Homework 9 (20 points)

Code generation

Implement a code generator that translates the mini-Pascal programming language into MIPS32 assembly language using the following rules:

1. Traverse the symbol table to generate the data declaration section containing both scalar and array variables; **(2 points)**
2. Load variable references into register `$t0` (integers and booleans) or `$f0` (reals); **(1 point)**
3. Evaluate binary operations in expressions using two registers as follows: **(5 points)**
   a. Recursively evaluate the left operand and make the result available in register `$t0` / `$f0`;
   b. Push the result from register `$t0` / `$f0` onto the stack;
   c. Recursively evaluate the right operand and make the result available in register `$t0` / `$f0`;
   d. Pop the left operand from the stack into register `$t1` / `$f1`;
   e. Perform the operation is performed using the operands from `$t1` and `$t0`;
4. Store the value from register `$t0`, respectively `$f0`, into the memory reference for assignment statements; **(1 point)**
5. Use branch and jump instructions for the for, while and if statements (see the lecture);

   (10 points)
6. Test the generated code using the SPIM simulator available at: http://spimsimulator.sourceforge.net/. See also the documentation available in the SPIM folder in OLAT. **(1 point)**

Example 1: *x = x + y*

```
lw    $t0, x
addi  $sp, $sp, -4
sw    $t0, 0($sp)
lw    $t0, y
lw    $t1, 0($sp)
addi  $sp, $sp, 4
addi  $t0, $t0, $t1
sw    $t0, x
```

Example 2: *((a+b)+c)+((d+e)+f)*

```
lw    $t0, a
addi  $sp, $sp, -4      push
sw    $t0, 0($sp)
lw    $t0, b
lw    $t1, 0($sp)       pop
addi  $sp, $sp, 4
addi  $t0, $t0, $t1     a+b
addi  $sp, $sp, -4      push
sw    $t0, 0($sp)
lw    $t0, c
lw    $t1, 0($sp)       pop
addi  $sp, $sp, 4
addi  $t0, $t0, $t1     (a+b)+c
addi  $sp, $sp, -4      push
sw    $t0, 0($sp)
lw    $t0, d
addi  $sp, $sp, -4      push
sw    $t0, 0($sp)
lw    $t0, e
lw    $t1, 0($sp)       pop
addi  $sp, $sp, 4
addi  $t0, $t0, $t1     d+e
addi  $sp, $sp, -4      push
sw    $t0, 0($sp)
lw    $t0, f
lw    $t1, 0($sp)       pop
addi  $sp, $sp, 4
addi  $t0, $t0, $t1     (d+e)+f
lw    $t1, 0($sp)       pop
addi  $sp, $sp, 4
addi  $t0, $t0, $t1     ((a+b)+c)+((d+e)+f)
```