Samuel Tapia - 862097969
Abdelrahim Hentabli - 862093221

Project 2: Liveness Analysis

The summary of my implementation of the block-level liveness analysis as follows. In order to compute the LiveOut variables, we must first gather the UEVar and VarKill sets. We started by initializing three `maps<basic_block, sets<string> >`, one for each set of variables UEVar, VarKill, and LiveOut. Also, we created a `map<Value*,string>` for the valueNames at each memory location. We first iterate through each block and compute the UEVar and VarKill variables. Once those sets have been computed with their corresponding block, we are able to compute the LiveOut set. Using the iterative analysis approach, we iterate the CFG and compute the LiveOut of each block until we no longer see a change in the LiveOut set per basic block.

Store & Load

```
else if(inst.getOpcode() == Instruction::Store){
      if(valueNames.find(inst.getOperand(0)) != valueNames.end()){
            UEVar[&basic_block].insert(valueNames[inst.getOperand(0)]);
       }
       VarKill[&basic_block].insert(inst.getOperand(1)->getName().str());
}
else if(inst.getOpcode() == Instruction::Load){
        valueNames[&inst] = inst.getOperand(0)->getName().str();
}
```

The store opcode will check to see if operand 0 exists in the map, if it does not we add it to the map. Additionally, it will also check if the instruction itself is on the map, if it is not we assign an LVN.

The load opcode will set the name of the register that it is loading into to be the same name as the variable that is stored at that memory location. This allows us to find out if the variable is indeed being killed or not.

BinaryOp & ICmp

```
else if (inst.isBinaryOp() || inst.getOpcode() == 52){
      if(VarKill[&basic_block].find(valueNames[inst.getOperand(0)]) ==
                  VarKill[&basic_block].end()){
            UEVar[&basic_block].insert(valueNames[inst.getOperand(0)]);
      }
```

Samuel Tapia - 862097969
Abdelrahim Hentabli - 862093221

```
        if(VarKill[&basic_block].find(valueNames[inst.getOperand(1)]) ==
                      VarKill[&basic_block].end()){
              UEVar[&basic_block].insert(valueNames[inst.getOperand(1)]);
        }
}
```

The binary op and icmp section checks operand 0 and 1, if it does not find it in the VARKill map for the current basic block it inserts the value name into the UEVar map for the current basic block for both of the operands respectively.

Liveness Algo

```
bool cont = true;
set<string> dest1;
set<string> dest2;
set<string> dest3;
while(cont){
       cont = false;
       for(auto& basic_block : F){
           dest3 = LiveOut[&basic_block];
           for (BasicBlock *Succ : successors(&basic_block)) {
               dest1.clear();
               dest2.clear();
               set_difference(LiveOut[Succ].begin(), LiveOut[Succ].end(),
                              VarKill[Succ].begin(), VarKill[Succ].end(),
                               inserter(dest1, dest1.begin()));
               set_union(dest1.begin(), dest1.end(),
                         UEVar[Succ].begin(), UEVar[Succ].end(),
                         inserter(dest2, dest2.begin()));
               set_union(dest2.begin(), dest2.end(),
                         dest3.begin(), dest3.end(),
                         inserter(dest3, dest3.begin()));
           }
           if(!(LiveOut[&basic_block] == dest3)){
              cont = true;
           }
           LiveOut[&basic_block] = dest3;
       }
}
```

Samuel Tapia - 862097969
Abdelrahim Hentabli - 862093221

This section is after creating all the UEVar and VarKill sets for all the basic blocks. We iterate through the CFG though each basic block multiple times until the LiveOut stops changing for all basic blocks. Each time we update each basic block's LiveOut with the LiveOut function:

$$\textbf{LiveOut}(n) = \cup_{x \in succ(n)} (\textbf{LiveOut}(x) - \textbf{VarKill}(x) \cup \textbf{UEVar}(x))$$

When we are complete we print out the UEVAR, VARKILL, and LIVEOUT for each basic block name.

```
for (auto& basic_block : F)
{
       errs() <<"----- "<< basic_block.getName()<< " -----\n";
       errs() <<"UEVAR: ";
       for(auto it : UEVar[&basic_block]){
          errs() << it << " ";
       }
       errs()<<'\n';
       errs() <<"VARKILL: ";
       for(auto it : VarKill[&basic_block]){
          errs() << it << " ";
       }
       errs()<<'\n';
       errs() <<"LIVEOUT: ";
       for(auto it : LiveOut[&basic_block]){
          errs() << it << " ";
       }
       errs()<<'\n';
}
```