HW6 Report

胥昊天

2025-04-24 21:29:40

git graph

```
(base) keats@OMEN-Yanxu:~/FudanCompilerH2025$ git log --graph
* commit 62945bd2e0f8ef486158789e634a05a2da601182 (HEAD -> main, origin/main)
| Author: xht03 <1620318777@qq.com>
| Date:   Thu Apr 24 21:26:29 2025 +0800
|
|     HW6 done.
|
* commit a9404a4c40c6e262dd1fe5f4b28b4e6ea1920b61
| Author: xht03 <1620318777@qq.com>
| Date:   Thu Apr 24 20:27:53 2025 +0800
|
|     finish writing.
|
* commit ca73a0b51ae6c76134cbc89c48c17438be18d809
| Author: xht03 <1620318777@qq.com>
| Date:   Thu Apr 24 18:52:42 2025 +0800
|
|     backup halfway.
|
* commit b1785c626d1f58714d19465256fe91b18ba97a08
| Author: xht03 <1620318777@qq.com>
| Date:   Thu Apr 24 15:11:22 2025 +0800
|
|     pull HW 6 and 7 from gitee
|
```

Figure 1: git log

设计思路

转换策略

采用模式匹配的方式，将 IR 树中的各种模式转换为对应的四元式。主要支持以下模式：

- Move: `temp <- temp`
- Load: `temp <- mem(temp)`

- Store: `mem(temp) <- temp`
- MoveBinop: `temp <- temp op temp`
- Call: `ExpStm(call)`
- ExtCall: `ExpStm(extcall)`
- MoveCall: `temp <- call`
- MoveExtCall: `temp <- extcall`
- Label: `label`
- Jump: `jump label`
- CJump: `cjump relop temp, temp, label, label`
- Phi: `temp <- list of {temp, label}`

核心数据结构

- `visit_result`: 存储转换后的四元式列表
- `output_term`: 指向表达式计算产生的临时变量
- `quad_block`: 当前正在处理的四元式基本块
- `quad_func`: 当前正在处理的四元式函数
- `quad_prog`: 整个四元式程序
- `temp_map`: 临时变量映射，用于生成新的临时变量和标签

代码实现

详见代码仓库

实现难点与解决方案

1. **表达式求值与临时结果存储**

   难点：树形 IR 中表达式求值是自底向上的，需要正确传递中间结果。

   解决方案：使用 `output_term` 字段存储表达式的计算结果，确保子表达式的结果能够正确地传递给父表达式。

2. **变量定义与使用的跟踪**

   难点：需要准确地跟踪每条四元式定义和使用的变量，这对后续的数据流分析至关重要。

   解决方案：为每个四元式维护 `def` 和 `use` 集合，在遍历过程中动态更新。

3. **不同类型节点的模式匹配**

   难点：IR 树中某些节点可能对应多种四元式模式，需要根据上下文进行判断。

   解决方案：使用条件语句检查节点类型和子节点特征，选择合适的四元式表示。

实验心得

这次实验是编译器构建过程中的重要一环，为后续的代码生成和优化奠定了基础。通过这次实验，我不仅学习了四元式的生成技术，也加深了对编译原理整体架构的理解。