PA5实验报告

计65 钱姿 2015010207

代码实现

- 修改了InferenceGraph.java
  - 修改了`addEdge`函数，增加了图中没有某个节点的判断。
  - 修改了`makeEdges`函数，对需要进行连接的节点之间进行连边
- 修改了GraphColorRegisterAllocator.java
  - 修改了`alloc`函数，首先初始化了*liveUse*，调用*analyzeliveness*获取，随后调用*inferenceGraph*的*alloc*函数
- 修改了Tac.java
  - 增加了`getassigned`函数，获得x∈Def(bb)

问题思考

基本块中两个结点连边的条件

采用PPT中的方法，方法描述如下：

> 如果程序中存在某点，一个结点在该点被定义，而另一个结点在紧靠该定值之后的点是活跃的，则在这两个结点间连一条边

使用*liveOut*、*liveUse*、*liveIn*、*def*来描述，描述如下：

> 设***T***是变量***X***的一个定值点，即***X*∈*Def(bb)***，则对所有***Y*∈*liveOut(T)***且***X*≠*Y***，联结边***(X,Y)***

在此代码框架中，如果想要使用*liveOut*来进行连边的话，需要将所有的*liveUse*加载到继承器中，并且调用*analyzeLiveness*来获取TAC的*liveOut*

实现完整版干涉图染色的寄存器分配算法

PA5框架中实现的简化版干涉图染色寄存器分配，忽略了spilling，未考虑当寄存器放不下将变量压入栈的情况，实现完整版的干涉图染色的寄存器分配算法还需要实现在处理染色过程中找不到临接点个数小于k的结点的情况(k是给定染色数量)

当发生溢出时，需要为被溢出的临时变量分配存储单元，并插入访问这些单元的存/取指令。这些存/取指令访问的是新创建的临时变量（具有很小的活跃范围），在重建了TAC语句之后，需要重新分析每一句TAC的*liveOut*，重新建立图尝试进行染色，直到染色成功。

伪代码如下所示：

```
private boolean color() {
    ...
    else {
            //throw new IllegalArgumentException(
                    //"Coloring with spilling is not yet supported");
        node = random(nodeset);//随机挑选一个node
        this.bb.spill(node,fp);//将需要spill的node传进去
        this.bb.annalyzeLiveness();
        return false;
    }
}
```

```
void spill(Temp node,Temp fp) {
    node.reg = fp;
    node.offset = getNewOffset(fp);
    //遍历所有tac语句，从而将node替换成新变量，并加上load和store语句
    for (tac in basicBlock.taclist) {
        if(tac.hasnode(node))//如果有该变量
            tac.addload();//增加从内存读语句
            tac.replace(node);//替换掉原有变量
            tac.addstroe();//增加写回内存语句
    }
}
```