Decaf PA5

计64 翁家翌 2016011446

连边的条件

在同一块 BasicBlock 中，liveUse 两两连边（反正只是简易版，最优连法我猜不用这么多条边），liveOut 和 Def 连边。

连边代码如下：

```
// With your definition of inference graphs, build the edges.
void makeEdges() {
    for (Temp i: bb.liveUse)
        for (Temp j: bb.liveUse)
            if (i != j && nodes.contains(i) && nodes.contains(j) &&
!neighbours.get(i).contains(j))
                addEdge(i, j);
    for (Tac tac = bb.tacList; tac != null; tac = tac.next) {
        switch (tac.opc) {
            case ADD: case SUB: case MUL: case DIV: case MOD:
            case LAND: case LOR: case GTR: case GEQ: case EQU:
            case NEQ: case LEQ: case LES: case NEG: case LNOT: case ASSIGN:
            case LOAD_VTBL: case LOAD_IMM4: case LOAD_STR_CONST:
            case INDIRECT_CALL: case DIRECT_CALL: case LOAD:
                if (tac.op0 != null)
                    for (Temp i: tac.liveOut)
                        if (!i.equals(tac.op0) && nodes.contains(i))
                            addEdge(i, tac.op0);
            case PARM: case STORE:
                break;
            case BRANCH: case BEQZ: case BNEZ: case RETURN:
                throw new IllegalArgumentException();
        }
    }
}
```

实现完整干涉图染色寄存器分配算法

通过调小 `Mips.java` 中的寄存器个数，我发现有两个地方一定要改：

1. `InferenceGraph.java` 中的 `color()` 函数，在没有颜色可以染的时候（`n == null`）需要添加代码。只需随便移掉一个节点即可。不失一般性，每次移除第一个，代码如下：

```
else {
    removeNode(nodes.iterator().next());
    color();
    return true;
}
```

2. `GraphColorRegisterAllocator.java` 中的 `findReg` 函数，把 `BruteRegisterAllocator.java` 中 `findReg` 函数的后半段拿进来。具体逻辑是：
    1. `temp.reg == null` 意味着要泄露到内存中，选择一个暂时没什么用的寄存器当备用，不妨记为 `$ta`
    2. `findRegForRead`：把 `$ta` 的值存到内存中，用 `$ta` 把要用的变量给load进来，在结束之后把原先 `$ta` 的值塞回去
    3. `findRegForWrite`：把 `$ta` 的值存到内存中，把执行结果给 `$ta`，存到内存中，在结束之后把原先 `$ta` 的值塞回去

    代码如下（load/spill的最后一个参数表示在bb之前(true)还是之后(false)加这个tac）

```
private void findReg(Tac tac, Temp temp, boolean read) {
    // already in reg
    if (temp.reg != null) {
        temp.reg.var = temp;
        return;
    }

    // find a reg do not need to spill
    for (Register reg : regs) {
        if (reg.var == null || !isAlive(tac, reg.var)) {
            bind(reg, temp);
            if (read) {
                load(tac, temp, true);
            }
            return;
        }
    }

    // find a reg which var's offset already fixed to spill
    for (Register reg : regs) {
        if (reg.var.isOffsetFixed()) {
            spill(tac, reg.var, true);
            bind(reg, temp);
            if (read) {
                load(tac, temp, true);
            } else {
                spill(tac, temp, false);
            }
            load(tac, reg.var, false);
            return;
        }
    }

    // random select a reg to spill
    Register reg = regs[random.nextInt(regs.length)];
```

```
        callingConv.spillToStack(reg.var);
        spill(tac, reg.var, true);
        bind(reg, temp);
        if (read) {
            load(tac, temp, true);
        } else {
            spill(tac, temp, false);
        }
        load(tac, reg.var, false);
    }
```

可是我怎么尝试都没办法跑通过全部测例，在和杜政晓同学讨论之后修改代码还是不行。然而，我使用 `BruteRegisterAllocator` 在寄存器个数很少的时候也是会挂，然后就不知道该怎么办了……（我觉得我对于框架理解还不够深入……或者还有其他地方要改？