Benjamin Jones
Compiler Theory
University of Cincinnati
14 March 2012

Final Report

Compiler Description

This compiler is implemented in C, utilizing glib-2.0 for doubly-linked lists and hash-tables, and compiles programs written the programming language specified in the included “project.pdf” file. The scanner (“scanner.c”) is implemented using finite deterministic automata utilizing a generic finite state machine (fsm.c). It produces tokens from the source file serially, making available each new token to the parser. The parser utilizes a recursive descent architecture and is responsible for performing lexical analysis, syntax analysis, type checking, code generation, and error reporting.

Build Procedure

Requirement(s):
 -- maketools
 -- gcc
 -- glib-2.0
 -- pkg-config

Note: The machine code generated is designed for 32bit machines utilizing gcc's -m32 flag. To compile on the x86_64 architecture, 32bit libraries are needed.

Simply run 'make' to generate the executable 'wcompiler.out'

Using the Compiler

Syntax: ./wcompiler.out -f <source_file> [-d]
 -- Option -d provides extensive debugging.

This will generate a file named 'temp.c' in the current directory. To build the generated C, simply run 'make runtime'.

The final executable will be named prgrm.out

Testing

Programs to test the compiler have been written and are in the testPgms directory.

Interesting Notes

Infinite loops were eliminated be removing left recursion in the original grammar. The compiler generates primitive C, with labels and computed gotos. This allows the possibility of supporting many platforms. Error handling has a simple flag that when set can resync the compiler to the next declaration/statement.