**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**

2.1 基于寄存器的虚拟机

2.2 基于栈的虚拟机

2.3 指令集设计

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**

摘要

编译器能将以某种语言写的程序转换为等价的目标语言写成的程序。编写编译器，

> 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。

虚拟机通常有两种实现方式，基于栈的虚拟机和基于寄存器的虚拟机。基于栈的虚拟机几乎所有操作都是在栈上执行，没有任何通用寄存器，指令也比较简单。基于寄存器的虚拟机通常则更加

复杂，因为指令涉及多种寄存器操作，因此编译生成的目标代码也需要进行寄存器分配。两者分别适用于不同的场景，基于寄存器的虚拟机单条指令能够完成更多的操作，性能更高，而基于栈

的虚拟机因为指令较为简单，编译器在生成目标代码时也更加简单。

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。

> 2. 编译器

编译器是一个语言翻译程序，能够把源语言书写的程序翻译成等价的目标语言书写的程序。整个翻译过程通常被划分为多个阶段，每个阶段都将源程序的一种表示形式转换成另一种表示形式，

并提供给下一个阶段使用。将编译过程划分为多个阶段可以降低实现复杂度，而且每个阶段都可以采用各自适当的方式进行实现而不影响其他阶段，降低各个阶段间的耦合。

1. 词法分析是编译过程的第一个阶段，任务是对源程序字符流进行扫描和分解，从而识别出一个个的单词。词法分析方法通常包括基于有限自动机的方式，或硬编码方式。

2. 语法分析的作用是识别由词法分析给出的单词符号序列是否是给定文法的正确句子。目前常用的语法分析方法有自顶向下语法分析方法和自底向上语法分析方法。自顶向下分析包括确定分析和不确定分析，自底向上分析又包括算符有限分析和LR分析。它们各有优缺点，分别适用于不同的场景。

3. 语义分析和中间代码生成是生成源程序有误语义错误，并收集类型信息。语义分析会进行类型检查，判断每个算符是否具有语言规范允许的运算对象。中间代码生成则将源程序变成一种内部

表示形式，这是一种结构简单、含义明确的记号系统。

4. 代码优化阶段的任务是对生成的中间代码进行变换或改造，消除不必要的指令，或者把一个指令序列替换为另一个完成相同功能的较快的指令序列，从而使生成的代码更加高效。常用算法有公共子表达式删除、常量传播、强度削弱、循环优化等。

5. 目标代码生成则把中间代码转换成特定机器上的绝对指令或可重定位指令，它是编译过程的最后阶段，工作十分复杂，涉及硬件系统功能部件的使用、指令的选择，各种数据类型变量的存储空间

分配以及寄存器分配等。

此外，一个完整的编译器还应该包括符号表管理和错误处理。这两个内容贯穿整个编译流程，编译各个阶段都涉及到构造、查找和更新各种符号数据，同时编译过程中可能发现源程序的各种错误，

需要提供错误的性质以及错误发生位置，以便用户能够快速且方便地定位错误并解决错误。