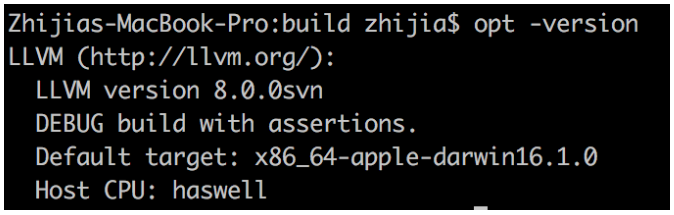
Project 1: Warmup (LLVM)

Points:
- 5 out of 35 (total of project)

Due:
- Jan 17 (Friday) 23:59 (late submission may be accepted within one day, but gets 20% grade deduction)

Objectives:
- install and set up the LLVM 8.0.1 environment
- get familiar with its basic toolchain and flags
- know how to generate some CFGs

Tasks:
- **Step-1**: Install and setup the environment (some installation tips are given later in this doc). Once done, send TA a screen snapshot of a successful installation (use email title: **CS201-2020-Project1-YourStudentNo**). For example, run "`opt -version`" for LLVM

  ```
  Zhijias-MacBook-Pro:build zhijia$ opt -version
  LLVM (http://llvm.org/):
    LLVM version 8.0.0svn
    DEBUG build with assertions.
    Default target: x86_64-apple-darwin16.1.0
    Host CPU: haswell
  ```

  TA will reply you with a small program (`test.c`) to continue the following small tasks.

- **Step-2:** Perform a list of small experiments with the **received program** (`test.c`), including:
  a) learn and try the basic uses of the following commands:
     `clang,opt,llc,lli,llvm-link,llvm-as,llvm-dis`
  b) translate between different code representations using some of the above tools with different flags.
     i) source (.c) to binary (executable)
     ii) source (.c) to objective (.o)
     iii) source (.c) to machine assembly (.s)
     iv) source (.c) to LLVM bitcode (.bc)
     v) source (.c) to LLVM IR (.ll)
     vi) LLVM IR (.ll) to LLVM bitcode (.bc)
     vii) LLVM bitcode (.bc) to LLVM IR (.ll)
     viii) LLVM IR (.ll) to machine assembly (.s)
     ix) interpret the LLVM IR (which directly prints the output without compilation)
  c) generate the CFG(s) of `test.c` (*Tip: you may need to install graphviz and use "`dot -Tpdf`"; Graphviz has issues on OSX Mojave, It works fine if installed using Anaconda.*)

  Reference: The LLVM Compiler Infrastructure

- **Step-3**: Write a report that lists your experiments with commands and inputs/outputs. Submit your report (in PDF) along with your working folder, which includes the temporary files you used and generated for the experiments (try to use different file names so your experiment history would be preserved, instead of being overwritten).

Grading:
- The grading will be mainly based on the completeness of the tasks and the clarity of the report.

LLVM and Clang Installation

Unix System
The following uses macOS as an example (Linux would be similar)

Before Installation:

- Make sure the command line environment works well;
- Make sure the minimum required **Cmake** version 3.4.3 is installed. Check your current Cmake version using command "`cmake --version`". If not, download and install it.

Installation:

- Create your own work directory:
```
$ mkdir mywork
$ cd mywork
```

- Get LLVM 8.0.1:
```
$ wget
https://github.com/llvm/llvm-project/releases/download/llvmorg-8.0.1/llvm-8.0.1.src.tar.xz
$ tar xf llvm-8.0.1.src.tar.xz
$ mv llvm-8.0.1.src llvm/
```

- Get Clang 8.0.1:
```
$ wget
https://github.com/llvm/llvm-project/releases/download/llvmorg-8.0.1/cfe-8.0.1.src.tar.xz
$ tar xf cfe-8.0.1.src.tar.xz
$ mv cfe-8.0.1.src clang/
```

- Build LLVM and Clang:
```
$ mkdir build
$ cd build
$ cmake -G "Unix Makefiles" -DLLVM_ENABLE_PROJECTS=clang
  -DCMAKE_BUILD_TYPE=Release ../llvm
$ make -j 8   #replace 8 with the number of cores on your machine
```
# building process may take from 45 mins to nearly 3 hours, depending on how powerful your machine is
# This builds both LLVM and Clang for release mode

- Add "`build/bin/`" to your PATH
  on macOS, append `export PATH=/PathToYourLLVM/build/bin/:$PATH` to `.bash_profile`
  (similarly set up the environment variables for Windows and Linux)

- Verify the installation
  `$ clang --version` should show your Clang and LLVM version.
  `$ opt -version` should show the LLVM version.

The above is a simplified version of LLVM and Clang installation. See the original and install extra libraries as needed.

- Getting Started: Building and Running Clang
- Building LLVM with CMake

Windows System
For Windows, open the following link
- [Getting Started: Building and Running Clang (Using Visual Studio)]
- [Getting Started with the LLVM System using Microsoft Visual Studio]

Before Installation:

- **CMake**. This is used for generating Visual Studio solution and project files. Get it from: https://cmake.org/download/
- **Visual Studio 2017 or later**
- **Python**. It is used to run the clang test suite. Get it from: https://www.python.org/download/
- **GnuWin32 tools** The Clang and LLVM test suite use various GNU core utilities, such as grep, sed, and find. The gnuwin32 packages are the oldest and most well-tested way to get these tools. However, the MSys utilities provided by git for Windows have been known to work. Cygwin has worked in the past, but is not well tested. If you don't already have the core utilies from some other source, get gnuwin32 from http://getgnuwin32.sourceforge.net/.

Installation:
- Download & unzip the source code of llvm and clang (You may need 7-zip to unzip tar.xz files):
  https://github.com/llvm/llvm-project/releases/download/llvmorg-8.0.1/llvm-8.0.1.src.tar.xz
  https://github.com/llvm/llvm-project/releases/download/llvmorg-8.0.1/cfe-8.0.1.src.tar.xz

- Rename them to 'llvm' & 'clang' directories and put them in 'mywork' directory.

- Build LLVM and Clang (in powershell):
  ```
  $ mkdir build
  $ cd build
  $ cmake -DLLVM_ENABLE_PROJECTS=clang -G "Visual Studio 15 2017" -A x64 -Thost=x64
  ..\llvm
  ```

  -Thost=x64 is required, since the 32-bit linker will run out of memory.
  To generate x86 binaries instead of x64, pass -A Win32.

  ```
  $ cmake --build . --config Release -j 8
    #replace 8 with the number of cores on your machine
  ```

- Add "`build/bin/`" to your PATH

- Verify the installation
  `$ clang --version` should show your Clang and LLVM version.
  `$ opt -version` should show the LLVM version.