学习总结

本学期经过几个月的辛苦努力，从词法分析器开始一步步完成了一个复杂度高，完整的编译系统。从样例构造，词法分析，语法分析，错误处理，到中间代码生成，目标代码生成，代码优化，一步一个脚印，每一次都是一次对自己的挑战和提升。前三次作业难度适中，由于OO课打下了较好的基础，用Java完成词法分析和语法分析都相对轻松。但是debug的过程也比较痛苦，主要涉及对多递归程序的调试。这个过程提升了一定的debug能力，为之后的编码过程打下了坚实基础。同时，生成语法树方便后续的代码生成和语义分析也是在和同学们的交流中知道并设计好，这一点为之后的编译器的顺利完成提供了可能。之后遇到了第一个难度较大的部分就是错误处理，这一部分需要用一个`Visitor`对先前生成的语法树进行遍历，并分析课程组要求的相关错误。在反复思考后选择了后序遍历+局部深度搜索的方式进行处理。至此第一阶段结束。

第二阶段的任务是代码生成，在思考后觉得生成`mips`能提高自己的核心能力，因此选择了`MIPS`部分，中间代码生成走了很多弯路。开始的时候我对中间代码理解出现偏差，首先仿照LLVM设计了一些中间代码，但是生成的时候过于重视临时变量和寄存器。因此束手束脚，生成了一种介于中间代码和目标代码之间形式的二不像代码。最后在课程组提供了中间代码参考样式之后，经过和舍友的讨论，着手重构了中间代码设计。分成了十几个中间代码类，并继承一个`IRCode`父类进行统一管理。同时放弃了书本上的语法制导翻译，直接递归下降遍历的方式进行中间代码生成。由于架构适当，这一部分重构的速度很快，仅仅一天就搞定了。之后便是`MIPS`代码生成。一开始的时候由于想快速通过代码生成作业，选择对前端的中间代码直接生成目标代码，没有合理地组织成函数-流图-基本块的形式。同时后端也遇到了很多bug，经过接近四天的debug，最终终于调好了后端。并完成了基本的乘除优化。但发现优化之后不能通过代码生成作业。因此只能关掉优化，提交了普通版本。之后很长时间由于忙碌于其他科目的事情，没有投入精力到优化中。直到竞速截至前的两周才又开始优化，先对前端的代码进行了整理，组织成了合适的架构，又反复的调试，将活跃变量分析等数据流分析写好，又写好了循环代码优化。之后坚定的重写了一套后端，有了最终的根据基本块进行代码生成的版本。整个过程不能说是一番风顺，还是遇到了很多困难，但是这个过程也夯实了理论基础，动手实现数据流分析和一些优化也让我体验了如何实现一个现代的编译器。整体来说还是收获颇丰

总的来说，就是，**想，都是问题，做，才是答案。**编译器的完成让我又一次体会到了实践出真知这句话的魅力。只有不断实践，不断努力，才能夯实理论基础，提升自己的编程水平与理论实践水平。

同时，理论到实际的变现有时需要付出很多额外的精力，加入很多必须的限制约束。比如优化部分的Deg图和死代码删除，很多时候想真的删除某个代码需要满足很多的约束条件。

此外，编码之初合理的架构设计与目标明确，对于避免重构和提高编码效率有很大的正向作用。只有良好的架构设计才能方便后续的代码优化和迭代。总的来说还是高内聚，低耦合。

课程评价与建议

整体来讲编译课程设计还是很合理的，每次作业循序渐进，让我们有充分的动手实践的机会。然而以下几点还是整个学习过程中遇到的一些问题。

1. **部分章节理论课与实验脱节。**比如中间代码设计与生成部分。书本上讲的是语法制导翻译，但是并没有给出推荐的方法告诉我们自己如何生成代码。优化部分，到达定义分析和活跃变量分析只详细讲了特定情况下如何求，并没有讲适合我们实验文法的代码如何分析。同时没有详细讲数据流分析之后如何删除冗余代码。以及很多优化部分内容都只是浅尝辄止，实际用的时候发现书到用时方恨少。查阅大量网上资料花费了不少时间。之后可以考虑给出一些代码生成的例子或者详细一些的引导，降低大家的学习成本。
2. 课程组**没有给出推荐的`Java`或者`C++`编译器代码架构**，全凭大家自己设计，虽然提升了我们的创新设计能力，但是也丧失了向先进的现代编译器学习和模仿的机会，很多同学的实现方法比较拟合化，只对特定的数据有效。之后可以考虑提供一些优秀的小型编译器（不是那种pascel版本的）代码让大家阅读，最好是C++或者Java的代码。

3. **优化部分的时间尽量适当延长，不要和理论考试挤在一起**，很多同学的13-16周各个一般专业课事情很多，并没有充足的时间进行合理设计和优化。

致谢

感谢我的舍友文思勇肖伟强，我们三个人不断地互助学习，不断攻克每一个难关。

感谢吴佬和林佬那么多次的无私分享，以及很多hxd的帮助。