Compilers

Lab 10

Piotr Błaszyński

25th May 2022

Tasks (explained later in the document):

- add compilation of creating and using static one-dimensional arrays,
  - grammar rules,
  - generate the result code,

The support for **one-dimensional arrays** consists of two components: remembering the array size and referencing the array element (note: recalling that references can occur on the left and right side of expressions).
Basic grammar rules for declarations (simplified compared to C - in C, size can be specified with a constant expression and multiple variables can be declared in one line):

```
arr_decl
 :INT ID '[' LC ']' ';'
       {symbol_table[$2], typ=ARRI, rozmiar=$4; }
 :FLOAT ID '[' LC ']' ';'
       {symbol_table[$2], typ=ARRF, rozmiar=$4; }
```

Basic grammar rules for indexing:

```
arr_expr
        :ID '[' wyr ']'
```

The size of an array in MIPS is specified after a colon (*.word* is better). When referring to an array, multiply the offset from the starting address by 4. The starting address of the array is loaded using *la*. Use round brackets to enter or retrieve the value at the address (e.g. *sw* $\$t0, (\$t4)$)

For code:

```
int a[10];
a[3] = 2;
x = a[2+1];
```

Generate the following code (mnemonics):

```
.data
        a: .word 0:10
        x: .word 0
        result1: .word 0
.text
li $t0, 2
la $t4, a
li $t5, 3
mul $t5, $t5, 4
add $t4, $t4, $t5
sw $t0, ($t4)

li $t0, 2
li $t1, 1
add $t0, $t0, $t1
sw $t0, result1
la $t4, a
lw $t5, result1
mul $t5, $t5, 4
add $t4, $t4, $t5
lw $t0, ($t4)
sw $t0, x
```