

COMPILER DESIGN

PROJECT 1-C

Simple calculator language — Code generation

1 Code generation

Convert the *abstract syntax tree (AST)* constructed during parsing into a sequence of assembly code.

2 Getting started

Begin by grabbing some starting source code at:

`bit.ly/cosc-371-2526f-p1c`

Extract the archive into a new directory. Only a few files, listed below, are included; all of the other Java source code should be copied from your Project 1-B solution.

- **Compiler.java:** Minor additions to `main()` so that the AST, the root of which is contained in the `Program` object, is then called upon to generate code via its `toAssembly()` method. The result is emitted into a `.asm` file.
- **Program.java:** The `toAssembly()` method is provided, emitting *stub code* to surround the code generated by walking the AST. It is during that traversal that each statement in the program should generate its own assembly. The code generated in `Program.toAssembly()` does three basic things:
 1. **Prologue:** Set up the beginning of the assembly file, defining symbols and beginning the code that will make up a `main()` function, within which all other code will be embedded.
 2. **Per-expression:** For each statement in the program, pop its result from the stack and pass it to `printf()`, along with a textual representation of statement's code, so that the program will show each expression and its evaluation.
 3. **Epilogue:** Complete `main()` by having it return. Also, generate a *statics* section that contains each expression as a formatting string as part of what is passed to `printf()` after each expression is evaluated.

- `compile.sh`: A script that, given the name of a Simple Calculator source code file (`.src`), will:
 1. Run our compiler on the source code, producing an assembly file (`.asm`). It also captures any output from your compiler in a log file (`.log`).
 2. Run the assembler (`nasm`) on that assembly file, producing an *object file* (`.o`) of unlinked machine code.
 3. Run the linker (`ld`, via `gcc`) on the object file, producing an *executable* that contains the object file's machine code plus some of gcc's own stub code (e.g., to call `main()`) (no file extension).

3 Your assignment

Complete the code generation for our simple-calculator language. Specifically, add `toAssembly()` methods that traverse the AST, creating and returning (as a `String`) the assembly code corresponding to the program represented by the AST.

A good implementation of the code generator should:

- Correctly generate code that carries out the program.
- Emit code that a human can match (via comments embedded in the assembly) to the source.

To test your compiler on a program, use the `compile.sh` script. For example, if I have a Simple Calculator source code file that looks like this...

```
# Hello

4
(+ 3 1)
(* 20 5)
(* 20 (- 1))
(% 92 60)
```

...then I should be able to compile and run it and see the following...

```
$ ./compile.sh test.src
$ ./test
4 = 4
(+ 3 1) = 4
(* 20 5) = 100
(* 20 (- 1)) = -20
(% 92 60) = 32
```

4 How to submit your work

Copy/upload **all of the Java source code files for this project** in the project-1-C folder in your shared *Google Drive* folder for this course.

This assignment is due on Thursday, Oct-02, 11:59 pm.