CS3020/CS6240: Mini-Programming Assignment #2
A Small Introduction to the LLVM Infrastructure
AST, IR and Compiler Options
Due Tuesday October 20th, 2015 at 11:59 PM

Introduction This study is aimed at familiarizing you with the LLVM compiler infrastructure (`http://llvm.org`), which has lot of very interesting research modules with active development from many compiler communities across the world. You will also use it for the next couple of mini-assignments.

It aims to give you some familiarity with LLVM's C-family frontend CLANG (`http://clang.llvm.org/`). More specifically, this assignment asks to you to do a *study* of the AST, IR and options of the LLVM compiler.

- **Download LLVM:** Obtain source code of LLVM and Clang `http://llvm.org/releases/download.html`. Install LLVM. Also have its source-code in a location so that you can *browse* it. Use `cscope` or `tags` facility of linux and vim so that you can read it.

- **LLVM and Clang:** Understand the directory hierarchy of LLVM and Clang. Read the documentation *"Introduction to Clang"* at `http://clang.llvm.org/docs/IntroductionToTheClangAST.html`.

- **AST structure:** Use the options provided with Clang to view the LLVM/Clang-AST for some non-trivial programs. Understand the design of LLVM-AST to some level. Report your findings.

- **AST traversals:** Read the tutorial to write AST visitors at `http://clang.llvm.org/docs/RAVFrontendAction.html`. Write a short note about the visitor mechanism to traverse the AST and do semantic analysis. (The *visitor pattern* is *design pattern*, a paradigm from software engineering which you are using in your ANTLR.)

- **Error messages:** Do a study of how the error messages are handled in LLVM source code, primarily the assert mechanism of LLVM. Report on the handling of some specific error messages.

- **LLVM-IR:** Do a study of the LLVM-IR. Print the IR for some non-trivial programs and study them. Report on your study.

- **Assembly language:** Generate the assembly language codes of some simple C/C++ programs. Understand how the C/C++ compilers *mangle* the names of various entities. Report your findings.

- **Compiler toolchain and options:** For a set of programs, go all the way; print the AST, print the LLVM-IR, print the assembly code. Play with the various compiler frontend/middle-end/optimizer options. Report your findings.

- **Kaleidoscope:** Read the *Kaleidoscope* at documentation `http://llvm.org/docs/tutorial/index.html`. Extend the language in a *minor* way, like adding binary operators, adding more syntax flavors etc. Report your findings.

You do not have to limit yourself to the above list. You are welcome to explore more options of LLVM. For example, you could (i) write some AST visitors to do simple semantic checking, like print the uninitialized variables, enforce *style guidelines* in the source code like COMPASS tool[1] of the ROSE compiler does, (ii) use the *Clang Static Analyzer*[2] for static analysis of some simple programs. You can also play more with the Kaledioscope. (This section is not mandatory.)

As usual, you can do this assignment along with your team member with at most two members per team. It is suggested that you team up with the same partner as the one for your COOL compiler project phases.

Submission You will be evaluated on a *technical report* on your study. You need not submit any code, but, if you wish, you can add the code listings in a separate appendix. The report has to focus on the Frontend, LLVM-AST (its design and its traversals), LLVM-IR, the LLVM compiler options, Kaledioscope aspects, and finally the ASM code generated. The report should be a PDF, preferably generated from a lyx/latex file.

Evaluation Your report should show a *good* understanding on each of the points asked. The usual rules of plagiarism apply to your report. In particular **do not** directly copy the material from manuals/websites. Your report should professionally refer the website(s) from where you downloaded the code and the manual(s) you referred to. Very good reports will fetch bonus points.

[1] `http://rosecompiler.org/compass.pdf`
[2] `http://clang-analyzer.llvm.org/`