CIS*4650 (Winter 2021) Compilers

Overview of the Implementation Project

The implementation project weighs 45% of your final grade, and it is made of three checkpoints, each worth 15%. The project can be done individually or in a group of two. If you decide to work in a group, you are required to sign up in the Groups page on CourseLink by 11:59 pm on Feb. 8, 2021, informing us the members of your groups. After that time, you are not allowed to change the members of your groups. If you want to work in a group but don't know who is available, you can start a group first as a way of invitation, and other people may contact you or you can contact others who are available. Either way, you are encouraged to talk to each other first and find out if you are likely to collaborate successfully for the completion of the project. If you don't sign up by the due time, we will assume that you want to work individually for the project so that we can roll out the implementation process in time.

Source Language

C- is a simplified C programming language[1] with some advanced features removed so that we can build a compiler within one semester time. Once compiled, C- programs can run on the Target Machine Architecture through the TM (Tiny Machine) simulator. The specification for C- language will be provided in a separate document "C-Specification.pdf". By the end of the semester, you or your group should have a working compiler that takes a C- program as input and produces a TM assembly language program, which can run on the TM simulator for execution.

Target Machine Architecture

Your compiler will produce assembly language output for the TM simulator, which will be explained later in the lecture notes on "TMSimulator". The C source code for the TM simulator will also be provided, which once compiled, can be used to execute the assembly language programs generated by your compiler.

Command-line Options

Your compiler will accept several command line options, only one of which can appear for a given invocation of your compiler. Of course, you will build them up incrementally, and only those related to a given checkpoint are required in the related demos and submissions. Here are all the options to be implemented for the project:

-a : perform syntactic analysis and output an abstract syntax tree (.abs)
-s : perform type checking and output symbol tables (.sym)
-c : compile and output TM assembly language code (.tm)

In each case, the command line option instructs the compiler to perform compilation up to the

step indicated, and output the related results into a file with the same name as the source file, but with the extension of .cm replaced with that indicated for the corresponding option above. Compilation does not need to continue past this point. In the cases where no command line options are given, simply output a message about the usage.

You are free to design the format of your output. However, it should be reasonably complete and understandable, clearly reflecting the internal structures from which it was derived. In the case of -c, the resulting .tm file can be loaded directly into the TM simulator and then gets executed.

Note that outputting the intermediate results for several stages of the compilation process helps us debug a program and show the relevant information during the demos for evaluation. As a result, it will be helpful to show an adequate amount of meaningful information when it is needed, especially when your compiler encounters problems such as core dumps.

Schedule

	Topics	Due dates
Checkpoint 1	Lexical and Syntactic Analysis (scanner and parser)	Mar. 8, 2021
Checkpoint 2	Semantic Analysis (type checking and symbol table)	Mar. 24, 2021
Checkpoint 3	C- Compiler (code generation)	Apr. 9, 2021

Please refer to separate descriptions for each checkpoint with regards to the related submission requirements.

Test Environment

Your project should compile and run on a Linux server at linux.socs.uoguelph.ca since that is the environment we will use to evaluate your demos and submissions. You are free to develop your code in any platform, but it is your responsibility to ensure that your code can run on a Linux server as mentioned above.

---

[1] Taken from “Compiler Construction: Principles and Practice” by Kenneth C. Louden, PWS Publishing Company, 1997.