CIS*4650 (Winter 2020) Compilers

Implementation Project - Checkpoint Three

Due time: Apr. 8, 2020 by 11:59 pm.

Functionality:

With the first two checkpoints, you are able to produce the abstract syntax tree for a valid C- program and identify both syntactic and semantic errors in the given program. For Checkpoint Three, your compiler should be complete, accepting valid *C*- source programs and generating assembly code suitable for execution on the TM simulator (downloadable from our CourseLink account).

Execution and Output

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 code for the TM simulator. Please refer to the related lecture notes for implementation suggestions of the runtime environments.

Documentation

Your submission should include a document describing the entire compiler construction process. In addition to what has been done, the overall design, lessons gained in the implementation process, assumptions and limitations as well as possible improvements, and the contributions of each member if relevant, you can now 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 should be about six double spaced pages (12pt font) or equivalent, and should be organized with suitable title and headings. The organization and presentation of this document will form part of your grade.

Test Environment and Programs

You need to verify your implementation on the Linux server at linux.socs.uoguelph.ca, since that will be the environment used for testing your submissions. To build the implementation incrementally, we recommend that you do the following steps: (a) download and get familiar with the TM Simulator; (b) focus on the expressions and assignments in the “main” function for the initial implementation; (c) expand the implementation to cover control structures, function calls, nested blocks, and arrays; and (d) handle the runtime error for “out of bound” of indexed variables.

In addition, each submission should include ten *C-* programs, which can be named `[1234567890].cm`. In particular, 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 should be included in your submissions since there will be no face-to-face demos for this checkpoint.

(2) Electronic submission:
- All source code required to compile and execute your "*CM*" compiler program
- Makefile and all required test files
- The required README file for build-and-test instructions
- Tar and gzip all the files above and submit it on CourseLink by the due time.

(3) Evaluation: Since the face-to-face meetings are not feasible due to the pandemic of COVID-19, the TA's and I will try to test your submissions ourselves first, and if we can't get your programs working properly, we will contact you for further clarifications. To help the process go smoothly, try to be as complete and accurate as possible in your README file for the build-and-test instructions. Once the marking is done, your mark along with the feedback will be posted in the related dropbox of your CourseLink account. Wish everyone the best in completing the compiler project!