HW8 report

xht03

2025-05-22 22:28:10

二、算法原理

图着色寄存器分配算法通过构建干涉图（Interference Graph），将寄存器分配问题转化为图着色问题。在干涉图中：

- 每个临时变量作为一个节点
- 如果两个变量在某个程序点同时活跃，则它们之间有一条边
- 图的着色使用固定数量（k）的颜色，对应可用的物理寄存器

算法的主要流程包括：

1. 构建干涉图，识别移动相关节点
2. **简化**：移除低度数（<k）非移动相关节点
3. **合并**：合并移动指令对应的非冲突变量
4. **冻结**：将移动指令冻结，使低度数移动相关节点可以被简化
5. **溢出**：当其他策略无法继续时，选择高度数节点进行潜在溢出
6. **选择**：为简化后的节点分配实际的寄存器（颜色）

三、实现细节

3.1 简化（Simplify）

简化操作是算法的第一步，目标是移除干涉图中度数小于 k（可用寄存器数量）的非移动相关节点。

关键点：

- 寻找度数小于 k 的非移动相关节点
- 将找到的节点加入简化栈
- 从图中移除该节点及其关联的边
- 每次只简化一个节点，因为简化会改变图的结构

3.2 合并（Coalesce）

合并操作处理移动指令相关的变量对，如果可能，将它们合并为同一个节点。

关键点：

- 使用 Briggs 保守合并策略，确保合并后不会导致更难着色
- 处理机器寄存器的特殊情况，确保预着色节点的约束
- 维护合并信息，以便在 select 阶段使用

3.3 冻结（Freeze）

冻结操作用于处理不能合并的移动指令，使低度数的移动相关节点可以在下一轮中被简化。

关键点：

- 简单地移除一个移动指令对，使相关节点不再被视为移动相关
- 这些节点在下一轮中将可以被简化处理

3.4 溢出（Spill）

溢出操作在其他策略无法继续时使用，选择高度数节点作为潜在溢出候选。

关键点：- 选择度数最高的节点作为溢出候选

- 只是软溢出，真正决定是否溢出发生在 select 阶段
- 从图中暂时移除节点，加入简化栈

3.5 选择（Select）

选择是最后一个阶段，为简化栈中的节点分配实际的寄存器（颜色）。

关键点：- 首先为机器寄存器预分配固定颜色

- 从简化栈顶开始，逆序处理节点
- 考虑邻居已使用的颜色，寻找可用颜色
- 如果无可用颜色，标记为溢出
- 最后验证着色结果