EECS 337

Compiler Design

2009 Fall Semester

Homework 6

Due on Thursday, December 3, 2007

Objective: Refactor the compiler.

Refactor. In this assignment, you will rewrite from scratch or refactor the code that you submitted for assignments 1 through 5. It is your choice as to whether the code is rewritten from scratch or it is refactored. However, you must submit your own code, and do not have the option of using the solutions that are posted on-line (except as a source of inspiration).

Make all modifications and improvements suggested in homework 5. Hand in the revised code and any supporting material to demonstrate your code refactoring (for example, test cases and their output).

The code's internal structure must be modified so that the software is easier to understand, and would be easier to modify and expand in a hypothetical future homework assignment. The code must compile and must run correctly on the sample code at the bottom of page 986. Furthermore, you must submit two additional pieces of source code (roughly 10-20 lines each), and verify that the code runs correctly on your new source code. Every function must have at least one, and possibly multiple test cases. The code must be fully commented. Test your code with valgrind on the 3 pieces of source code and submit valgrind's analysis. Your compiler must have sections that handle errors in the source code.

You need to address any issues that hamper the correctness, readability, and evolvability of your code, including but not limited to: compile-time or run-time errors, missing error-handling, duplicated code, long or overly complex functions.

Code Generation (extra credit). Write a compiler back-end, completing a full end-to-end compiler. The code generator should be comprised at least of:

- A function to find the beginning and end of each basic block
- Functions to initialize, manage, delete, and print register and address descriptors, functions
- `getreg`
- Functions to generate assembler code
- Test cases for all of the above

Hint: for simplicity, assume that array references must be loaded from and stored in memory before and after each use.