University of Arizona, Department of Computer Science

CSc 453 — Assignment 5 — Due Tue Dec 8, 23.59 — 10%

Christian Collberg
November 18, 2009

1 Introduction

Your task is to extend the interpreter from the last assignment to handle procedure calls and to write a code generator for a small subset of LUCA.

2 Procedures [35 points]

Extend the interpreter from the last assignment to handle procedure calls. For this part of the assignment the following rules apply:

- Procedures can be recursive.
- Procedures *cannot* be nested.
- As in the previous assignment, the compiler should be named `lucac` and the interpreter should be named `lucax`. They should be called like this:

```
> lucac x.luc > x.vm
> lucax x.vm
```

You will be given 30 test cases, each worth 1 point. There will be additional secret test cases worth a total of 5 points.

3 Code Generation [35 points]

Extend your LUCA compiler to generate Mips code for the Spim emulator. For this part of the assignment the following rules apply:

- The only types supported are `INTEGER` and `BOOLEAN`.
- There are no procedures.
- There are no structured types (arrays and records).
- All control structures are supported.
- Your compiler should be called `lucam`. To compile and run a program `x.luc` the compiler and `spim` should be called like this:

```
> lucam x.luc > x.s
> spim -f x.s
```

You will be given 30 test cases, each worth 1 point. There will be additional secret test cases worth a total of 5 points.

4 Code Optimization [20 points]

Extend the Mips code generator with a simple "peephole optimizer". It does the following:

1. Examine a "window" of instructions, maybe 3-4 instructions long.
2. Improve the code in the window.
3. Slide window down one instruction
4. Repeat until no more changes can be made.

Implement at least these two optimizing transformations:

Algebraic simplifications: These are some possibilities:

```
x := x + 0;     ⇒
x := x - 0;     ⇒
x := x * 1;     ⇒
x := 1 * 1;     ⇒  x := 1
x := x / 1;     ⇒
f := f / 2.0;   ⇒  f := f * 0.5;
```

Reduction in strength: Multiplications (and divisions) by constants can be replaced by cheaper sequences of shifts and adds:

```
x := x * 32 ⇒ x := SHL(x, 5);

x := x * 100
      ⇓
x := x * (64 + 32 + 4)
      ⇓
x := x * 64 + x *32 + x *4
      ⇓
x := SHL(x,6) + SHL(x,5) + SHL(x,2)
```

Construct some test cases that clearly show off what your code optimizer can do and submit these along with the rest of your files. The README file should explain what your optimizer can do.

5 Code in Luca! [5 points]

Write a *real* program in Luca. By "real" we mean a program that is non-trivial and does something cool. I define what is real and non-trivial; if in doubt talk to me or the TA. (**Hint:** These programs will, of course, be added to our set of test cases. It is therefore in your best interest to write a really nasty program that your compiler can handle but the other teams' compilers cannot. `<-;`.)

The README file should explain what your program does, how to compile and execute it, etc.

6 Put it all together! [5 points]

Pretty up your compiler and interpreter! Add command-line arguments so that you can switch between generating code for the interpreter and the code generator, to turn optimization on and off, to print debugging code, to print help messages, etc. Generate pretty error messages (not XML!). For this part call the compiler `lc` and the interpreter `lx`.

Make an appointment with the TA to show off your compiler! You should try to impress her with how nifty your compiler is, how great code it generates, how well structured the code is, how sweet the Luca program you wrote is, how many test cases you wrote yourself, etc. It's bragging time!

7 Honors Section

Your Mips code generator should be able to handle procedures.

8 Submission and Assessment

- The deadline for this assignment is Tue Dec 8, 23.59. It is worth 10% of your final grade.
- You should submit the assignment electronically to `d2l.arizona.edu`.
- You can work alone or in teams of 2. You must submit a `README` file that lists the members of your team and how much each team member contributed to the assignment.
- You can download 60 test cases from the class website: `http://www.cs.arizona.edu/~collberg/Teaching/453/2009/Assignments/index.html`. Each will give you one point if you get it right and zero points if you get it wrong. No partial credits. We won't check for the correctness of line numbers.
- Your electronic submission *must* contain a working `Makefile`, and *all* the files necessary to build the compiler and interpreter. If your program does not compile "out of the box you *will* receive *zero* (0) points. The TA will *not* try to debug your program or your makefile for you!

Don't show your code to anyone outside your team, don't read anyone else's code, don't discuss the details of your code with anyone. If you need help with the assignment see the TA or the instructor.