编译原理PA5实验报告

提前上学2018 李嘉图

2019 年 1 月 6 日

1 任务描述

PA5的任务是，在现有框架的基础上，建立干涉图并实现基于图染色的、不需要考虑spill的寄存器分配算法。

2 实验操作

2.1 干涉图的建立

干涉图的建立分为两步：第一步是将当前基本块的$liveIn$中的变量加载进来，第二步是对于相干的节点进行连边。两个节点连边，表示他们互相干涉，不能使用同一个寄存器。对于任意两个**不同的**结点i, j，他们之间有边，当且仅当存在一条TAC使得：

$$(i \in def \wedge j \in liveOut) \vee (j \in def \wedge i \in liveOut) \tag{1}$$

即一个变量被定值了，另一个变量仍然活跃。活跃的定义是，在基本块后的某个位置，这个变量在被定值之前被使用了。如果这两个变量使用一个寄存器，当一个变量被定值，另一个变量的信息便丢失了。

特别的，对于最开始加入的加载语句，任意两个活跃变量都不能共享同一个寄存器，因此要两两连边。

2.2 函数层面图染色的实现

如果在函数层面实现图染色，主要的方法是一样的。我们求出了每个TAC 的`LiveOut`, `Def`, 依然可以按照上面的策略进行图染色。一个不同

的地方是，在函数开始要将所有参数从栈中取出，将返回值压入栈中。对于对参数和返回值有特殊约定的体系结构，要特殊处理。

2.3 spill的实现

在当前的框架中，设寄存器的个数为k，如果干涉图无法被我们的近似算法k染色，那么程序将无法正确运行。这时需要将一些变量的值存入内存从而空出寄存器。一个简单的策略是，当一个节点的度数比剩余寄存器的个数大的时候，将其spill进内存。当需要访问的时候，将其从内存中load出来。这需要对框架做一些修改：

修改`backend/InferenceGraph.java`中的函数`color`，当找不到一个度数小于等于k的点的时候，将其变为一个溢出的节点。

```
color:
    ...
    if degree[now] > k:
        for tac 'T' such that 'now' is in liveUse(T):
            insert 'load' before T
        for tac 'T' such that 'now' is in def(T):
            insert 'spill' after T
        delete node 'now' from graph
        color again
```

如此不断删除一些点，直到某次染色过程中，干涉图已经可以被k种颜色染色。

而spill和load的实现需要依赖目标机体系结构。一种简单的、无关目标机的实现策略是，预留三个寄存器作为三地址码的三个参数（在最坏情况下，一条指令的三个参数均是溢出的），load是将其载入到对应的寄存器中，spill是将寄存器中的值存入栈中。如果实现这种策略，可以给每个变量加一个'spill'标记。如果一个变量是spill的，从对应的预留寄存器中读入，否则从其绑定的寄存器中读入。