CS - PART - 1

Step 2b Commands Used:

- **Compile source to binary (executable):**
  clang test.c -o test
- **Compile source to object file (.o):**
  clang -c test.c -o test.o
- **Compile source to machine assembly (.s):**
  clang -S test.c -o test.s
- **Compile source to LLVM bitcode (.bc):**
  clang -emit-llvm -c test.c -o test.bc
- **Compile source to LLVM IR (.ll):**
  clang -emit-llvm -S test.c -o test.ll
- **Convert LLVM IR (.ll) to LLVM bitcode (.bc):**
  llvm-as test.ll -o test.bc
- **Convert LLVM bitcode (.bc) to LLVM IR (.ll):**
  llvm-dis test.bc -o test.ll
- **Convert LLVM IR (.ll) to machine assembly (.s):**

  llc test.ll -o test.s

Step 3:

Code used in Step 3: To modify the HelloPass to print the number of predecessors and successors of each basic block we have used the following code:

This Command Create hellpass.dylib

clang++ -shared -o HelloPass.dylib -fPIC $(llvm-config --cxxflags) hellopass.cpp $(llvm-config --ldflags) $(llvm-config --libs) $(llvm-config --system-libs)

Command: to run Hellopass.dylib on test.ll file:

opt -enable-new-pm=0 -load /Users/hritvikgupta/Downloads/CS201-F23-Template/Pass/HelloPass/HelloPass.dylib -Hello < /Users/hritvikgupta/Downloads/CS201-F23-Template/test/phase1/test.ll > /dev/null

Code Modified in HelloPass.cpp:

```
for (auto &BB : F)
{
  size_t Pred = distance(pred_begin(&BB), pred_end(&BB));
  size_t Succ = distance(succ_begin(&BB), succ_end(&BB));
  errs() << "The Basic block with (name=" << BB.getName() << ") contains "
      << Pred << " predecessor(s) and "
      << Succ << " successor(s).\n";
}
```

Screenshot output