Compilers

Lab 7

Piotr Błaszyński

13th April 2022

Tasks (explained later in the document):

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

A conditional statement (in its simpler form) consists of two parts: a part with a conditional expression and a block of code executed when the condition is met. Rules for the version with else should be developed by yourself.
Base grammar rules for conditional instructions:

```
if_expr
        :if_begin code_block {gen_label_end();}
if_begin
        :IF '(' cond_expr ')' {gen_condition_and_branch
           ();}
```

Labels in MIPS code are identified by a name and a colon symbol; jumping to a label is done by using the jump statement and the label name:

```
    b  LBL42
    li $v0, 6
LBL42:
    li $v0, 5
LBL43:     syscall
    sw $v0, x
```

Each new conditional instruction (including else, later also loop instructions) generates a new label (with the next number). The labels should be put on a stack (a separate label stack), at the moment of matching (semantic action for *if_expr*) of the whole conditional construction the label should be removed from the stack and inserted into the code (insert into the code with a colon, store without). A set of sample jump statements (you can use, for example, statements that compute an expression (seq, etc.)):

```
beq $t0,$t1,label #jump when equal
bne $t0,$t1,label #jump when not equal
bge $t0,$t1,label #jump when $t0 greater, equal
bgt $t0,$t1,label #jump when $t0 greater
ble $t0,$t1,label #jump when $t0 smaller, equal
blt $t0,$t1,label #jump when $t0 smaller
```

For code:

```
z=3;
if (x < 5+4 )
{
        y = 2+4;
        z = 3*3;
}
z = z*3;
```

Generate the following code (symbolically):

```
z=3;
if (!(x < 5+4) )
goto LBL5
{
        y = 2+4;
        z = 3*3;
}
LBL5:z = z*3;
```

And then generate the following code (mnemonics, symbolically):

```
li $t0,3
sw $t0,z
calculation of 5+4 (4 lines)
lw $t2, x
lw $t3, result23 #result(5+4)
bge $t2,$t3,LBL5
#6 lines for y=2+4
#6 lines for z = 3*3;
LBL5: lw $t0,z
#5 remaining lines for z = z*3;
```

Note, this is one possible method of generating code for conditional statements, you can e.g. add an extra label to use natural (inverse to the above) jump statements.