**项目总结文档**

1. 成员组成

蒲超东 2016302580031

卢玉田 2016302580011

喻为2016302580034

陈昱帆 2016302580038

1. 分工

蒲超东：

IF和WHILE控制流程语句实现、变量声明以及赋值的实现、目标代码生成

卢玉田：

部分AST及parser的实现、IR代码生成、整体解析及IR生成实现、JIT实现、优化器支持

喻为：

部分词法分析器实现、部分AST及parser的实现、其余部分的辅助实现

陈昱帆：

部分词法分析器实现、部分AST及parser的实现、其余部分的辅助实现

1. 个人总结

蒲超东：

本次实验中，我主要负责了后面IF条件控制流程及WHILE循环控制流程的实现、变量声明及赋值的实现以及最后生成目标代码的实现。其中IF语句及WHILE语句的实现的过程当然也牵扯到了前面的相应语法的解析以及IR代码的生成，需要定义新的抽象语法树节点以及对应的解析器，而因为所要实现的VSL语言的语法与llvm官方文档中的实现有很大的不同，所以实现过程中也遇到了不少问题，查阅各种资料，最终得以实现。变量的声明及赋值、目标代码生成的实现也参考llvm官方文档稍作修改实现。

卢玉田：

首先参考官方文档结合实验手册中VSL语言的语法，实现部分抽象语法树（AST）的构造以及语法的解析。对每个AST节点参考官方文档中IR代码生成的部分构造各自相应的codegen方法以实现各节点到IR代码的转变。通过LLVM框架实现JIT，并通过定义mainloop方法以及handle方法实现驱动以达到解析输入程序并生成IR代码的目的。根据教程添加优化器支持。

喻为：

解决了部分词法分析器的问题，包括对正确原程序的识别识别出源程序中的各个单词符号，并将其转换成内部编码形式。在语法分析器中实现编写递归调用程序实现自顶向下的分析。用语法分析搭建一个AST的对象。以及少数注释的添加。

陈昱帆：

实现了对部分词法的解析，以及创建ASTParser并将源文件解析成AST，为进一步的实现和解析做准备。调用ASTParser的createAST方法，得到完整的AST，根据实际需要对AST进行manipulate。在后期进行了一部分的整合和代码规范，添加注释，和对一些头文件的维护。

1. 挑战及思路

本次实验主要是参考llvm的官方文档来实现的，遇到的主要问题是我们所要实现的VSL语言编译器与官方文档中的Kaleidoscope语言有许多不同，所以虽然整体实现上有所参考，但是具体的实现却需要自己去摸索实现，所以在实现时遇到了各种大大小小的问题，比如官方文档中所有的语法解析都是按表达式来解析的，并且VSL中的许多语法在官方文档中都没有涉及到（如RETURN语句、WHILE语句等），最终我们改变了思路，按照VSL语言的语法，改变了语法的解析。在IR代码的生成过程中，因为牵扯到了LLVM的部分，所以实现起来也很困难，参考了llvm的相关文档并了解了相关的实现方法。

1. 课程感想与建议

感想：总的来说，本课程是一门实践性十分强的课程，将我们上学期所学的编译原理的相关知识充分运用起来以实现一门类C语言的编译。本课程不仅让我们更加了解了编译器的相关工作原理，更让我们有了一次亲手实现一门语言的编译器的经验，真正地让我们将书本上的知识应用于实践，巩固了相关知识，增强了实践能力，真正让我们有所收获。

建议：本课程给我们的最深印象就是难，不仅是实现上很有难度，而且官方文档也很难读，并且本课程完全没有任何指导，只能由我们独自摸索完成，所以更是加大了完成的难度。所以我们的建议就是本课程可以适当地加入一些指导环节。