HW10 实验报告——Quad SSA 优化

胥昊天

2025-06-05 23:30:00

实现思路

1. **条件常量传播**
   - 为每个临时变量分配一个运行时值（RtValue），其状态为 NO_VALUE、ONE_VALUE 或 MANY_VALUES。
   - 通过遍历基本块和语句，传播常量信息，并标记哪些基本块可达。
   - 采用迭代直到收敛（固定点），每次有新信息传播时重新遍历。
2. **无用代码和不可达代码消除**
   - 替换所有能确定为常量的临时变量为常量。
   - 删除所有不可达的基本块和已确定为常量赋值的语句。
   - 对条件跳转，如果条件恒真/恒假，直接替换为无条件跳转。

三、代码实现

1. `Opt::calculateBT()` 的实现

- 首先将入口基本块标记为可执行。
- 遍历所有基本块和语句，根据语句类型进行如下处理：
  - **LABEL**：如果该块只有一个出口且可达，则递归标记后继块可达。
  - **CJUMP**：如果左右操作数均为常量，根据 relop 判断跳转方向，只标记可达分支；否则两分支都标记为可达。
  - **MOVE/MOVE_BINOP**：如果右侧为常量或单值变量，则传播常量；否则传播 MANY_VALUES。
  - **PHI**：如果所有前驱块传入的值一致且可达，则传播该常量，否则传播 MANY_VALUES。
  - **LOAD/CALL/EXTCALL**：结果均为 MANY_VALUES。
- 每次有新信息传播时，重新从头遍历，直到所有信息收敛。

2. `Opt::modifyFunc()` 的实现

- 遍历所有基本块和语句，进行如下优化：
  - 删除所有不可达的基本块。
  - 删除所有目标为单值的 MOVE/MOVE_BINOP/PHI 语句。
  - 对 PHI，如果某个参数为单值，则在前驱块插入常量赋值，并替换 PHI 的参数。

  - 替换所有使用到的单值变量为常量。
  - 对 CJUMP，如果左右均为常量，则直接替换为 Jump，并更新出口标签。
- 最后更新函数的临时变量和标签计数。

3. 关键数据结构

- `block_executable`：记录每个基本块是否可达。
- `temp_value`：记录每个临时变量的运行时值（RtValue）。
- `label2block`：辅助映射标签号到基本块对象。

四、常见问题

- **Temp_map 构造函数参数不匹配**
  由于 `Temp_map` 只提供了无参构造函数，不能传递参数初始化，直接用无参构造即可。
- **Phi 传播的正确性**
  需要确保只有所有可达前驱传入的值一致时，才能将 PHI 结果视为常量，否则应为 MANY_VALUES。
- **goto + 迭代收敛**
  为了保证每次有新信息传播时能及时重新遍历，采用了 `goto start` 的方式实现高效的固定点迭代。

参考资料： -《虎书》Modern Compiler Implementation in C/Java/ML, Chapter 19.3