Pass-LLVM-Liveness

Writing a pass to find all the Liveness. Project 3 from CS201 UCR.

Setup(From HelloPass)

1. Go to the test folder, and use the command below:

```
./run.sh
```

Code Explanation

1. The implemented Pass extends from `FunctionPass` class and overrides `runOnFunction(Function &F)` function. And it basically finds the UEVAR, VARKILL, LiveOut expression in basic block.
2. In this implement, I used the vector of a set of string to store all the variables. Count is the number of Basic Blocks in the Function.

```
vector<set<string>> uevar(count);
vector<set<string>> varkill(count);
vector<set<string>> liveout(count);
```

3. First I compute the UEVAR and VARKILL by identify the instruction opcode, if it's Load, then it may be in the UEVAR. If it's Store, the operand is in VARKILL.

```
for(auto& inst : *worklist[i]){
  if(inst.getOpcode() == Instruction::Load){
    // errs() << "This is Load \n";
    // errs() << inst << "\n";
    Value* op1 = inst.getOperand(0);
    string temp = op1->getName();
    // errs() << temp << "\n";
    if(varkill[i].find(temp) == varkill[i].end()){
      uevar[i].insert(temp);
    }
  }
  if(inst.getOpcode() == Instruction::Store){
    // errs() << "This is Store\n";
    // errs() << inst << "\n";
    Value* op1 = inst.getOperand(1);
    string temp = op1->getName();
    // errs() << temp << "\n";
    varkill[i].insert(temp);
```

```
  }
}
```

4. Then use the formula to compute the liveout expression.

```
for(BasicBlock *succ : successors(b)){
  auto search = BB.find(succ);
  int i = search->second;

  set<string> dest, dest1;
  set_difference(liveout[i].begin(), liveout[i].end(),
varkill[i].begin(), varkill[i].end(), inserter(dest, dest.begin()));
  set_union(dest.begin(), dest.end(), uevar[i].begin(), uevar[i].end(),
inserter(dest1, dest1.begin()));
  set_union(dest1.begin(), dest1.end(), liveout[id].begin(),
liveout[id].end(), inserter(liveout[id], liveout[id].begin()));
  dest.clear();
  dest1.clear();
}
```

5. Finally output the three expression with ofstream function.

```
ofstream outfile = CreateOut(F);
for(int i = 0; i <= worklist.size()-1; i++){
  errs() << "i" << i << "\n";
  // errs() << "BasicBlock";
  string temp = worklist[i]->getName();
  errs() << temp << "\n";
  outfile << "----- " << temp << " -----\n";
  // worklist[i] -> printAsOperand(errs(), false);
  errs() << "UEVAR" << "\n";
  outfile << "UEVAR: ";
  for(auto it = uevar[i].begin(); it != uevar[i].end(); it++){
    errs() << *it <<"\n";
    outfile << *it << " ";
  }
  errs() << "\n";
  errs() << "VARKILL" << "\n";
  outfile << "\nVARKILL: ";
  for(auto it = varkill[i].begin(); it != varkill[i].end(); it++){
    errs() << *it <<"\n";
    outfile << *it << " ";
  }
  errs() << "Liveout" << "\n";
  outfile << "\nLIVEOUT: ";
  for(auto it = liveout[i].begin(); it != liveout[i].end(); it++){
    errs() << *it <<"\n";
    outfile << *it << " ";
```

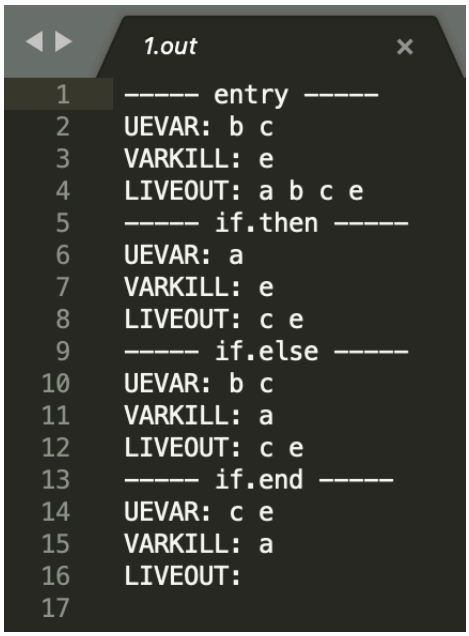
```
  }
  errs() << "\n";
  outfile << "\n";
}

outfile.close();
```

Tests

Run the script in the test folder. We can get two result, one is in the terminal, the other is the output under the same folder.

1. In the terminal, we can see the varkill, uevar and liveout.

2. And in the test folder, there are two output.



2.out

```
----- entry -----
UEVAR:
VARKILL: a c
LIVEOUT: a b d e
----- do.body -----
UEVAR: a b
VARKILL: c
LIVEOUT: b c d e
----- if.then -----
UEVAR: c d
VARKILL: c f
LIVEOUT: b d e
----- if.else -----
UEVAR: d e
VARKILL: a e
LIVEOUT: b d e
----- if.end -----
UEVAR: b
VARKILL: a
LIVEOUT: a b d e
----- do.cond -----
UEVAR: a
VARKILL:
LIVEOUT: a b d e
----- do.end -----
UEVAR: a
VARKILL: a
LIVEOUT:
```