一些分享

大家好，我是17级计算机学院的学生赵博，在这里想要跟大家分享一些我在编译器设计与实现中的一些经验与思考。在去年的课设竞速排名中，我取得了第六名；在今年暑假举办的全国大学生计算机系统能力大赛编译竞赛中，我所在的队伍取得了一等奖，第三名。也算是有一些经验，踩过一些雷。

我主要从下面几个方面谈一谈：编译器设计、中间代码、目标代码。

一、编译器设计

在编译器设计的要求上，实验课的要求是使用C/C++语言实现一个小型编译器。在理论课中我们学过，编译器设计要考虑的因素有很多，典型的有：正确性、具体架构、编译速度、生成代码的性能等。在实验课中，对架构与编译速度几乎没有什么要求。所以会有一些同学设计出几千行的单文件编译器，也有同学设计出代码层次清楚、耦合度低的编译器；有同学会设计出单遍的编译器，也有同学会设计出多遍的编译器。对比而言，我更推荐**多遍+解耦**的方式。代码生成与优化是非常复杂的工作，如果使用if-else类似判断强耦合在源代码中，会给后续的优化与纠错排查带来很多困难。

二、中间代码

中间代码对于编译器是非常重要的，很多的优化会在中间代码层面上展开。形式上，实验课对中间代码的要求是自行设计的四元式。在很多时候也足以满足优化的需要。学有余力的同学可以去了解其他形式的中间代码，比较他们的异同。

在中间代码上可实现的优化有很多，典型的有：公共子表达式删除、死代码删除、函数内联、常量传播等。这些都是在理论课程中学过的优化方法。然而，如何**把优化思想应用到更加复杂的场景**（函数调用、数组操作等），这是大家需要去思考的。大家可以参考其他学长的一些优化文档，思考他们的做法是否合理，有没有问题。等到有一个清晰的思路再动手实现，而不是草率的开始。

在优化的架构上，我更推荐大家使用**独立的若干pass**实现优化(LLVM 的实现方法)。简单而言，是将各个优化实现为独立的优化器，优化器的输入与输出均是中间代码，他们彼此之间没有直接联系。这样可以在最大程度上解耦，也更方便后续的优化实现与纠错。

三、目标代码

在目标代码中，最重要的就是寄存器的分配。在全局寄存器的分配上，常见的分配方式有**引用计数、线性扫描、图着色**等。在临时寄存器的分配上，常用的有**临时寄存器池、线性扫描**等。这些分配算法大家可以自行选择实现。除此之外，还可以实现一些窥孔优化删除掉特定的冗余，这就要同学们根据自己的编译器去设计实现。

另外，在寄存器的使用上，我建议大家遵循寄存器使用与保存的规范。

四、一些建议

1. **C++ & git & 多个Pass** 可以为你省去很多被C、混乱版本、耦合度高这些问题浪费的时间。
2. 把目标聚焦在更加**通用型的优化**上，优化的质量比数量更加重要。
3. 在实现优化前认真思考。优化的实现是很容易出错的。让优化实现的更为健壮，需要你在理论课的思想上进一步思考。
4. 在中间代码与目标代码中，输出一些**用以调试的日志信息**，可以帮助你更高效地定位问题所在。
5. **合理地投入时间。**优化是没有止境的，但是时间是有限的。我不建议大家在优化中投入过多时间而耽误其他课程的复习，也不建议大家敷衍了事。希望大家在优化实现过程中有所收获。