Assignment 5 Code Generation

Section 1. Analysis

1.1 Allocate memory for basic types (integer, float).
Test file: input/code_gen/simple_main.src

1.2 Allocate memory for arrays of basic types.
Test file: input/code_gen/basic_array_1.src
input/code_gen/basic_array_2.src

1.3 Allocate memory for objects.
Test file: input/code_gen/object.src

1.4 Allocate memory for arrays of objects.
Test file: input/code_gen/object.src

2.1 Branch to a function's code block, execute the code block, branch back to the calling function.
Test file: input/code_gen/function_call.src

2.2 Pass parameters as local values to the function's code block.
Test file: input/code_gen/function_call.src

2.3 Upon execution of a return statement, pass the return value back to the calling function.
Test file: input/code_gen/function_call.src

2.4 Call to member functions that can use their object's data members.
Test file: input/code_gen/object.src

3 Statement
Test file: input/code_gen/simple_main.src
3.1 Assignment statement: assignment of the resulting value of an expression to a variable, independently of what is the expression to the right of the assignment operator.
3.2 Conditional statement: implementation of a branching mechanism.
3.3 Loop statement: implementation of a branching mechanism.
3.4 Input/output statement

4.1. For arrays of basic types (integer and float), access to an array's elements.
Test file: input/code_gen/basic_array_1.src
input/code_gen/basic_array_2.src
Float is not implemented

4.2. For arrays of objects, access to an array's element's data members.

4.3. For objects, access to members of basic types.
Test file: input/code_gen/object.src

4.4. For objects, access to members of array or object types.

5.1. Computing the value of an entire complex expression.
Test file: input/code_gen/simple_main.src

5.2. Expression involving an array factor whose indexes are themselves expressions.
Test file: input/code_gen/basic_array_1.src

5.3. Expression involving an object factor referring to object members.
Test file: input/code_gen/object.src

Section 2. Design

ComputeMemSizeVisitor.java is implemented to calculate the memory size for each variable and temporary variable. Temporary variable is introduced in this process because operations such addOp, mulOp, function call, etc, require to store intermediate results so they can be used later. Therefore, a temporary variable name generator is implemented in this visitor. Temporary variables together with other variables and parameters are stored in the symbol table, each of them have size and offset for memory access, each scope has scope size as well.

CodeGenVisitor.java is the code generation visitor. It's used to write Moon assembly in .m file. CodeGenVisitor is used after ComputeMemSizeVisitor. It needs the offset calculated in the previous step to access the value from memory. Code generation is implemented using stack based code generation. Register 14 is initialized with topofstack to track the stack frame. The stack frame contains the values of all the local variables declared in the function. To access the variable, use the offset from the symbol table together with register 14. The location of the stack frame is managed by adding/subtracting stack frame sizes, which is also recorded in the symbol table. Before a function is called, register 14 is incremented by the stack frame size of the function. After the function returns the calling function restores the register 14.

Section 3. Use of Tools

No new tools used in the assignment

Assignment Submission

/src - contains the source code

/input/src/code_gen - contains the test files for A5

/output - contains the output files, separated by folder with the test file name as folder name

/compilerdriver.jar - executable

Jar executable is built by Maven using Java11, which requires Java Runtime Environment 11+
If you wish to rebuild the jar, use the pom.xml. Otherwise, you can compile and run the program from source code.

How to run jar executable

Run with default .src files in /input folder

```
$ java -jar compiledriver.jar ./input/src/code_gen/code_gen_bubblesort.src
```

output files will be available in ./output/code_gen/code_gen_bubblesort/