Report of Assignment2

In order to implement a pass to calculate the instruction frequency, we need to do the below things:

1. Build a pass according to the instruction in <https://llvm.org/docs/WritingAnLLVMPass.html>
2. Create a new folder llvm/lib/Transforms/AccessFrqCal
3. Set up CMakeLists.txt by adding

add\_llvm\_loadable\_module( LLVMAccessFrqCal

AccessFrqCal.cpp

PLUGIN\_TOOL

opt

1. Set up lib/Transforms/CMakeLists.txt by adding

add\_subdirectory(AccessFrqCal)

1. Pass is in the file AccessFrqCal.cpp, and test is the file AccessFrqCal.c
2. In the AccessFrqCal.cpp, several important declarations need to be done:

* Declares a “AccessFrqCal” class that is a subclass of [FunctionPass](https://llvm.org/docs/WritingAnLLVMPass.html" \l "writing-an-llvm-pass-functionpass)

**struct** **AccessFrqCal**: **public** FunctionPass

* Declares pass identifier used by LLVM to identify pass

**static** char ID;

**AccessFrqCal** () : FunctionPass(ID) {}

* Declares a [runOnFunction](https://llvm.org/docs/WritingAnLLVMPass.html" \l "writing-an-llvm-pass-runonfunction) method, which overrides an abstract virtual method inherited from [FunctionPass](https://llvm.org/docs/WritingAnLLVMPass.html" \l "writing-an-llvm-pass-functionpass).

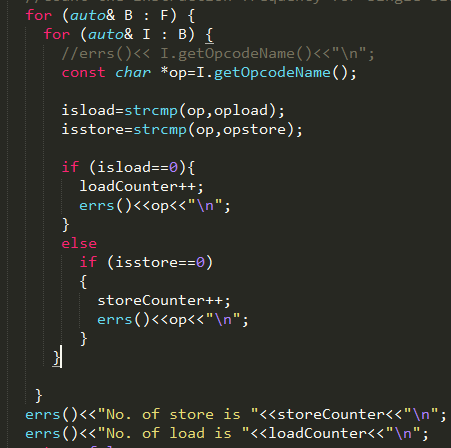
bool runOnFunction(Function &F) **override**

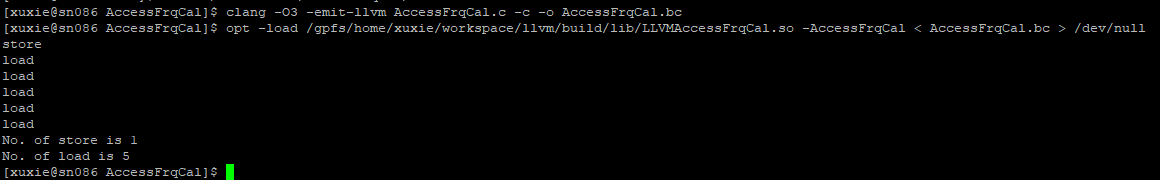
* [register our class](https://llvm.org/docs/WritingAnLLVMPass.html#writing-an-llvm-pass-registration) Hello, giving it a command line argument “**AccessFrqCal**”, and a name “Hello World Pass”

**static** RegisterPass< **AccessFrqCal** > X("**AccessFrqCal** ", "Hello World Pass",false,false);

* compile the file with a simple “gmake” command in llvm/build

1. In runOnFunction(), counter the “store” and “load” instructions’ frequency , and print out the instructions list and total number used.





1. In step2, we only count one time per block, but some blocks may be used more than one time, so we need to get each block frequency by introducing BlockFrequencyInfoWrapperPass through function getAnalysisUsage and recover back by getAnalysis





1. After getting out the block frequency, we should recursive each block in each using time to count the times of load and store respectively, or the Frequency of load/store=load/store each block time\* total block frequency. But I don’t know how to use the frequency which is get out in step3, coz it is a type of BlockFrequencyInfo, so this step is unfinished.