Compilers

Lab 9

Piotr Błaszyński

26th April 2022

Tasks (explained later in the document):

- add compilation of loop statements,
  - grammar rules,
  - generate the resulting code,

Loops can be implemented using conditional jump and counter. The counter is a variable, the value of this variable should be modified at the appropriate moment (end of the loop, beginning of the loop (except for the first iteration)).
Input grammar rules for a loop statement (similar to a loop in C, all elements required):

```
for_expr
        :for_begin code_block
        {gen_end_label_and_jump();}
for_begin
        :FOR '(' init_expr ';' cond_expr ';' inc_expr ')
           ' {gen_condition_and_jump();}
```

For the following loop code:

```
for (i = 0 ; i < 10 ; ++i )
{
        z = z + i ;
}
z = z*3;
```

Generate the following code (symbolically):

```
 i=0;
 goto LBL5
LBL6:
 ++i;
LBL5:
 if(i>=10)
   goto LBL7:
 {
     z = z + i ;
 }
 goto LBL6:
LBL7:
 z = z*3;
```

Thanks to the above construction, there is no need to remember the code of conditional and increment expression (the price for such simplification is adding an additional label). The labels are numbered in the order they appear (the jump to label LBL5 appears first). The labels (in the example LBL6 and LBL7) should be stored in the label stack. In the semantic action called after the whole structure (including the code block) of the loop is matched, the labels should be removed and the instruction to jump to the second one should be generated, and the first label should be placed in the code (with a colon).

Generated assembler code (mnemonics):

```
 li $t0,0
 sw $t0,x
 b LBL5
LBL6:
#these 4 lines can be written easier (not necessary)
 lw $t0, i
 li $t1, 1
 add $t0, $t0, $t1
 sw $t0, i
LBL5:
 lw $t2, i
 li $t3, 10
 bge $t2, $t3, LBL7
 lw $t0, z
 lw $t1, i
 add $t0, $t0, $t1
 sw $t0, result1
 lw $t0, result1
 sw $t0, z
 b LBL6
LBL7:
 lw $t0, z
 li $t1, 3
 mul $t0, $t0, $t1
 sw $t0, result2
 lw $t0, result2
 sw $t0, z
```

Note, that this is one possible method of generating code for a loop statement; there are even more possible modifications than for conditional statements. For example, you can change the position of jumps, store other values.