编译实验总结感想（21373450）柳浩东

学习收获

理论知识掌握：通过编译课程的学习，我初步了解了高级语言和LLVM，MIPS之间的对应关系。了解了DFA,NFA，自顶向下，自底向上分析等文法分析方法。掌握了死代码删除，循环不变式外提，到达定义分析等一些代码优化的方法。体会到了好的架构和差的架构给后续推进带来的便利和阻碍。

通过**词法分析**、**语法分析**的学习，我深入理解了编译器是如何把源代码的一个个字符转换成可识别的Token序列。再到具有语义的语法树。理解了高级语言在计算机视角下的样子。对语言这个词的含义有了更加深刻的认识。

通过**符号表管理**和**错误处理程序**的使用，我理解了计算机是如何一点一点的检查出用户输入的错误。原本高深莫测的编译器架构在此刻也变得逐渐清晰。

通过**语义分析**和**生成LLVM目标代码**，我真正做到了把一种语言翻译成另一种语言，意识到了翻译过程中必然带来许多冗余，也掌握了一些消除冗余的手段。在调试LLVM代码的过程中学习了Ubuntu系统的简单使用。

实践能力提升： 在这个项目之前，我编写过的最长的代码也就一千多行。虽然我的编译实验最后只生成到LLVM，但总的代码量也达到了六千行。其中也运用了许多之前从未用过的JAVA语句。

自主学习能力：独自构建一个功能完善的编译器是迄今为止我遇到的最困难的任务，为此我不得不一边参考往届的编译器作业，一边阅读龙书，虎书这些外国著作，CSDN更是翻阅了无数次。LLVM的全英文指令集介绍也大致浏览了一遍。这些都对我自主学习的能力的提升起了很大的帮助。

总的来说，通过这门课程，我不仅学到了丰富的知识，更强化了分析问题和解决问题的能力，这都将对我的未来学术和职业发展产生帮助。

对课程的建议

错误处理部分和后面的代码生成部分结合的作业里，出现了一些理论上在错误处理作业里不应该出现的错误。希望以后能够统一起来。

目标代码生成的MIPS教程过于简陋，相比之下LLVM的教程就显得十分细致，这也是我最后放弃生成MIPS的原因。相比于LLVM有着Ubuntu系统下的功能完善的报错评测，以及根据c语言生成正确的LLVM样例。生成目标代码为MIPS的路线中教程给的样例数量不足，而且Ubuntu把c语言编译出的MIPS代码又无法在Mars上运行，这就导致了学生很多时候只能猜测着构造目标代码，无法从一段一段标准而简洁的MIPS代码中总结经验。建议能够模仿LLVM的教程，给出多样的MIPS样例。虽然更加详细的教程必然带来更加高的平均分，更加卷的结果。但我认为与其比谁在黑暗中能够坚持摸索的更远，不如让大家都在光明下共同进步。

竞速排序的任务只有MIPS可以完成，但是理论上LLVM的代码本身也有优化的空间，课程组也给出了LLVM的优化教程。我认为直接限制生成其他代码同学的优化操作，不让他们发挥理论课上学到的知识，是有失公平的，虽然生成MIPS的代码优化任务显然更难，但是也可以在降低适当分值的基础上，设置LLVM和PCODE的代码优化的分数。

建议课程组公开代码优化最后的评分细则，比如前百分之几对应多少分，极端数的出现是否会影响大家的得分等等。排名带来的不确定性一方面打击了同学选择代码优化的积极性，一方面也容易导致一些能力不够的同学选择竞速排序，最后得不偿失。