Clayton Green (230089109)
kgreen1@unbc.ca
November 20, 2013

CPSC 425 - Compiler Project Phase V

Introduction

Included is a C*13 compiler that partially generates quadruples. Missing is functionality for creating quadruples for many of the statements. What is working is the preamble quadruple and adding the function quadruples and compound statement quadruples.

Participation

Clayton Green - All

Project Status

Scanner - Completed
Parser Basic - Complete
Parser Full - Not Complete
+ Add error recovery (synch)
Semantic Analyzer - Not Complete
+ Missing a few minor checks
Code Generation - Not Complete
+ Missing many statement quadruples

Other TODO
+ Update previous documents
+ Comment source files

Architecture and Design

The code generator takes an annotated AST from the semantic analyzer and generates quadruples. Initially the code generator counts the number of globals and creates the preamble quadruples

```
(start,n,-,-)
(rval,-,-,t1)
(call,main,-,-)
(hlt,-,-,-)
```

Meaning there are *n* globals pushing a return value to the stack for main function, then calls the main function. To create the remaining quadruples the AST is traversed using the visitor pattern.

The code generator requires an AST from the semantic analyzer. So when compiling up to the code generation phase an input file is scanned by the lexical analyzer and those tokens are turned into an AST by the parser. The parser then passes this AST to the semantic analyzer which does type checking on the AST and links variables and their declarations. If at any point one of these phases encounter an error, compilation is stopped and the compiler moves onto the next file after reporting the errors.

Implementation

Currently the code generator does not calculate displacement and offset for the variables and temporaries. Right now it is purely symbolic, meaning that a variable in a quadruple is referenced by its name. Likewise for temporary variables and functions.

Building and Use

Extract:

```
$ tar zxf green-code-generation.tar.gz  -- Compressed tar
$ tar xf green-code-generation.tar      -- Standard tar
```

This extracts the project into a directory named green-parser-basic.

Build:

```
$ cd green-code-generation
$ make
```

This will build the compiler and put the executable in the folder
green-code-generation/Executables/

Run:

```
$ cd Executables
$ ./compiler [Phase] [Messaging] [-e error-file] [-o output-file] input-file
```

Phase

-l	run up to the lexical analyzer phase
-p	run up to the parser phase
-s	run up to the semantic analyzer
-t	run up to the code generation phase

Messaging

-v	verbose

For more indepth information of the compiler use the -h or --help flag.

Code

Code Generator Class

The code generator takes in an annotated AST and generates a list of quadruples. Currently the constructor creates the quadruple list preamble, and the administrator calls the GetCode function to get the code generator to traverse the AST and create quadruples.

AST Node Visitor Code Generator Class

This class follows the visitor patterns and traverses the annotated AST from the semantic analyzer, creating quadruples where appropriate. Traversal is in the order of how quadruples should appear.

Tests and Observations

Add Testing