**目 录**

**第1章 绪论**

1.1 研究背景和意义

1.2 编译器技术的研究现状

1.3 主要研究内容

1.3.1 虚拟机

1.3.2 编译系统

**第2章 虚拟机设计与实现**

高级语言与目标机

2.1 指令集设计

2.2 基于寄存器的虚拟机

2.3 基于栈的虚拟机

2.4 数据的存储

2.5 指令的存储

2.6 数据对齐

2.7 虚拟机实现

2.8 虚拟机测试

**第3章 编译器设计**

3.1 符号表设计

3.1.1 符号的表示

3.1.2 作用域的标识

3.1.3 查找和建立标识符

3.1.4 内建符号的处理

3.2 词法分析

3.2.1词法分析算法的设计

3.2.2 词法单元的识别

3.3 语法分析

3.3.1 语言文法的定义

3.3.2 语法分析算法的设计

3.4 语义分析与代码生成

3.4.1 语法制导的翻译技术

3.4.2 代码生成

**第4章 编译器实现与测试**

4.1 词法分析器的实现

4.1.1 标识符的识别

4.1.2 数字的识别

4.1.3 注释的识别

4.1.4 字符与字符串的识别

4.1.5 其他符号的识别

4.2 语法制导的解析程序实现

4.2.1 变量的解析

4.2.2 enum的解析

4.2.3 函数的解析

4.2.4 语句的解析

4.2.5 编译错误

4.2.6 编译警告

4.3编译器测试

4.3.1 变量测试

4.3.2 enum测试

4.3.3 函数测试

4.3.4 语句的测试

4.3.5 整体测试

**第5章 总结与展望**

5.1 本文总结

5.2 未来展望

**致谢**

**参考文献**

* 1. 研究背景和意义

编译器是程序员使用的关键工具，必须生成高效的目标代码，但更重要的是，编译器必须生成正确的目标代码，只有可靠的编译器才能生成可靠的应用程序[2]。编译器本身是一个大而复杂的应用程序。编译器构造原理和技术可以说是计算机科学理论与实践相结合的最好典范。

实现一个编译器，可以学习到编译的一些知识，如词法分析、语法分析、代码生成等，还能清晰地了解程序的运行逻辑，例如变量定义、函数调用、参数传递、内存分配等。因为编译器是一个比较复杂的程序，在实现编译器的过程中，需要设计程序模块、设计符号表、设计运行栈、设计中间代码等，并将它们组合在一起，是非常考验系统设计能力的，对编程能力的提升也是十分巨大的。

1.2编译器技术的研究现状

上世纪50年代，IBM的John Backus带领了一个研究小组对FORTRAN语言及其编译器进行开发。由于当时人们对编译理论的了解不多，开发编译器既复杂又艰苦。与此同时Noam Chomsky也开始他对自然语言结构的研究。Chomsky研究出根据语言文法的难易程度以及识别它们所需的算法来对语言分类，即Chomsky架构。它包括了文法的四个层次：0型文法、1型文法、2型文法和3型文法。其中2型文法（上下文无关文法）被证明是程序设计语言中最有用的。解析问题（用于上下文无关文法识别的有效算法）的研究是在60年代和70年代，它相当完善地解决了这个问题，现在它已是编译原理中的一个标准部分。人们接着又深入研究有效生成目标代码的方法，人们通常称为优化技术，并被一直使用至今。

因为编译器作为基础软件，历史十分悠久，很多人都在研究，所以编译器的很多技术已经相当成熟。例如主流的三大C/C++编译器：GCC、Clang、cl，词法分析器生成程序Lex、Flex，语法分析器生成程序Bison、Yacc，编译器基础设施框架LLVM等。

GCC（GUN Compiler Collection），一套由GNU开发的编程语言编译器，它是一套以GPL及LGPL许可证发布的自由软件，现已被大多数操作系统采纳为标准的编译器。GCC 6.0提供了很多新的特性，如OpenMP 4.5、段寄存器支持、目标克隆、扩展存数规则等，完全支持C++14，并支持C++17的实验功能。

LLVM这个名字源于Lower Level Virtual Machine，但这个项目并不局限于创建一个虚拟机，它已经发展成为当今炙手可热的编译器基础框架[4]。LLVM最初以C/C++为编译目标，近年来经过众多机构和开源社区的努力，LLVM已经能够为ActionScript、D、Fortran、Haskell、Java、Objective-C、Swift、Python、Ruby、Rust、Scala等众多语言提供编译支持，而一些新兴语言则直接采用了LLVM作为后端。可以说，LLVM对编译器领域的发展起到了举足轻重的作用。

Lex是一个词法分析器生成器，yacc是一个语法分析器生成器，它们结合使用可以完成编译器前端的大部分工作，生成的LLVM IR由LLVM进行代码优化、机器码生成等，为实现编程语言提供了方便，避免了很多重复的工作。

1.3主要研究内容

1.3.1 虚拟机

本文的虚拟机指的是为编译器生成的目标语言提供运行环境的程序。为了让目标代码能够执行，较为简单的方式就是自定义指令集，并实现对应的虚拟机，而生成的目标代码就是各种指令。

虚拟机提供存储和运算功能，编译过程产生的数据和生成的指令，存储在虚拟机的数据段和代码段中，编译结束后，将启动虚拟机逐条执行生成的指令，实现程序的运行。

实现一个虚拟机，就是以软件的方式实现硬件功能，为外部提供统一而简单的接口。如真实计算机的CPU内带有多个寄存器，也提供内存用以存储数据，则虚拟机也需要提供类似的功能。不仅仅是实现指令的功能，如何提供寄存器，如何提供内存，不同数据在虚拟机中如何存储，如何进行数据对齐，函数调用时调用帧的结构，都是虚拟机需要考虑的内容。

1.3.2编译器