COMP 512, Spring 2015
Programming Project: An ILOC Optimizer [1]

1 Introduction

This document describes the programming assignment for COMP 512 at Rice University in the Spring Semester of 2015. For the project, each student will build an optimizer that attempts to improve the running time of a set of *test codes* across a set of *test inputs*. Test codes and test inputs will be provided as early in the project as feasible.

All test codes will be presented in a subset of the **ILOC** intermediate form. The specific subset is described in § 6.1 of the **ILOC** Simulator document, available on the Projects page of the COMP 512 web site. Your optimizer will need to:

⇒ Scan and parse the **ILOC** code

⇒ Build an internal representation for the **ILOC** code

⇒ Apply up to four optimizations of your choice to the **ILOC** code

⇒ Print the transformed **ILOC** code in a form that can be read and executed by the **ILOC** simulator

To measure the effectiveness of your optimziations, you will use the **ILOC** simulator to measure the number of cycles that it takes for both the original and the transformed code to execute on the provided input data files. (You may, of course, create your own input data files as well.0

You will report your decisions and experiences in an oral presentation, expected to last no more than fifteen minutes, during the last week of classes. Your report should include: the specific optimizations you chose to implement, the order in which you executed them, and the number of cycles that the simulator reported for the original and transformed programs. In addition, you should comment on the lessons that you learned in the course of the project.

2 Rules

As in any programming assignment in a for-credit course, there are a number of rules that you must obey. The intent of these rules is to create a fair competition among the labs and to remove ambiguity. As the semester proceeds, we will likely add to these rules.[2]

1. You are expected to do your own implementation work. You are encouraged to discuss the project with your classmates, your advisors, the professor, and any other person whom you believe may have insight. However, the implementation must be your work.

[1] **ILOC** is a compiler intermediate representation described in Appendix A of the book, *Engineering a Compiler* [1]. The specific **ILOC** subset used in this project is described in the documentation for the **ILOC** simulator, which is available on the course web site: `http://www.clear.rice.edu/comp512`

[2] Since this semester is the first run of this particular exercise, it is almost certain that unanticipated issues will arise. We will update this document and post notices to Piazza on any changes to the rules.

2. Your optimizer must run on the CLEAR system. You can develop and debug it on any system available to you. It should, however, run correctly on CLEAR.

3. You may use any programming language available on the CLEAR system to implement your lab, except PERL. You should work in a programming language in which you are comfortable. The runtime speed of your optimizer is not a consideration in grading; it should, however, not be embarassingly slow.

   An optimizer that takes more than a couple of minutes to process any of the examples would be considered embarassingly slow. If your optimizer is embarassingly slow, expect to explain why it is slow in your oral report.

4. The primary goal of your optimizer is to reduce the number of cycles that the simulator reports when the input codes are executed. That is, labs will be compared based on the execution speed of the optimized code that they produce.

3 Software

The **ILOC** simulator is available in `~comp512/Project/Sim` on the CLEAR systems. The directory should include both source and executable versions. The simulator documentation is available on the web page, as well as in the `project` subdirectory with the code. Section 2 of the simulator document explains the available command-line options, including the `-d` option that allows you to specify a test input.

The test codes and test inputs will be available in `~comp512/Project/TestCodes` on CLEAR. The first code, `bsortsmall.i` is already posted. Two test inputs, `bsortsmall1.d` and `bsortsmall2.d`, are posted in the same directory. More test codes and test inputs will be available in the coming weeks.

4 Deadlines

Reports will be made in class during the last week of classes.

5 Questions

Post questions to the class discussion board on Piazza.

References

[1] Keith Cooper and Linda Torczon. *Engineering A Compiler*. Elsevier Morgan-Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2011.