# Programming Assignment 1 Part A

## Overview

The goal of this assignment is to familiarize you with the LLVM compiler and the Intel Simulation and Analysis Engine (SAE), which you will be using for other assignments and research projects. This assignment is divided into two parts: a) LLVM tutorial and b) SAE tutorial. This is the documentation for the LLVM tutorial and you will be writing an LLVM pass which can print the LLVM instruction per function. This documentation assumes you are working on a UNIX-like system and are familiar with make+gcc tool chain.

## Installing LLVM From Source

1. We will be using latest LLVM release 3.9.1. The source code can be downloaded from here.
2. Read the documentation about how to install LLVM from source. Specifically pay attention to section "Getting Started Quickly (A Summary)" and "Requirements". Note that the documentation used svn to download source code but we will be using a stable release 3.9.1. Also, read the clang install guide. They are following the same steps but clang documentation is more detailed.
3. Downloading the LLVM and clang source code (you can also download other optional packages). After downloading the source code, you have to decompress and organize these packages in the way required by the documentation.
4. By now, you should have a well-organized, ready-to-build source tree. Now create a separate directory named "build" outside of the LLVM source tree (as described in the clang installation guide).
5. Continue with the step 10 in the section "Getting Started Quickly (A Summary)". You have to install latest CMake (>= CMake 3.4.3) to build the source code. CMake 3.7.2 can be downloaded from here. The compilation has two steps: in the first step, you will run CMake to generate Makefiles and in the second step, you will run make to build the source. After make is finished, run "make check-all" to test your build. Running a serial build will be *slow*. Make sure you run a parallel build; for make, use make -j.
6. After compiling, you can run "make install" to install llvm and clang. You will need root permission if you want to install llvm system-wide.

## Writing an LLVM Pass

1. Read the documentation about "Writing an LLVM Pass," which can be found here. In particular, read the sections "Introduction — What is a pass?" and "Quick Start — Writing hello world," which provides an example of how you can write a pass that prints all the

function names of a C file. However, do not build your pass in the LLVM source tree. Instead use the provided skeleton project. It uses a simple Makefile to build a pass without having to integrate with the LLVM build system.

2. Given a C source file *hello.c*, use the following command to compile it down to LLVM bitcode file *hello.bc* with debugging information included:
   *clang hello.c -c -emit-llvm -O0 -g*
3. Now, given the bitcode file *hello.bc*, We have provided you a skeleton project which can print the bit code file in a textual human-readable format. Run *make* to build the provided tool and it will produce a dynamic library named *IRPrinter.so.* You can now load the tool and test it as follows:
   *opt -load ./IRPrinter.so -disable-output -printIR hello.bc*
4. The output of the provided pass is the same as running "*llvm-dis hello.bc -o -*". It will dump the entire bitcode, including all LLVM function, global variables and metadata. The provided code used *Module.print()* to dump the entire bitcode file, but we are only interested in viewing function-level information, such as function signature, basic blocks, etc.
5. Now you are required to write a pass which can dump the information for each LLVM function inside of the bitcode. Your output should resemble the output from the skeleton project but only prints information per function. Your pass must at least print following things:
   a. Function signature, including function name, return type and arguments (you do not need to support variable number of arguments)
   b. Print every basic block inside the function and for each basic block, print all instructions and its label
   c. Print function prototypes as well as function definitions
   d. For each LLVM instruction, print its corresponding source code line number. You will need to read the subsection "C/C++ front-end specific debug information" of the document "Source Level Debugging with LLVM".

# Submission Guideline

Programming Assignment 1.a is due on the same day, Jan 31st, 11:59pm. You will be submitting all your code, your own test cases (if there is any) and a short report describing your solution and the structure of your archive (less than 1 page).