图标

描述已自动生成

**编译原理实验报告**

|  |  |  |  |
| --- | --- | --- | --- |
| **专业** | **：** | **计算机科学与技术** |  |
| **班级** | **：** |  |  |
| **学号** | **：** |  |  |
| **姓名** | **：** |  |  |
| **电话** | **：** |  |  |
| **邮箱** | **：** |  |  |

**独创性声明**

本人郑重声明本报告内容，是由作者本人独立完成的。有关观点、方法、数据和文献等的引用已在文中指出。除文中已注明引用的内容外，本报告不包含任何其他个人或集体已经公开发表的作品成果，不存在剽窃、抄袭行为。

特此声明！

作者签名：

日期：2022年 7月 4日

|  |  |
| --- | --- |
| 成绩 |  |
| 教师签名 |  |

目 录

[1 概述 1](#_Toc107844093)

[1.1 实验目的 1](#_Toc107844094)

[1.2 实验任务 1](#_Toc107844095)

[1.3 实验要求 1](#_Toc107844096)

[2 编译工具链的使用 3](#_Toc107844097)

[2.1 实验任务 3](#_Toc107844098)

[2.2 实验实现 3](#_Toc107844099)

[3 词法分析器的设计与实现 4](#_Toc107844100)

[3.1 SysY语言词法分析器的设计 4](#_Toc107844101)

[3.2 SysY语言词法分析器的实现 4](#_Toc107844102)

[4 语法分析器的设计与实现 5](#_Toc107844103)

[4.1 SysY语言语法分析器的设计 5](#_Toc107844104)

[4.2 SysY语言语法分析器的实现 5](#_Toc107844105)

[5 一个L-翻译模式的递归下降翻译程序的实现 7](#_Toc107844106)

[5.1 L-翻译模式自顶向下语义计算 7](#_Toc107844107)

[5.2 一个L-翻译模式递归下降翻译程序的实现 7](#_Toc107844108)

[6 PL/0编译系统分析与修改 8](#_Toc107844109)

[6.1 PL/0编译系统分析 8](#_Toc107844110)

[6.2 PL/0编译系统的修改 8](#_Toc107844111)

[7 总结与展望 10](#_Toc107844112)

[7.1 实验总结 10](#_Toc107844113)

[7.2 实验感想 10](#_Toc107844114)

[7.3 实验展望 10](#_Toc107844115)

[参考文献 11](#_Toc107844116)

# 概述

## 实验目的

《编译原理实验》是配合《编译原理》课程理论教学独立开设的实验课程。目的在于通过一系列专门设计的实验，完成一个高级语言-SysY(C语言的一个子集)的定义和编译。熟悉业界经典的工具软件，从零开始完成一个完整的编译器，将该语言源程序编译为ARM汇编语言程序。

通过实验培养大型工程项目的设计与管理能力，提升系统软件实践能力。

## 实验任务

实验任务包括一个完整的编译器的构造过程，和三个独立的实验。完整的编译器构造过程即构造SysY语言的编译器，又细分为：词法分析，语法分析，符号表管理，语义分析，中间代码生成，代码优化和目标代码生成等阶段。三个独立的实验包括：

1. 编译工具链的使用；
2. 一个L-翻译模式的递归下降翻译程序的实现；
3. PL/0编译器的分析与修改。

## 实验要求

以上实验任务并不需要全部完成，可根据情况选做其中的部分实验(必做的实验除外)。

实验一（编译工具链的使用），实验二（词法分析），实验三（语法分析）的第１关（即语法正确性检查）是必做部分，第２关（抽象语法树的构造和输出）的６个测试用例除用例５（该用例含for语句）以外，原则上应能基本正确地输出其它５个用例的抽象语法树（输出形式不限）。实验三(语法分析)、实验七(中间代码生成)、实验九(代码优化)、实验十(功能评测)等四个实验的得分以老师最后打分为准，其它实验的成绩直接采纳平台评测的分数。

根据自己实际完成情况，据实撰写实验报告。

# 编译工具链的使用

## 实验任务

掌握gcc, clang, arm-linux-gnueabihf-gcc,qemu-arm等编译工具的使用。

## 实验实现

### GCC编译器的使用

从源代码可知，要实现Alibaba和Bili两个人对话的效果，需定义BILIBILI，可用-D参数实现。要定义生成可执行文件名，需使用-o参数。最终命令为：

cc -DBILIBILI def-test.c alibaba.c -o def-test

### CLANG编译器的使用

要生成汇编代码，需使用-S参数；要求优化等级为O2，需指定-O2参数；制定目标代码类型，需使用-target参数；要制定生成文件名，使用-o参数。最终指令为：

clang -S -O2 -target armv7-linux-gnueabihf bar.c -o bar.clang.arm.s

### 交叉编译器arm-linux-gnueabihf-gcc和qemu-arm虚拟机的使用

前两条指令构造方法与2.2.1、2.2.2类似，不再赘述。第三条指令，需使用-L参数指定文件的路径。最终的指令为：

arm-linux-gnueabihf-gcc -S iplusf.c -o iplusf.arm.s

arm-linux-gnueabihf-gcc iplusf.arm.s sylib.a -o iplusf.arm

qemu-arm -L /usr/arm-linux-gnueabihf/ iplusf.arm

# 词法分析器的设计与实现

## SysY语言词法分析器的设计

使用flex词法分析程序的自动构造工具构造此词法分析器。

## SysY语言词法分析器的实现

根据给定的flex框架，将其内容补充完整。主要内容包括，在Token枚举型变量中，增加需要补充的单词类型的代号；在flex的第二段内容中将Token中所有代号对应的文本内容补充完整；以及在第三段内容中在main函数中增加调用flex的语句，及循环调用yylex函数。

在flex的第二部分的符号识别中，需要注意的是单词种类匹配语句的排列顺序，这是由于flex匹配字符串的顺序是由前向后。在词法分析中，我们需要过滤掉注释语句。因此我们需要将注释语句的匹配置于开头。词法中的int、float等关键字，均符合标识符的构成规则，为使关键字得到正确识别，我们需要将关键字匹配部分置于标识符匹配之前。对于各种常量类型的判断，我们需要根据这些类型常量的构成模式来构造合适的正则表达式来进行匹配；对于少数难以通过正则表达式判断的错误形式，例如十六进制数字串中出现0~f之外的字符时（会被错误的识别为数字+标识符+数字），可以将其单独列出以便识别。

flex的第三段较为简单，只需要使用while语句循环调用yylex函数，并将其返回值赋值给一个预设的变量即可。

# 语法分析器的设计与实现

## SysY语言语法分析器的设计

使用bison构造语法分析规则。

与felx类似，bison也分为3部分，第一部分主要定义的是分析文法的终结符、非终结符，并设定了一些符号的优先级和运算符号的结合性。

第二部分是分析文法的规则集。冒号相当于规则中的→，冒号左侧内容为规则的左部，冒号右侧为规则的右部，右部的多种情形以“|”分隔。大括号的内容为成功匹配时执行的操作，一般为创建AST树的一个结点。

第三部分与flex类似，是普通的C代码。主要为语法分析以外的函数语句的内容，包括main函数和其他自定义函数。

AST树采用递归的方式进行构造。每个结点内包含的信息有以该结点为根的子树代表的语法类型，如外部定义序列、函数定义、表达式等；三个指向子结点的指针，这三个子结点内容为文法分析时的子语法项目，如for语句内包含的三个表达式、while语句括号内的一个表达式（此时子结点第一个指向表达式结点，第二个指向语句块结点）；以及这一语句在源代码中出现的行号。

.例如，对于while语句，while语句的一般形式为while(表达式)+语句块。而while语句属于语句的一种，那么对应的规则书写如（语句的其他形式略去）：

Stmt : WHILE LP Exp RP Stmt {$$=mknode(WHILE,$3,$5,NULL,yylineno);}

其中出现的$3，$5代表这一规则右部的第3个实体和第5个实体。

## SysY语言语法分析器的实现

要完成语法分析器的搭建，我们需要补充的内容为Makefile文件、flex 词法分析器sysy.l文件、bison语法分析器sysy.y文件以及AST树操作ast.c文件。

Makefile文件编写较为简单，只需要依次使用bison工具、flex工具与gcc编译器即可。

sysy.l文件的编写与之前的词法分析器编写基本完全一致，不再赘述。

sysy.y文件中内容较多。主要为补充给定文法中未涉及到的部分和不完整的部分，如for语句的实现、数组的识别以及无法识别变量定义、变量赋值交替出现时的错误报错问题。

ast.c文件的修改也较为简单，只需要在原有的switch函数上加入对补充的ast结点类型的遍历语句即可。

# 一个L-翻译模式的递归下降翻译程序的实现

## L-翻译模式自顶向下语义计算

L-翻译模式是一种在规约过程中实现语义计算的方法，其特点为既包含综合属性，也可以包含继承属性。

例如，在本题中，有如下规则：

S → B {R.ival = B.val; R.ilen = 1} R { S.val = R.val; S.len := R.len }

去掉将大括号内容，这条语句表示左部为S的一条推导规则。大括号内的内容表示的是右部属性的计算方法。其中，ival，ilen为继承属性，为自顶向下传递的属性，故在编写函数时将其作为函数的参数向下传递。而val，len为综合属性，为自底向上传递的属性，只需按照计算规则进行计算即可。

## 一个L-翻译模式递归下降翻译程序的实现

基于之前的描述我们可以较容易地构造出基于L-翻译模式的递归下降翻译程序。核心思路为，对于继承属性我们将其作为参数向下传递、对于综合属性我们直接按照所给表达式进行计算。

parserS函数的构建直接根据表达式即可构建。而parserR函数由于R的推导有两种情况，需要分别进行讨论。当分析中读取到换行符或小数点时，代表匹配到的是规则R→ε；其他情况下则是规则R→BR1，我们需要根据传递属性的公式，递归地传递给ParseR函数来向下分析。

# PL/0编译系统分析与修改

## PL/0编译系统分析

PL/0实际是一种极简化的PASCAL语言，它的诞生纯粹是为了教学，即演示如何设计编译系统。PASCAL语言是上世纪七八十年代非常流行的语言，其设计者为ACM图灵奖获得者Niklaus Wirth。它语法严谨，层次分明，程序易写，可读性强，是第一个结构化编程语言，现在仍有少量用户。目前PASCAL仍是信息学科奥林匹克竞赛的官方语言之一(另外两个是C和C++)。本次实验中所使用的编译系统是基于Wirth的PASCAL版本改写而来。

PL/0词法分析主要由函数getsym()完成,getsym会调用getch()。getsym返回下一个token，返回的token存储在全局变量symbol(enum类型)，其值则存储于全局变量id(标识符、关键字、界符、运算符)或num(数字)

## PL/0编译系统的修改

为完成任务，只需要完成修改要求中的几项任务即可。

1. 被编译的源程序(如gcd.pl0)作为编译器命令行参数

较为简单，将程序的main函数从无参数修改为接受命令行参数的形式，即一个整形变量argc和一个char类型的二重指针argv，同时删去读取源文件名的语句，而将源文件名直接指定为argv[1]即可。

1. 编译期间不再提示输入源程序名，不再询问是否输出虚拟机代码，不再询问是否输出符号表

直接删去这三处的提示输出语句即可。

1. 编译期间不输出符号表，也不输出源程序内容(除非发现错误)

观察main程序可知，控制符号表和源程序内容输出的两个参数分别为listswitch和tableswitch，只需要将这两个参数直接指定为0即可屏蔽这两项输出。

1. 编译期间不输出中间代码，但在虚拟机解释执行中间代码前调用listallcode()输出中间代码

分析程序可知，在block函数（分析分程序）的结尾，会调用listcode函数将中间代码输出。我们需要在block函数中取消对listcode函数的调用，并填充listallcode函数，使其能一次性将语句输出到fa2所指向的文件fa2.tmp中。listallcode函数的改写较为简单，只需要用一个while语句进行循环输出即可。其代码如下

void listallcode()

{

for(int i=0;i<cx;i++)

{

fprintf(fa2,"%d %s %d %d\n", i, mnemonic[code[i].f], code[i].l, code[i].a);

}

}

# 总结与展望

## 实验总结

本次实验中，我完成的内容包括编译工具链的使用，flex词法分析器的设计实现、bison语法分析器的设计实现、L-翻译模式的递归下降分析程序的实现以及PL/0编译系统的分析和修改。但受制于时间和精力，未能完成实现完整编译器剩余内容。

## 实验感想

实验中遇到的一个较大问题为实训2词法分析器中，flex工具的使用。flex工具极大地简化了词法分析器的编写工作，只需要填写正则表达式即可自动生成分析程序，而不是繁琐的进行大量if-else分支编写。但虽然课上对正则表达式有一定的讲解，但其与实际使用的正则表达式语法尚有一定的区别，需要花时间去查找资料和学习。在编写词法分析时，我也遇到了诸如3.2中提到的十六进制数被分为多部分等问题，这可能需要编写额外的辨识错误写法的正则式。这让我了解到，当错误的判断较为困难时，可以将这种错误类型进行单独的判断。

## 实验展望

本次实验未能完成的部分包括符号表管理、静态语义分析、中间代码生成与优化等部分，主要是由于在实验3语法分析中花费了过多时间来了解源程序结构和功能。语法分析中遇到的部分问题，如变量声明和赋值的交替出现的处理等，是我体会到完整实现一个编译器的复杂之处，使我认识到自己能力与实现编译器所需能力之间的差距。

# 参考文献

[1] 王生原，董渊，张素琴，吕映芝，蒋维杜. 编译原理（第三版）. 清华大学出版社，2015