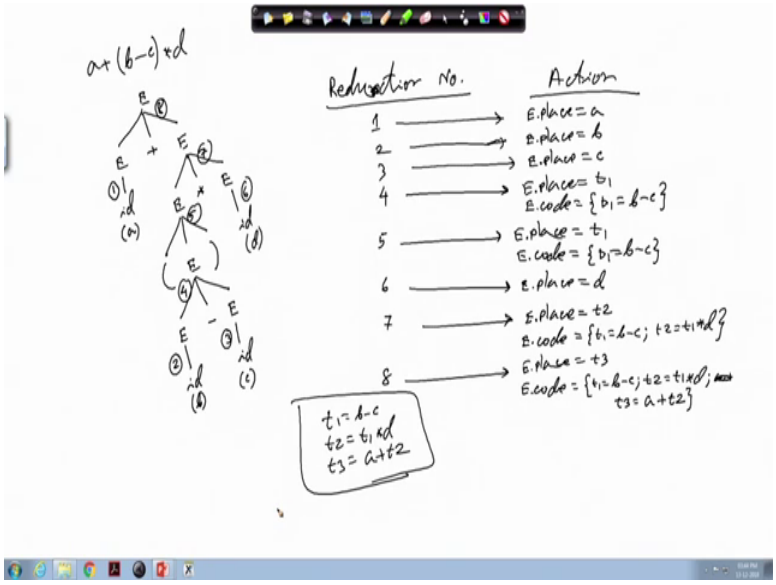
**Compiler Design**
**Prof. Santanu Chattopadhyay**
**Department of E & EC Engineering**
**Indian Institute of Technology, Kharagpur**

**Lecture - 59**
**Intermediate Code Generation (Contd.)**

So, next we will do some exercises on this 3 address code generation procedure. And as I told you just like doing the parsing; so this is also a very combustion job and you have to have patience for solving problems on this particular topic. So, first we take an example where we are having this arithmetic expression say a plus b minus c into d.

(Refer Slide Time: 00:33)




So, simple arithmetic expression like this. So, first we try to draw the arithmetic parse tree for this. So, this is E producing E plus E; then this E can give me id which is a and then this E from this E I have to generate this E star E. So, this E can give me within bracket E; then this E will give me E minus E, then this E will give me id which is b and this E will give me id which is c fine and then this E will give me id which is d.

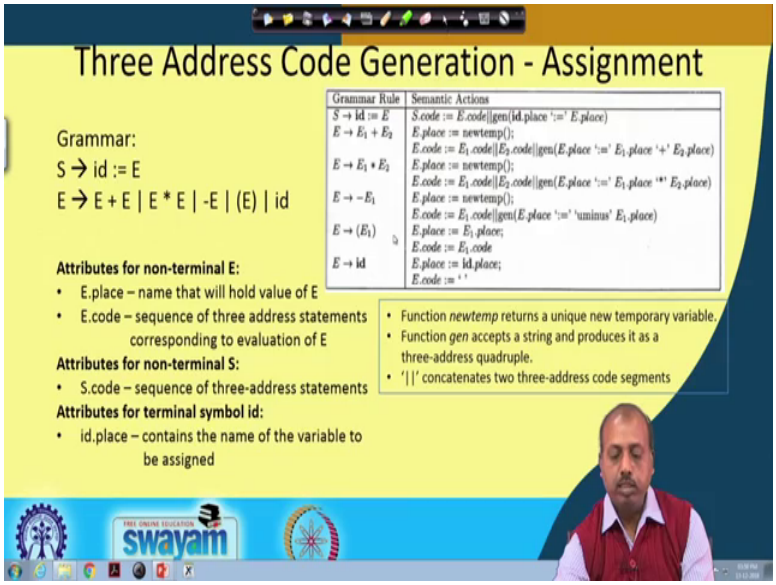
So, this is the parse tree that is produced. So, next job is to number the reductions; so this reduction will be done at the beginning. So, this is the this is a 1st reduction then this is the 2nd reduction; this is the 3rd reduction, after that this is the 4th reduction, this is the 5th reduction, this is the 6th one, this is 7th one, this is 8th one. So, how am I doing this?

So, I am just looking from the leaf nodes from leaf reductions from left to right and whichever reduction is possible I am marking them.

So, this is possible, so I marked it; after that you see the only thing that is possible is this one. So, I marked it as 2, after that only thing that is possible is this one. So, though this one is also possible, but you see that or to if you have looking from the left then this comes first. So, the l r parsing, so it will be it will be doing this reduction first because it does a left to right scan. So, it will do this thing then, so that way so if you just do it; so will you will be getting the reductions in this fashion.

Now I have to the corresponding action if you try to look into. So, you can mark the reduction number; the reduction number and the corresponding action fine.

(Refer Slide Time: 03:18)




So, at reduction number 1; so, E producing id; so, if you look into the rule. So, it says that E producing id; so this is E dot plus equal to id dot plus and E dot code is null. So, we can do that; so E dot plus equal to id dot plus. So, E dot plus equal to a and code is null. Similarly at reduction number 2, I will have this E dot plus equal to b fine, at then reduction number 3, I will have E dot plus equal to c.

Now this one; so, this now reduction number 4 E minus E. Now, I have to get a new temporary variable. So, if you look into the rule again; so E plus E and E minus E they are same. So, this E dot plus is equal to new temp and then E dot code is concatenation

of E 1 code, E 2 code and generate E dot plus equal to E 1 dot plus E dot plus. So, that way; so it will generate this new code E 1 dot; E 1 dot plus E 2 dot plus. So, at 4 at reduction number 4, I will have this E dot plus equal to t 1 and I will generate a code that E dot code; it will have t 1 equal to E 1 dot plus minus E 2 dot plus that is b minus c.

So, this code will be there; now reduction number 5, at reduction number 5 I will have this E reproducing within bracket E. So, this E dot plus equal to t 1; E dot plus equal to E 1 dot plus. So, that is equal to t 1 and E dot code equal to E 1 dot code; that is t 1 equal to b minus c, so that will be there. Now at reduction number 6, so I will have E producing id. So, this E dot plus equal to id dot plus; so that is d.

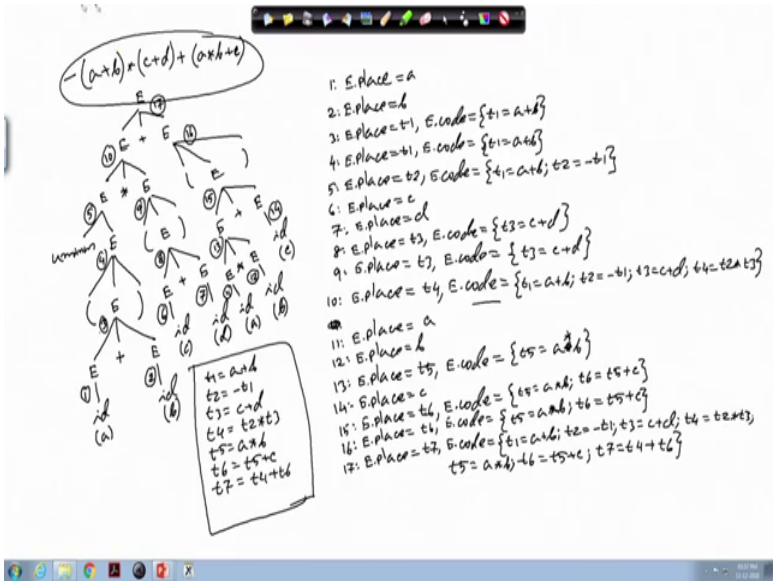
Now reduction number 7 E star E; so it has to get a new temp. So, E dot plus equal to new temp that is t 2 and then it will be having some code generator; E dot code equal to E 1 code E 2 code and this multiplication. So, 5 code; 5 code is t 1 equal to b minus c then 6 code, 6 code is null. So, there is nothing and then it will be generating another code that t 2 equal to E 1 dot plus multiplied by E 2 dot plus.

So, E 1 dot plus is E 1 dot plus from 5; that is t 1. So, that is t 1 multiplied by E 2 plus that is d; so, this will have this is reduction number 7. Now reduction number 8; in reduction number 8 it is E plus E; so a new temporary has to be generated. So, E dot plus equal to t 3 and then it will have the code like E dot code equal to E 1 dot code; E 1 dot code is null for reduction 1 it is null, E 2 dot code that is 7; 7's code is t 1 equal to b minus c, t 2 equal to t 1 into d.

And with that we will add another code which is this addition E 1 dot plus E 2 dot plus. So, E 1 dot plus is a E 1 dot plus is so sorry, this E this E dot plus that is t 3 equal to a plus t 2 because this 7 plus 7 plus is t 2. So, it will be doing like this. So, this is the final code that is generated; so we have got t 1 equal to b minus c then t 2 equal to t 1 star d and then t 3 equal to a plus t 2. So, this is the final piece of code that is generated.

So, you can; so you can generate three address code in this fashion. So, next we will be looking into another example which is again for arithmetic expression, but it is slightly more complex ok.

(Refer Slide Time: 08:27)



So, this is for the expression minus a plus b into c plus d plus a star b plus c; this is the expression now how to do this? So, the first of all I have to draw the parse tree. So, E producing E plus E and then this E should give me this one.

So, that is E star E; now this E should give me this unary minus E, unary minus and this is E, this E should give me within bracket E; this E should give me E plus E that is id plus id. So, this id is a, this id is b fine. Now, this E should give me this c plus d; so, how to get it? So, this is within bracket E bracket close. So, this should give me E plus E; E producing id E producing id; so that is c, this is d ok.

Now, this one; so this will give us within bracket E bracket close and then again this plus has to be generated. So, this is E plus E; so this E should give me E star E, this E should give me id that is a; this E should give me id that is b and then this E should give me id which is c; this is the whole parse stream. Now if I number the reductions, then this is the 1st reduction, this is 2nd, this is 3rd, this is 4th, this is 5th ok.

Now this is 6, this is 7, this is 8, this is 9, this is 10, this is 11, this is 12, then this is 13, this is 14, this is 15, this is 16, this is 17. So, we have got 17 reductions ok; so we can we have marked them like that. Now I have to I have the code generated and the action and the code generation. So, at reduction number 1 ok; so I will have E dot place equal to a; no code is generated and E dot place equal to a.

At reduction number 2; reduction number 2 I will have E dot place equal to b. At reduction number 3, so I need a new temp. So, this E dot place I need a new temporary variable, E dot place equal to t 1. And then I have to generate the code that E dot code is t 1 equal to a plus b. Then reduction number 4 is same E dot place and E dot code; so they will be copied. So, this E dot place equal to t 1 and this E dot code it will be equal to t 1 equal to a plus b.

Now, comes reduction number 5. So, reduction number 5 this unary minus; so I to get a new temporary variable for place. So, this E dot place equal to a new temporary t 2 and E dot code will be this E 1 dot code and then after that the minus of that. So, t 1 equal to a plus b followed by t 2 equal to minus of t 1; so that is a unary minus t 1; so that is reduction number 5.

Now reduction number 6; so this will have E producing id. So, E dot place equal to c then reduction number 7 E dot place equal to d. Then comes reduction number 8; so this is are to get a new temporary E dot place equal to t 3 and after that E dot code will be this E 1 dot code E 2 dot code and then this addition. So, E 1 dot code is nothing, E 2 dot code is; so this for the 6th and 7th; their codes are nothing. So, they only thing; so I have to have this code t 3 equal to c plus d. So, t 3 equal to c plus d code will be there.

Now, reduction number 9; at reduction number 9 so, this is within bracket E, so this E dot place equal to t 3 and E dot code is equal to t 3 equal to c plus d. Now, reduction number 10; at reduction number 10 what we do? It is E star E. So, this I have to get a new temporary. So, this E dot place equal to t 4; then this E dot code E dot plus equal to t 4.

And then E dot code will be this reduction 5 scored that is t 1 equal to a plus b t 2 equal to minus of t 1; after that and then that that is E 1. And for E 2 it is 9; so 9's code is t 3 equal to c plus d; these are put and then I have to have the code of this multiplication that is t 2 multiplied by t 3. So, t 4 equal t 2 multiplied by t 3; so that is the code for this line number 10.

Now, line number 11, reduction number 11; so that is what is reduction number, reduction number 11 is this one. So, E producing id; so this E dot place equal to a. Now reduction number 12 E dot place equal to b reduction number 13 is E star E; so I have to get a new temporary. E dot place is equal to t 5 E dot place equal to t 5.

Then reduction number 13; now I have to have the code part. So, this E dot code will be E ones code. So, E ones code is 13 sorry 13; so this is code for 11; code for 11 is null, code for 12 is also null. So, this is the code that I have is t 5 equal to a plus b. So, that is reduction number sorry a star b ok. Now, reduction number 14; so E producing id. So, this E dot plus equal to id dot plus that is equal to c.

And 15 reduction number 15 is this E plus E. So, I have to get a new temporary E dot plus equal to t 6 and then at I have to have this code generated; so, E dot code. So, code for 13 is t 5 equal to a star b and then I have got code for 12 sorry code for 14, code for 14 is nothing that is null. So, I have to now generate code for t 6 equal to 13 and 14 they are plus; so, that is t 5 plus c.

Now, reduction number 16. So this E dot place equal to t 6 and E dot code equal to this thing that t 5 equal to a star b, t 6 equal to t 5 plus c ok. Now, come 17; so 17 it will have a new temporary. So, this E dot place equal to t 7 and this E dot code; E dot code it will have this E 1 code that is 10 code parts. So, that is this one with t 1 equal to a plus b, t 2 equal to minus of t 1, t 3 equal to c plus d, t 4 equal to t 2 star t 3.

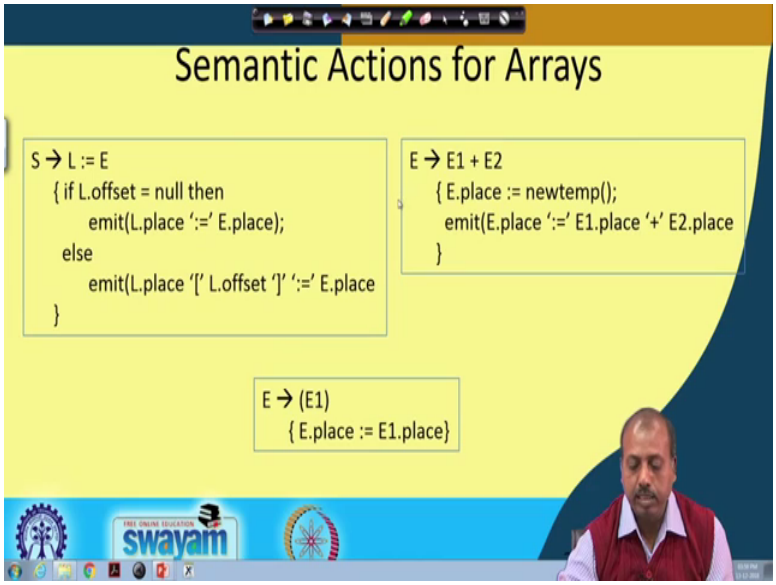
So, that is 10's code part; now 16, 16's code part is t 5 equal to a star b, t 5 equal to a star b, t 6 equal to t 5 plus c and after that I will have this 17 for that E plus E; so I will get this code generated that another line t 7 equal to this reduction 10's E dot place; that is t 4 plus reduction 16's E dot plus that is t 6.

So, this is the code that is finally, generated; t 1 equal to a plus b and t 2 equal to this thing. So, like that so you can generate the corresponding codes. So the final code that I have is this one that t 1 equal to a plus b, t 2 equal to minus of t 1, t 3 equal to c plus d, t 4 equal to t 2 into t 3, then t 5 equal to a star b, t 6 equal to t 5 plus c and t 7 equal to t 4 plus t 6; so this is the code that is generated. And you see that this is perfectly fine with the high level statement that we have here. So, it is perfectly fine with that and it is generating a proper code for that.

Now, the way we are writing the code writing thus this action; so here the rule that we have is making us to write like this, that I have to have this gen part and this code so, it was it was using this gen part. So, they are actually kept the codes are kept in a list sort of thing. So, as I was telling that at many places, we can keep the code in a list or we can use a you can write it on to a file directly. So, if you look into some later rules that we

have discussed in this; in this course like say array and all; so there we will be using this function emit.

(Refer Slide Time: 22:48)




So instead of gen, so we will use the function emit. The difference being that when it is gen; so it is generating the code and when you say emit as if it is writing on to some output file. And that way when it is emit, so it is just appending the code that is generated at this point with the already generated code. Whereas with the gen part, so with every at every non terminal the code part. So, that is holding the full code for that particular non terminal at that point. Like the example that we had taken, so example that we had taken here there we have seen that these gen parts. So, it was used for this E dot place and this thing.

So, this generating for code for arithmetic expression; so you are doing it like this. But here it is a bit combustion sometimes like as we have seen that the same thing we are writing again and again, but that is because of the fact that it is on a list. So, you cannot output half of the list for a particular folder. So, it will not have the other parts with which will it join ok. So, if it can be; so you can always write it in terms of emit functions, but we have to be a bit more careful.

Because as we have seen that in case of Boolean expression; so Boolean expression you can write you using this emit code and you can correct it putting them on a linked list and all, but for this gen code sort of things. So, this gen code sort of things it is in a list,

but in case of emit sort of things. So, it is you can put it on to a file directly. So, you do not have that much of overhead in doing the realization of those codes ok. So, in our next class we will be taking some more examples of this arrays and this complex programming language constructs, Boolean expressions etcetera and we will do some exercise to see how the code can be generated for those complex cases.