Project Assignment 3 - Due May 1, 2020

Overview

This is the final project for the course, and includes everything implemented for the previous two projects. The end result should be a compiler that takes in a subset of the C language and emits x86 assembly.

Documentation

The deliverable should include a document describing the specification and design of the program, along with a user guide. Please include the following sections at a minimum:

Program Overview
Usage
Design discussion with limitations and tradeoffs
Language specification - what do you and don't you recognize
Intermediate language design - This should include an overview of the features and tradeoffs in the representation

In addition, code should be commented reasonably (not excessively), including a comment for each function, and in-function comments for any part of the code that needs additional explanation.

x86 Output

The ideal output should be buildable using GCC's assembler. The assembly file should end with a .s, and can be assembled into an executable using

gcc example.s

Use the output of gcc -S example.c for information on the formatting of the assembly. At a minimum you will want to be able to emit instructions to mov, add, sub, mul, div, call, ret, and a variety of jump instructions. In addition, you may want to include pushes and pops.

Documentation on the assembly file format is here:
https://en.wikibooks.org/wiki/X86_Assembly/GAS_Syntax

Don't get hung up on details of building with GCC if it causes trouble though. I will judge the quality of the assembly output by hand.

Program

The programming language used is up to your project team, but everyone on the team should be able to understand and explain the code. Your program should accept command line arguments dictating its behavior. For this stage, the project should include at least the following command line options.

- An option to output the sequence of tokens and labels.

- An option to output the parse tree. Any type of pretty-printing for a tree is fine. The parse tree does not need to be complete, but should be correct as far as it goes. Please document limitations.

- An option to output the symbol table. This should include a list of discovered symbols and their types. You may use nesting or any other reasonable method to represent different levels of name space.

- An option to output the intermediate representation. This should be a list of instructions in your representation.

- An option to write out the intermediate representation to a file with a specified name.

- An option to read in an intermediate representation specified instead of a source file.

- An option to output a final assembly file.

- An option to turn on optimization, running at least one simple optimization pass.

Anything above and beyond that contributes to the additional 15% outlined below.

Grading

Your grade for this assignment will be a composite of the overall group deliverable and your individual contribution. For the group deliverable, the grade will be based on:

75% - Working program containing minimum requirements
10% - Appropriate documentation
15% - Additional features from the optional feature list

This score will then be scaled based on your level of participation. I will use the git commit logs to determine whether you had a reasonable level of contribution. If there is an issue, I will contact your teammates to get their perspective on your contribution, and may invite you to explain parts of the code to demonstrate understanding if needed.