Lab Work – Dominance[1]

Name: _____ ID: _____

One of the key properties of the Static Single Representation is stated below:

> The definition point of a variable v dominates every use of v in the control flow graph of a program.

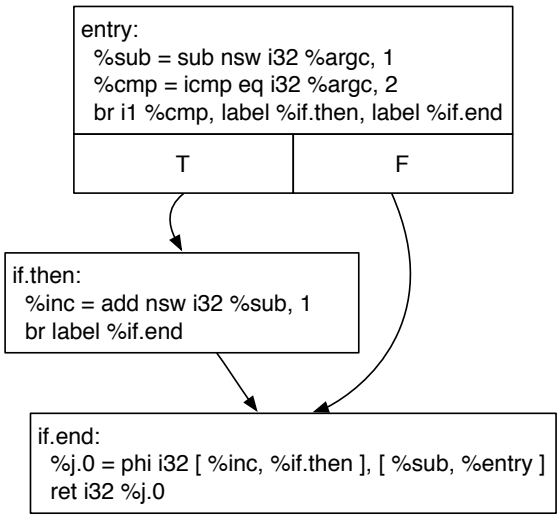
We say that node n in a directed graph with root s dominates another node $m, n \neq m$, if every path from s to m must go through n.

1. Implement a pass that verifies if an instruction i, that defines a variable v dominates every instruction that uses v. Your pass must return true to every program that is produced by `clang` followed by `opt -mem2reg`. As an example, consider the output of your pass for the program below:

```
#include <stdio.h>

int main(int argc, char** argv) {
  int j = argc - 1;
  if (argc == 2) {
    j++;
  }
  return j;
}
```

The CFG of the above program is given on the left, below. Your pass must produce the output on the right:



Expected output for the example:

```
Function main

* Analyzing   %sub = sub nsw i32 %argc, 1
 - use:   %j.0 = phi i32 [ %inc, %if.then ], [ %sub, %entry ] [OK]
 - use:   %inc = add nsw i32 %sub, 1 [OK]

* Analyzing   %cmp = icmp eq i32 %argc, 2
 - use:   br i1 %cmp, label %if.then, label %if.end [OK]

* Analyzing   br i1 %cmp, label %if.then, label %if.end

* Analyzing   %inc = add nsw i32 %sub, 1
 - use:   %j.0 = phi i32 [ %inc, %if.then ], [ %sub, %entry ] [OK]

* Analyzing   br label %if.end

* Analyzing   %j.0 = phi i32 [ %inc, %if.then ], [ %sub, %entry ]
 - use:   ret i32 %j.0 [OK]

* Analyzing   ret i32 %j.0
```

[1]The material necessary for this assignment is available at `http://homepages.dcc.ufmg.br/~fernando/classes/dcc888/lab/exercises/Dominance.tgz`

You will have to use another LLVM pass – `DominatorTree` – to solve this question. This pass is already part of the LLVM distribution, and you should use it[2]. A few bits of C syntax are given below, to help you in this task:

```
#include "llvm/Analysis/Dominators.h"

void getAnalysisUsage(AnalysisUsage &AU) const {
    AU.addRequired<DominatorTree>();
    AU.setPreservesAll();
}

DominatorTree &DT = getAnalysis<DominatorTree>();

DT.dominates ((const Instruction *)Def, (const Instruction *)User);

DT.dominates ((const Instruction *)Def, (const BasicBlock *)BB);
```

2. The notion of dominance for the uses of phi-functions is a bit more elaborate than for the other instructions. If $v = \phi(\ldots, v_1 : b_1, \ldots)$, such that v_1 is alive in the edge that reaches the instruction through basic block b_1, then we say that the definition of v_1 must either dominate b_1, or be the same block as b_1. If this observation is not followed, then, in our initial example, we would have that the instruction `%inc = add nsw i32 %sub, 1` would not dominate the instruction `%j.0 = phi i32 [ %inc, %if.then ], [ %sub, %entry ]`. Explain the tests that you have used to ensure that your pass deals with this definition of dominance.

3. As we have mentioned before, every program that you produce with `clang` will have the dominance property. Implement a program – in bytecode ASCII – that does not have it. Ensure that your pass deliver the correct output to this program.

[2]The class `DominatorTree` is available in `IR/Dominators.cpp`, and you can learn much about LLVM's data-structures by reading that code