CIS*4650 (Winter 2016) Compilers

Implementation Project - Checkpoint Three

Due time: Apr. 6, 2016 by 11:59 pm.

Functionality:

For Checkpoint Three, your compiler should be complete, accepting *C*- source programs and generating assembly code suitable for execution on the TM simulator (downloadable from our CourseLink account).

Errors should be handled in the most reasonable way possible, always attempting to recover from the errors and to process the entire program, without segmentation fault or core dumps. Note that it is possible that the process will reach a point where recovery is not possible. In such cases, your compiler should terminate, not dump core. Whenever possible your compiler should attempt to recover from the errors and continue the processing of the source code, possibly detecting multiple errors in the process.

You will find some types of errors are easier to handle in a graceful way than others. You should attempt to do your best under the circumstances, and certainly handle cases that are obvious or easily detected. It is not realistic that you will efficiently handle all possible error conditions (at least not within the scope of one semester course); do what is reasonable and realistic in the time you have available.

Execution and Output

Errors detected by your compiler should be reported to stderr in a consistent and relatively meaningful way. You will find it useful to consider how other compilers you may have used report errors during processing.

You are now responsible for all command-line options specified for the project (-a, -s, -c). In particular, the -c option should now produce assembly language output for the TM simulator. Please refer to the section A.4 of the C- language specification for implementation suggestions of the runtime environments.

Documentation

Your submission will include a document describing what you or your group has done as a whole, and the contribution of each group member to the work. This document should span the entire compiler construction process. However, you can outline any major changes in structure or design that took place over time (i.e. identify significant changes from previous checkpoints in the modules that were implemented earlier). Some of the things you might want to talk about include issues you encountered in translating abstract syntax into assembly code, error handling, problems encountered during design and implementation and their solutions, source language

issues, attempted repair of detected errors if implemented, and the like.

The document itself should not exceed ten double spaced pages (10pt font) or equivalent, not including figures, and should be organized with headings as you see fit. The organization and presentation of this document will form part of your grade.

Test Programs

Each submission should include ten *C*- language programs. These test programs will be used to evaluate the whole project. It should go without saying that you should be submitting programs that work well with your own compiler.

The test files should be named [1234567890].cm. Programs 1.cm through 4.cm should compile without error (producing assembly language output which will be executed on the TM simulator --- note that these programs should represent a variety of levels of complexity, not all easy or brutal). 5.cm through 8.cm should exhibit various lexical and syntactic errors (but no more than 3 per program --- each file should test different specific aspects of your compiler). For 9.cm and 0.cm, anything goes; there is no limit to the number and type of errors in this code.

All test files should have a comment header clearly describing the key aspects of the compiler being tested, including the nature of any errors that may be present.

Makefile

You are responsible for providing a Makefile to compile your program. Typing "make" should result in the compilation of all required components in order to produce an executable program: *cm*. Typing “make clean” should remove all generated files so that you can type “make” again to rebuild your program. You should ensure that all required dependencies are specified correctly in your Makefile.

Deliverables

(1) Documentation (hard-copy) should be submitted at the time of your demonstration.

(2) Electronic submission:
- All source code required to compile and execute your “*cm*” compiler program
- Makefile
- The required README file for build-and-test instructions.
- Tar and gzip all the files above before uploading it to the related dropbox in our CourseLink account by the due time.

(3) Demo: You should schedule a brief meeting of about 15 minutes with the instructor for a demo on April 7. This meeting will allow you to demonstrate your implementation and receive feedback from your instructor.