

CS201 Project Phase 1

Ravan Nazaraliyev 862393741

Step 1:

I installed and set up the environment. The below screenshots show verification:



```
ravan@ravan-Precision-5820-Tower:~/cs201$ clang --version
clang version 12.0.1
Target: x86_64-unknown-linux-gnu
Thread model: posix
InstalledDir: /home/ravan/cs201/install/bin
ravan@ravan-Precision-5820-Tower:~/cs201$ opt -version
LLVM (http://llvm.org/):
  LLVM version 12.0.1
  Optimized build.
  Default target: x86_64-unknown-linux-gnu
  Host CPU: skylake-avx512
ravan@ravan-Precision-5820-Tower:~/cs201$
```

Step 2: (b)

i) `clang test.c -o test`

Compiles test.c and creates binary executable test

ii) `clang test.c -o test.o`

test.o object file is created from test.c

iii) `clang test.c -S -o test.s`

Assembly code test.s is created for test.c

iv) `clang -emit-llvm test.c -c -o test.bc`

LLVM intermediate bytecode file test.bc is created from test.c

`clang -emit-llvm test.c -S -o test.ll`

LLVM .ll file test.ll is created from test.c

v) `llvm-as test.ll -o testll.bc`

Bytecode file is created from LLVM .ll file

vi) `llvm-dis test.bc -o testbc.ll`

LLVM .ll file is created from bytecode file

vii) `llc test.ll -o testll.s`

Assembly code is created from LLVM .ll file

Name	Size	Modified
 test	16.1 kB	8:03 PM
 test.bc	2.5 kB	8:16 PM
 test.c	331 bytes	5 Nov 2020
 test.ll	3.0 kB	8:17 PM
 test.o	16.1 kB	8:04 PM
 test.s	2.1 kB	8:04 PM
 testbc.ll	3.0 kB	8:19 PM
 testll.bc	2.5 kB	8:19 PM
 testll.s	2.1 kB	8:20 PM

Step 3:

a) In order to create libHelloPass.so, I run “cmake .” command.

```
ravan@ravan-Precision-5820-Tower:~/Downloads/CS201-F23-Template (1)/CS201-F23-Template/Pass/HelloPass$ cmake .
-- The C compiler identification is GNU 9.4.0
-- The CXX compiler identification is GNU 9.4.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found ZLIB: /usr/lib/x86_64-linux-gnu/libz.so (found version "1.2.11")
-- Configuring done (0.6s)
-- Generating done (0.0s)
-- Build files have been written to: /home/ravan/Downloads/CS201-F23-Template (1)/CS201-F23-Template/Pass/HelloPass
```

b) I will use LLVM .ll file from the previous part. The file is testbc.ll

c) I make the following change to the HelloPass.cpp file in order to print number of predecessors and successors of each block:

```

struct HelloPass : public FunctionPass
{
    static char ID;
    HelloPass() : FunctionPass(ID) {}

    bool runOnFunction(Function &F) override
    {
        errs() << "HelloPass runOnFunction: ";
        errs() << F.getName() << "\n";
        int number_of_basic_blocks=0;
        for(auto &basic_block : F)
        {
            int n_predecessor=0, n_successor=0;
            for(BasicBlock *Pred : predecessors(&basic_block)){
                n_predecessor++;
            }

            for(BasicBlock *Pred : successors(&basic_block)){
                n_successor++;
            }

            errs() << "Basic Block : "<<number_of_basic_blocks<<"\n";
            errs() << "Number of predecessors : ";
            errs() << n_predecessor<<"\n";
            errs() << "Count of Successors : "<<n_successor<<"\n"<<"\n";
            number_of_basic_blocks++;
        }
        errs() << "Number of Basic Blocks : "<<number_of_basic_blocks<<"\n";
        return false;
    }
}; // end of Hello pass
} // end of anonymous namespace

```

d) After making the changes to the code, I run “make” command to make changes take place.

The following command is used to take the LLVM .ll file as input and run the Helloworld program.:

```
opt -load ../../Pass/HelloPass/libHelloPass.so -Hello testvi.ll
```

```
ravan@ravan-Precision-5820-Tower:~/Downloads/CS201-F23-Template (1)/CS201-F23-Template/test/phase1$ opt -load ../../Pass/HelloPass/libHelloPass.so -Hello testbc.ll
WARNING: You're attempting to print out a bitcode file.
This is inadvisable as it may cause display problems. If
you REALLY want to taste LLVM bitcode first-hand, you
can force output with the '-f' option.

HelloPass runOnFunction: test
Basic Block : 0
Number of predecessors : 0
Count of Successors :1

Basic Block : 1
Number of predecessors : 2
Count of Successors :2

Basic Block : 2
Number of predecessors : 1
Count of Successors :1

Basic Block : 3
Number of predecessors : 1
Count of Successors :1

Basic Block : 4
Number of predecessors : 2
Count of Successors :1

Basic Block : 5
Number of predecessors : 1
Count of Successors :2

Basic Block : 6
Number of predecessors : 1
Count of Successors :0

Number of Basic Blocks :7
HelloPass runOnFunction: main
Basic Block : 0
Number of predecessors : 0
Count of Successors :0

Number of Basic Blocks :1
```