编译原理 PA5实验报告

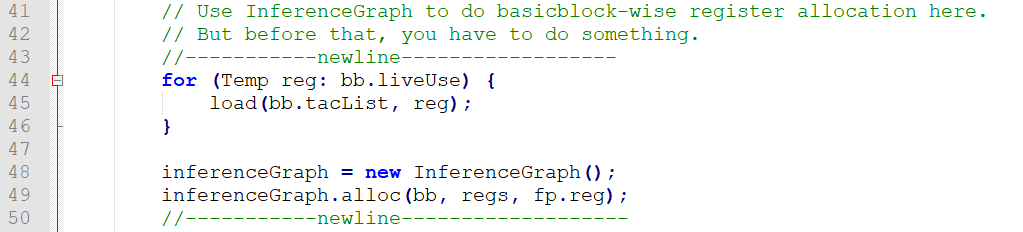
计62 徐晟 2016011253

1. 本次实验要求

本次实验要求使用graph coloring的算法进行寄存器的分配，而我们需要修改对应的GraphColorRegisterAllocator.java和InferenceGraph.java。

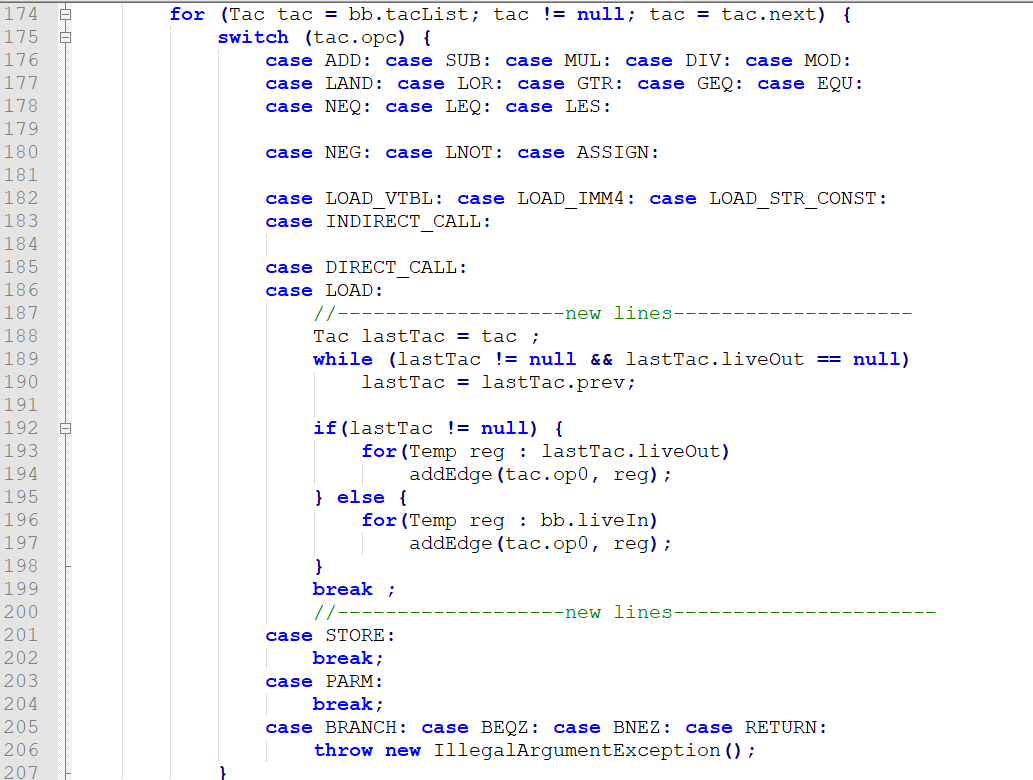
1. 修改的部分

在GraphColorRegisterAllocator.java的修改非常简单，只需要先加入一个inference graph，然后对应的alloc就好了。当然在alloc之前，我们需要把这个basic block的所有liveUse都load进去，添加代码如下：

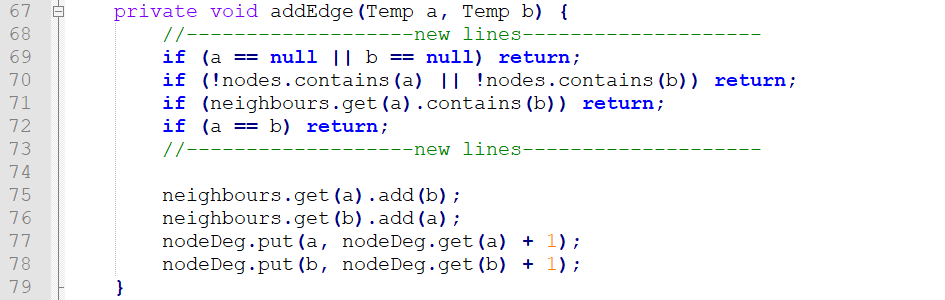


而在InferenceGraph.java中，由于助教写的框架不完善，所以需要进一步的修改，首先是**加边部分**：我一开始尝试把所有的tac.liveOut，bb.liveUse和tac.op0都连边，发现建图之后边数太多。后来考虑到我应该按照类似求解DU链的方式求解寄存器和变量之间的关系，所以我分成了两种情况：首先把tac一直向前推移到最前面，然后按照tac.liveOut是否为空，如果不空则和tac.op0连边，否则用bb.liveUse和tac.op0连边。

具体添加代码如下：



而这并没有结束，因为在建立inference graph的时候，显然会出现重边或者自环，所以在addEdge()函数中还要先进行一次判断：



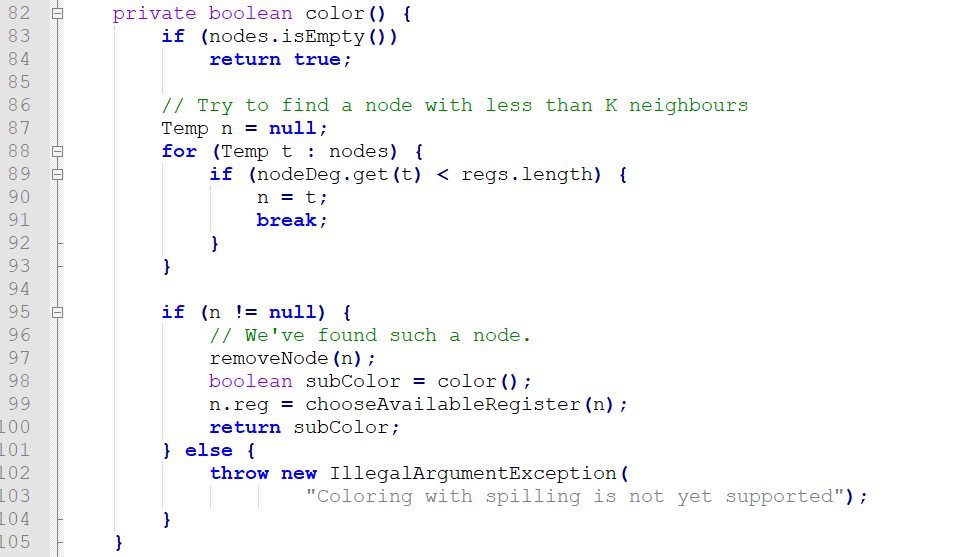
判断node节点是否存在，是否已经加过边了，以及是否是自环。

经过这些修改，我几乎完成了所有测例，但是**t0.decaf依旧不能通过**。

我在MIPS.java中把GraphColorRegisterAllocator替换成了BruteForceRegisterAllocator，获得了同样的result，所以我认为这可能是框架或者测例的问题，而不是我自己修改程序的问题。

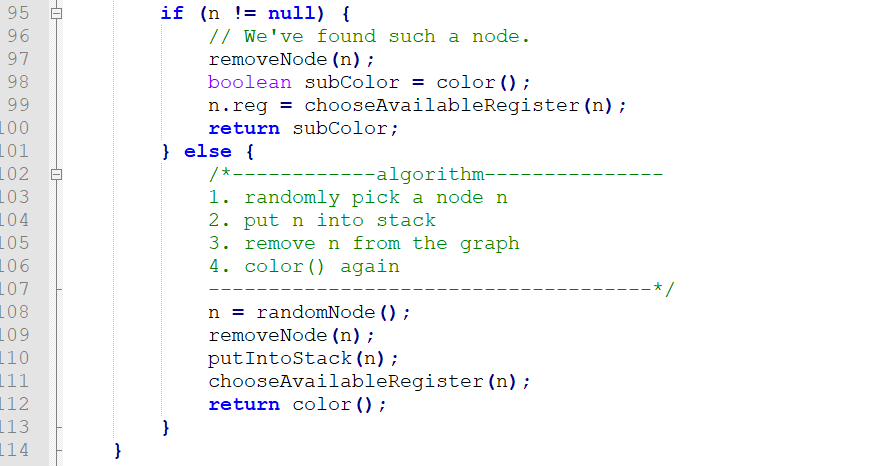
1. 完整的染色算法

完整的染色算法其实就是修改InferenceGraph.java中的color()函数，也就是在没有找到小于度为k的点时进行spilling，也就是随机吧一个节点取出放入系统栈空间中，然后再重新进行color().



这是原本的代码，其中最后一句throw Exception就是因为没有实现spilling算法。

伪代码可以类似的写成这样子：



我个人认为这样就可以了。