University of Arizona, Department of Computer Science

CSc 453 — Assignment 4 — Due Wed Nov 18, 23.59 — 10%

Christian Collberg
October 25, 2009

1 Introduction

You have two tasks to complete this time. First, you should extend your compiler from assignment 3 to also generate code for a virtual machine. Second, you should write an interpreter that reads and executes this virtual machine code.

1. The interpreter should be implemented using *indirect threaded code*.
2. You should write your interpreter in C or C++ using `gcc`.
3. 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
   ```

4. You can design the bytecode yourself. It should be stack-based. To make this a realistic exercise, the bytecode in the `.vm`-file should be written as *integers*, not strings. I.e. the file shouldn't look like this:

   ```
   load x
   load y
   iadd
   store z
   ```

   but rather something like this:

   ```
   10 1 10 2 5 11
   ```

5. You should test the interpreter on lectura.
6. For this assignment you don't have to handle procedures. All other aspects of Luca from the preceding assignments have to be handled.
7. Arrays are indexed from 0; that is, an array declared as `ARRAY 100 OF INTEGER` has the index range `[0..99]`. It is a checked run-time error to go outside these index bounds. You should generate the following error message:

   ```
   <RUNTIME_ERROR pos="3" message="Array index out of range"/>
   ```

   Note that the source code line number is part of the error message.

8. Division by zero should generate this error message:

   ```
   <RUNTIME_ERROR pos="3" message="Division by zero"/>
   ```

   Note that this applies to the mod operator too.

9. For simplicity, all basic types (`INTEGER`, `REAL`, `CHAR`, `BOOLEAN`) are 4 bytes wide. I.e., `INTEGER` and `REAL` correspond to `int` and `float` in C on lectura.

2 Submission and Assessment

- The deadline for this assignment is Wed Nov 18, 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.
- If you work in a team you should only submit one copy of the assignment.
- You can download 70 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 0 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 lexer *and* parser. 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.