CS5863:Assignment #1
An Introduction to the LLVM Infrastructure, IR and Compiler Options

Due February 11th, 2022 at 11:59 PM

Introduction This assignment is aimed at setting up and 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 almost all the assignments/projects of this course.

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 IR and options of the LLVM compiler.

Download LLVM: Obtain source code of LLVM and Clang. Please clone the official GitHub repository of LLVM, corresponding to `release/12.x` branch. Please note that we would use **LLVM version 12** for the purpose of all the assignments/homeworks in this course.

Refer to `http://llvm.org/docs/GettingStarted.html#git-mirror` for instructions (make sure to use appropriate flags to checkout the `release/12.x` branch). Build LLVM and Clang from source. You can refer `http://llvm.org/docs/GettingStarted.html#compiling-the-llvm-suite-source-code`.

1. **LLVM:** Understand the directory hierarchy of LLVM. Read the documentation *"Getting Started with LLVM"* at `http://llvm.org/docs/GettingStarted.html#directory-layout` for LLVM.
2. **LLVM-IR:** Do a study of the LLVM-IR. Print the IR for five non-trivial programs and study them. Submit a report on your study along with the generated *.ll* files.
3. **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.
4. **Compiler toolchain and options**
   (a) For a set of programs, go all the way from the preprocessing stage to binary; print the LLVM-IR, print the assembly code. Play with the various compiler frontend/optimizer options.
   (b) Apply trivial optimizations like *constant propagation* and *dead code elimination.* Print their CFG and Dom Tree and observe how the IR gets transformed.
   (c) Explore the relevant tools in LLVM - `llvm-as, llvm-dis, llc, lli`, etc. with different examples.

   Report your findings. Support your observations with the inputs used and the obtained output.

Submission You will be evaluated on a technical report on your study. Also submit code samples for both C/C++ and .ll files. The report has to focus on the Frontend, LLVM-IR, and the LLVM compiler options. Additional material on LLVM projects and tools (ex. openmp, polly) will be considered for bonus points.

Your submission should be a gzip archive with name Asgn1_ROLLNO.tar.gz containing the following files.

- The report in PDF format, preferably generated from a lyx/LaTeX file.
- The test cases and the outputs obtained (dot files, etc.)
- Single bash script file containing the necessary commands.

  Note: It is expected that on running the script from its directory, we should be able to obtain all the outputs/observations that you have claimed in the report.

Evaluation Your report should show a good understanding on each of the points asked. The rules of plagiarism would be strictly enforced - `https://cse.iith.ac.in/academics/plagiarism-policy.html`. In particular ***do not*** directly copy *any* 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.