COSC 4P75 Assignment 4

Fall 2011/12

Due: Jan. 16, 2012 @ 12:00 noon.

Code Generation

Write and test the code generator for the project language (http://www.cosc.brocku.ca/Offerings/4P75/Syntax.pdf). The implementation language is Java. Make appropriate use of Java libraries and object-oriented techniques.

The project is a single-pass compiler. Code generation is accomplished by augmenting the classes from the semantic analyzer of assignment 3 and should use the lexical analyzer from assignment 1 as an input filter. It should use the techniques discussed in class. The code generated will be targeted to the instruction set of the virtual machine discussed in class. Sample generated code for the `Employee` and `Main` classes of the Payroll example discussed in class are available in the week 1 examples (http://www.cosc.brocku.ca/Offerings/4P75/Intro.zip).

The main class will be the class `Compiler` used for the semantic analysis in assignment 3. The main method will call the `ClassDcl` parse routine as before after initializing the global scope. The individual parse routines will be augmented with code for code generation. If there is a command line argument, the program should interpret it as the file name of the file to be compiled. If there is no command line argument or more than one argument, the program may proceed in any way determined by the programmer, including terminating with an error.

The source file should have the name `xxx.lng` where `xxx` is the file name specified as the command line argument. The compilation process should produce two text files in the same directory as the source file: `xxx.lis` - a listing file with error messages, and `xxx.spm` - a file for generated code. The global scope should be written (in binary form) to a file `Classes.syt` also in the same directory as the source file(s). To simplify marking, the source lines should also be emitted as comments (beginning with `//`) to the code file as they are processed. That is, the generated code for the line will follow it in the code file. The global symbol table dump introduced in assignment 3 for marking purposes should be removed in this phase.

Develop a set of test files to test code generation. In this case, only valid input need be tested. It is better to have a larger number of small tests than a smaller number of large tests. In each case, it should be easy to determine if the code generator passed the test (comments indicating expected output are a good idea).

The generated code will be executed using the Jamal interpreter. The interpreter and instructions can be downloaded from http://www.cosc.brocku.ca/Offerings/4P75/Jamal.zip. Jamal can be used in both command line and GUI style. For debugging the compiler, GUI style is probably easiest since you can track down where the generated code goes wrong. For final runs, the command line style is easiest.

Compile each of your test files using your compiler and then execute the generated code using Jamal, producing program output. Include in your submission:

- source listing of all classes in your program (in alphabetic order by class name within package)
- listing of each source (`.lng`) test file
- listing of each generated source listing (`.lis`) file
- listing of each generated code (`.spm`) file
- listing of input data (if required)
- listing of program output

Submission

Your submission will be in two parts: electronic and paper. The paper submission should include the listings above and should have a completed coversheet.

In addition to your paper submission, you must make an electronic submission of the assignment. Regardless of how you developed the project, create a directory on sandcastle containing the source (`.java`) and code (`.class`) files for your project. In this directory, at the sandcastle prompt, type:

```
submit4p75
```

and, when prompted, enter the appropriate information using `4` as the assignment number. The submission program will copy the entire working directory, and all subdirectories, to the marking directory, so you should ensure that the content of the directory is what you desire to submit. You could create a special directory for submission and only include what is needed, however, make sure the project will run from the command line in that directory before submitting it.

Note: Since the programs will be run versus a test suite, adherence to the program specification and submission instructions is of paramount importance. Submissions in violation of the guidelines may be subject to a penalty.

As usual, the complete paper submission should be placed in an envelope(s) to which a completed coversheet (http://www.cosc.brocku.ca/coverpage) is attached. The submission should be made to the **instructor's mailbox in the Department Office** in accordance with the assignment guidelines and not in violation of the regulations on plagiarism (http://www.cosc.brocku.ca/about/policies/plagiarism). **Note**: Assignments not including a coversheet will **not** be marked.