中山大學

SUN YAT-SEN UNIVERSITY

编译原理
Project 3&4

(2022 - 2023 学年春季学期)

课程名称	编译原理
学院	计算机学院
授课教师	张献伟
学生姓名	唐喆
学号	20337111
专业	计算机科学与技术
Email	tangzh_33@163.com
完成时间	2023 年 07 月 03 日

School of Computer Science and Engineering, Sun Yat-sen University

编译原理 Project 3&4 Generator & Optimizer

20337111 唐喆

2023 年 07 月 03 日

目录

1 概述与感想

1.1 概述

本次实验需在SYsU-lang实现的代码框架下，基于 LLVM 提供的接口完善编译器的语法分析器（Sysu-Generator）。

相较于实验二，实验三难度更加进阶，代码量和学习曲线更陡峭，需要学习的知识点更多。直观来说，代码量都呈几何倍数增加。在断断续续历时两周、踩了无数坑、重构了无数次后，本次实验我的提交代码累计 3000 行 +，数十次的 Commit 着实记录了曲折的 Debug 过程：

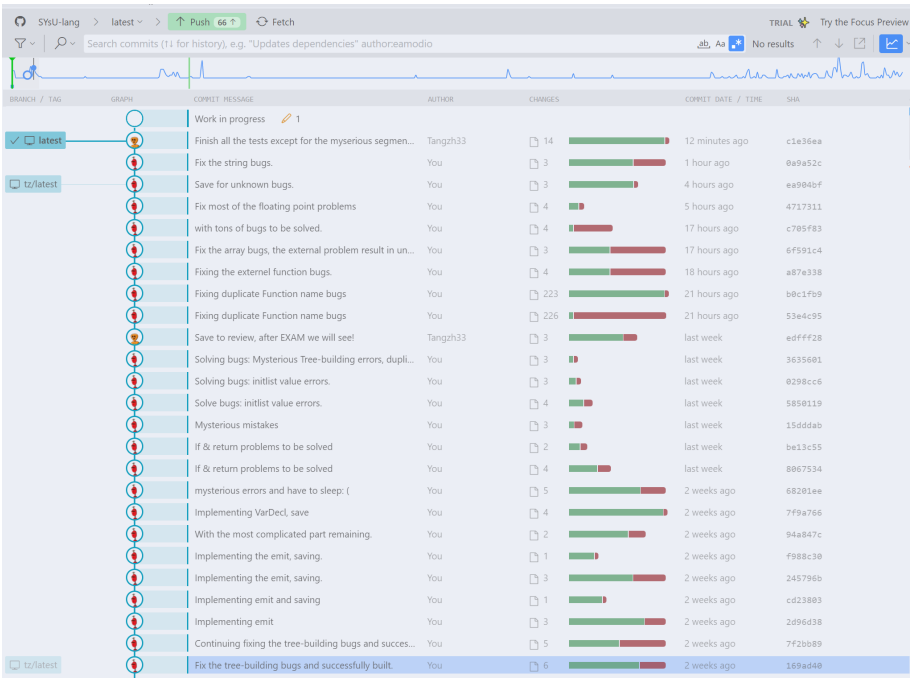

图 1: 实验三 Commit 记录

以及我超过 3000 行的代码量实现：

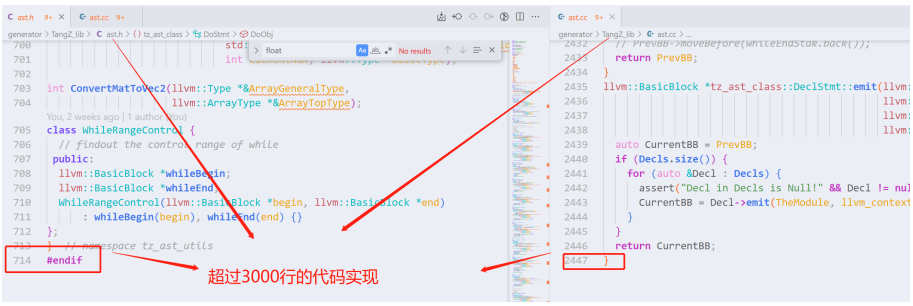

图 2: 实验三代码量统计

1.2 感想

在踩了这么多坑后，细节都记录在代码中了，但完成项目知道思想比细节更值得记录。以下是本次实验我的一些感想与个人技术总结。

因为个人原因，我相较大多同学很晚才开始做，因此有所谓的“后发优势”：在和同学们的讨论中，他们总是抱怨一边读入 JSON 一边发射 IR 会使得代码**超强耦合**，在添加新功能时会非常难以修改。因此，我在实现时给了自己一个规划，即：

- 先用数据结构在内存中建树，再处理树中内容维护好变量作用域，最后通过遍历树发射 IR。
- 学习顾宇浩学长的思路，用 C++ 的多态和继承建立严谨的树结构，方便维护；
- 使用虚函数 + 重载来发射 IR，这样我们就可以利用多态：同一个父类指针，但是不同的子类实现，从而统一的发射不同类型的 IR。

以下是我设计的其中一个节点：

```
class VarDecl : public Decl {
 public:
  bool isInited;
  bool isConst;
  // By default, the var is not global.
  bool isGlobal;
  Expr *InitExpr;
  VarDecl() = default;
  VarDecl(std::string name, llvm::Type *type, bool isConst, bool isInited,
          bool isGlobal, Expr *InitExpr)
      : Decl(std::move(name), type),
        isConst(isConst),
        isInited(isInited),
        isGlobal(isGlobal),
        InitExpr(InitExpr) {}
  VarDecl(llvm::LLVMContext &llvm_context, const llvm::json::Object *json_tree,
          const bool _isGlobal = false);
  llvm::BasicBlock *emit(llvm::Module &TheModule,
                         llvm::LLVMContext &llvm_context,
                         llvm::BasicBlock *PrevBB,
                         llvm::Value **ReturnValue) override;
  void print() override {
    printf(
        "VarDecl, Name: %s, isConst: %d, isInited: %d, isGlobal: %d, "
```

```
      "InitExpr: ",
      name.c_str(), isConst, isInited, isGlobal);
    if (isInited) {
      InitExpr->print();
    }
    printf("Finish VarDecl\n");
  };
};
```

尽管架构是先进的，但是实现是非常困难的。虽然其他同学的代码强耦合，但是由于去年蛮多学长都是这么实现，大家总是能找到请教和模仿的对象；但我的架构迥然不同，我需要大量的摸索，要求我在实现时解决许多独特的挑战，例如：

- 如何设计树的继承关系，使其拥有合理的虚函数-重载接口，方便发射 IR?
- 如何设计多态函数接口，使其足够兼收并蓄，所有的结构体都能访问、所有类型的 IR 能共享一套接口来发射?
- 如何静态实现全局常量的编译时计算，使其不用等待代码运行时就可以得出结果?
- 如何在 JSON 建树时就处理好变量作用域，使重名变量不会影响到发射 IR?

很骄傲的说，以上的问题我都给出了解决方案，并且这些方案均通过了测例的验证：如我使用了 JSON 中独特的 ID 号，加上对应变量的符号名，这样就可以实现一个唯一的变量索引等；同时虚函数 + 指针能写出功能非常强大的多态特性，使得我们发射 IR 时不用专门分辨指针类型，而是通过 C++ 编译器自动找到对应的 IR 发射函数。

除了特别的代码架构外，这次实验还有几个值得一提的经验记录：

- **抽象通用操作**：总结多种通用操作，如解析二元操作符的类型转换、强制类型转换等，这样可以大大减少代码量，提高代码的可读性和可维护性；否则常常顾头不顾尾，修改一处忘了另一处，常常在 debug 的路上；
- **拥抱 C++ 高级特性**：其实 C++ 已经非常强大，拥有 **dynamic_cast** 等强大的多态支持能力，我们可以利用这些特性，写出更加优雅的代码；
- **善用调试工具**：相较于 Python 和 Java 等自带虚拟机的语言，C++ 比较痛苦的点在于发生段错误时没有调用栈，从而很难定位问题。我使用的方法是 **Valgrind + GDB** 联合调试，方便打印调用栈。

以下是我的一个 GDB debug 脚本参考（支持 LLVM 编译、GDB 调试）：

```
file ~/sysu/bin/sysu-generator
b main
b tz_ast_class::ArraySubscriptExpr::emit
# b tz_ast_class::InitListExpr::emit
```

```
# b tz_ast_class::DeclRefExpr::emit
# b  tz_ast_class::ParmVarDecl::ParmVarDecl
# b tz_ast_class::FunctionDecl::FunctionDecl
# b tz_ast_class::FunctionDecl::FunctionDecl
# b tz_ast_class::BinaryExpr::emit
# b tz_ast_utils::RaiseOperandType
# b tz_ast_class::IntegerLiteral::IntegerLiteral
# b tz_ast_class::VarDecl::emit
b tz_ast_class::DoStmt::emit
# b ast.cc:240
# b ast.cc:855
# 指定输入文件
run < tmp/clangout.json
layout src
```

2 实验结果

在反复调试修改后，基于以上设计的 Generator 实现通过了自动评测机，成功上榜。除去因超时而跳过的测例外，我的实现共通过 634 个测例，性能分是 0.17088773367967985。评测机报告截图如下：

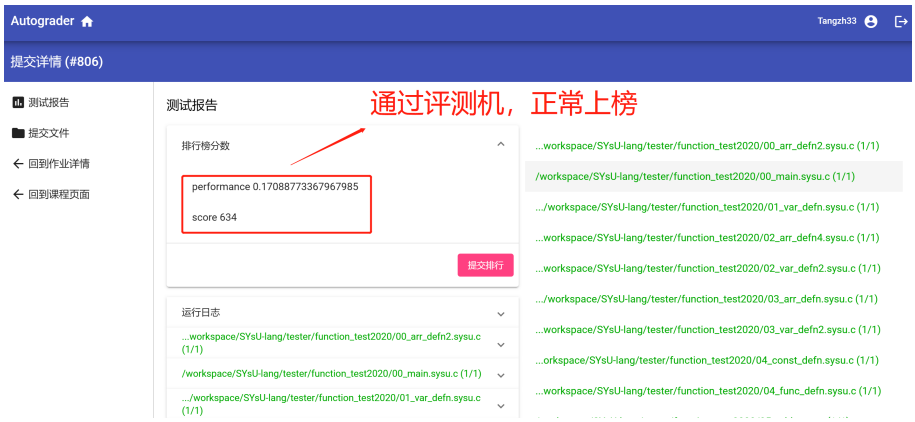

图 3: 实验三实现完整截图

值得指出的是，本次实验有几个较为麻烦的测例，我也截图如下。首先是**浮点测例**（相对路径：*tester/function_test2022/95_float.sysu.c*）：

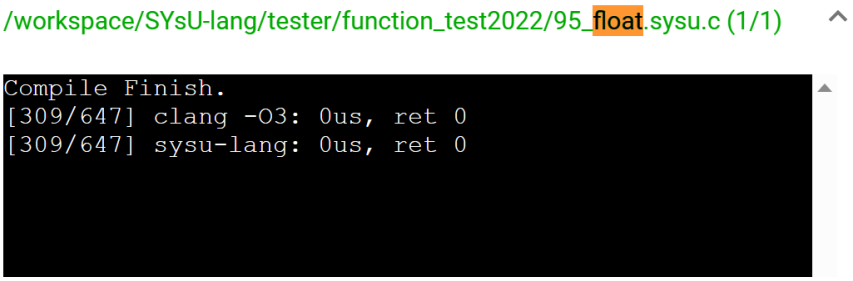

图 4: 实验三浮点测例截图

其次是短路判断测例（即逻辑判断中的短路实现，相对路径：*tester/function_test2022/51_short_circuit3.sysu.c*）：

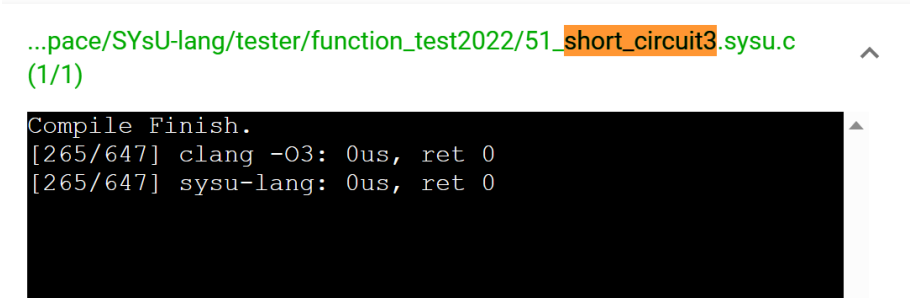

图 5: 实验三整数上限测例

以及最后的字符串拼接测例（相对路径：*tester/mizuno_ai/mizuno_ai.sysu.c*）：

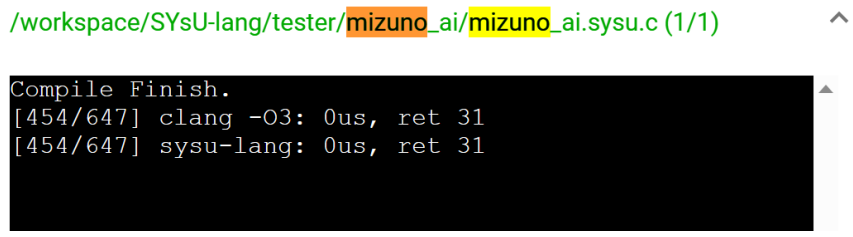

图 6: 实验三字符串拼接测例

最后，由于时间关系，我没有机会继续完成 Lab4 所要求的优化，目前性能分是 0.171。对此我感到非常遗憾与难过，希望在暑假我能有机会深入研究、继续完善。

3 实验感想

非常感谢张老师和 TA 们的理解和宽容，让我有机会在 DDL 后继续完成实验。由于超算比赛的蝴蝶效应，备赛 + 参赛耽误了太多事情，导致我一直在拆东墙补西墙的路上；本次作业基本上都是在期末周中挤时间完成，多次凌晨 debug 的经历属实难忘。但是在完成的那一瞬间，我还是非常骄傲的，骄傲于自己完整设计、掌控了一个超千行的大实验，骄傲于自己的代码能够通过评测机，骄傲于自己的代码能够按要求实现了实验三的所有功能。感谢老师和助教们的辛勤付出，让我能够在这个学期真正从大工程视角了解编译原理，也希望我的小小感想能让这个实验变的更好。