Languages and Compilers — Assignment 6
Ristretto IR generation

Due: Tuesday, 16 May 2017, 23:55

This assignment can be done in pairs. You can discuss the assignment with other students, but do not share solutions (except with your partner). Do not share code. Do not look at other students' code. Please read this entire document before starting work on the assignment.

1 Task

In this assignment, you will extend the Ristretto compiler to generate x86 assembly code. I have provided most of the necessary pieces, so you just have to implement IR generation.

The compiler is constructed as follows:

Ristretto source is compiled into ASTs and type-checked. If the code is okay, Ristretto ASTs are translated into IR trees (Drip). Drip trees are translated into a flattened IR closer to assembly language (Perc). Perc is then translated into x86-64 pseudo-assembly. This is identical to assembly code but supports named pseudo-registers as well as real registers. A naive register allocation pass over this pseudo-assembly code results in assembly code that can be compiled to a binary with gcc.

Each of the internal languages has its own abstract syntax and indeed its own parser (which is not used directly, but can be useful for testing). You can look in Drip.syntax, Perc.syntax, and Asm.syntax to see the syntax definitions. AST classes for Drip and Perc are generated during compilation of the compiler. See:

```
target/scala-2.12/src_managed/main/sbt-rats/ristretto/drip/DripSyntax.scala
target/scala-2.12/src_managed/main/sbt-rats/ristretto/perc/PercSyntax.scala
```

The compiler is complete except for the Ristretto to Drip translation. This code is in the object `ristretto.frontend.DripGen`. Some of the cases are filled in to give you an idea of how to write the code. You should complete this code to handle all constructs in the language. Look for occurrences of the `???` expression for where to insert your code. In particular, arrays and control-flow statements are not implemented yet.

To use the compiler, I have provided a script `rcc` that compiles to assembly code and then runs gcc to compile and link the assembly with the runtime library. The assembly code works on macOS and (at least some versions of) Linux. If you are running on a different architecture, you may need to modify ristretto.perc.AsmGen to generate correct assembly code for your OS. Linux should obey the same calling conventions as macOS, so it should work, but we have not tested it extensively.

2 Language specification

The Ristretto language specification can be found on the course webpage. The specification includes typing rules and the grammar.

3 Testing

Some example Ristretto files can be found in the repository. You are encouraged to write your own test cases and share them on Moodle.

4 Submission

Submit your source code and any supporting files. Include a README with build and execution instructions. Be sure to include your name(s) at the top of each source file. We will grade the last submission before the assignment deadline.