《编译技术》课程设计文 档

学号：76066001

姓名：张金源

2018年 11 月 18日

## 需求说明

### 1．文法说明

我获取的文法是PL/0文法，PL/0语言是一种类PASCAL语言，是教学用程序设计语言，它比PASCAL语言简单，作了一些限制。PL/0的程序结构比较完全，赋值语句作为基本结构，构造概念有

* 顺序执行、条件执行和重复执行，分别由begin/end,if then else和while do语句表示。
* PL0还具有子程序概念，包括过程说明和过程调用语句。
* 在数据类型方面，PL0只包含唯一的整型，可以说明这种类型的常量和变量。
* 运算符有+，-，\*，/，=，<>，<，>，<=，>=，(，)。
* 说明部分包括常量说明、变量说明和过程说明。

<程序>      ::=  <分程序>.  
<分程序>     ::=  [<变量说明部分>][<常量说明部分>][<过程说明部分>]<复合语句>  
<常量说明部分>  ::=  const<常量定义>{,<常量定义>};  
<常量定义>   ::=  <标识符>=<无符号整数>  
<无符号整数>  ::=  <非零数字>{<数字>}|0  
<标识符>                ::=  <字母>{<字母>|<数字>}  
<变量说明部分> ::=  var<标识符>{, <标识符>};  
<过程说明部分> ::=  <过程首部><分程序>{;<过程首部><分程序>};  
<过程首部>   ::=  procedure<标识符>;  
<语句>     ::=  <赋值语句>|<条件语句>|<当循环语句>|<过程调用语句>|<复合语句>|<读语句>|<写语句>|<空>   
<赋值语句>   ::=  <标识符> := <表达式>  
<表达式>  ::=  [+|-]<项>{<加法运算符><项>}//[+|-]只作用于第一个<项>

<项>    ::=  <因子>{<乘法运算符><因子>}  
<因子>   ::=  <标识符>|<无符号整数>|'(' <表达式>')'  
<加法运算符>  ::=  +|-  
<乘法运算符>  ::=  \*|/  
<条件>     ::=  <表达式><关系运算符><表达式>|odd<表达式>  
<关系运算符>  ::=  =|<>|<|<=|>|>=  
<条件语句>   ::=  if<条件>then<语句>  
<当循环语句>  ::=  while<条件>do<语句>   
<过程调用语句>  ::=  call<标识符>   
<复合语句>   ::=  begin<语句>{;<语句>}end  
<读语句>    ::=  read'('<标识符>{, <标识符>}')'   
<写语句>    ::=  write'('<表达式>{, <表达式>}')'  
<字母>     ::=  a|b|c|d...|x|y|z|A|B...|Z   
<数字>     ::=  0|<非零数字>  
<非零数字>   ::=  1|2|3...|8|9

### 2．目标代码说明

P-code是为目标代码。P-code 语言：一种栈式机的语言。此类栈式机没有累加器和通用寄存器，有一个栈式存储器，有四个控制寄存器（指令寄存器 I，指令地址寄存器 P，栈顶寄存器 T和基址寄存器 B），算术逻辑运算都在栈顶进行。指令格式为F，L，A

F: 操作吗， L: 层次差， A: 不同的指令含义。

P-code指令含义

|  |  |
| --- | --- |
| 指令 | 具体含义 |
| LIT 0,a | 取常量a放到数据栈栈顶 |
| OPR 0,a | 执行运算，a表示执行何种运算(+ - \* /) |
| LOD l,a | 取变量放到数据栈栈顶(相对地址为a,层次差为l) |
| STO l,a | 将数据栈栈顶内容存入变量(相对地址为a,层次差为l) |
| CAL l,a | 调用过程(入口指令地址为a,层次差为l) |
| INT 0,a | 数据栈栈顶指针增加a |
| JMP 0,a | 无条件转移到指令地址a |
| JPC 0,a | 条件转移到指令地址a |

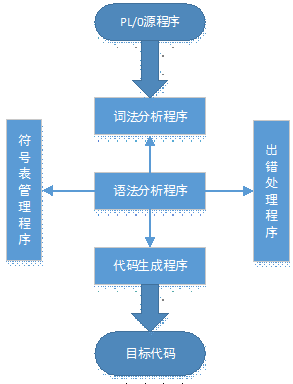
### 3. 优化方案\*

【说明需要完成的优化方案及其要求】

## 详细设计

【应包括但不限于以下内容】

### 1．程序结构



首先程序分为5个阶段，第一个阶段是词法分析程序，就是把每个单词分析下来。然后语法分析，语法分析是用递归子程序法对每种语法成分进行分析。代码生成程序是能够输出目标代码（P-code）。出错处理程序是处理源代码的各个错误（报错）一般为数字来报并输出在哪出现错误。符号表管理是从词法分析来存储的，把每次读到的符号存到栈里边。这编译系统用了一遍扫描，即以语法分析为核心，由它调用词法分析程序取单词，在语法分析过程中同时进行语义分析处理，并生成目标指令。目标代码为p-code。

### 2．类/方法/函数功能

本程序是用C/C++实现，函数说明如下：

void scanner 函数（getsym） ：

这函数的作用是作词法分析的函数，它把每个读的单子都记录下来，然后把每个单词都分析为各个的类别码，单词的值，和单词的类别。

int read函数：

这函数的作用是做读入程序的源代码（文件），它把源代码读入到编译程序里面，然后读入之后就把它分析，要是没找到对应的文件程序直接停而报错。

语法分析的函数：

采用了递归子程序方法进行语法分析，即为每个语法成分都编写了一个分析子程序，根据当前读取的符号，可以选择相应的子程序进行语法分析。

符号表处理：

这部分函数是来处理符号表的存储方式，包括符号表的组织方式

符号表上的操作

出错处理：

这函数需要能的发现源程序中出现的语法和语义错误，并应尽可能准确地指出出错位置和出错性质。

目标代码的生成和解释执行：

源程序经过编译得到了一个假想的计算机的目标代码，即P-code指令代码。P-code指令不依赖于任何实际的计算机，它的指令集极为简洁，指令格式也非常单纯。P-code指令可通过解释执行函数（interpret）。

### 3．调用依赖关系

根据程序的逻辑结构,在实现上我将我的程序划分为 5 个模块:符号表、声明处理、语

句处理、解释执行、错误处理。声明部分相对独立,其他部分相互嵌套调用。

#### 符号表

符号表是用来记录、查找源程序中的所有标识符的信息,通过符号表,为声明过程、语

句处理、解释执行提供相应的信息。符号表的管理包括符号表的设计(详见二中 4 符号表管

理方案),登陆符号表,符号表定位,符号表重定位。

#### 声明处理

声明包括常量声明,变量声明、过程声明、函数声明,声明部分主要工作是根据语法分

析,将合法的信息登入符号表,声明部分相对来说较为独立,其实质就是登陆符号表,在登

陆符号表的时候,因为需要对标识符值类型进行判断,对于过程声明和函数声明还需要判断

形参的个数和形参类型,所以需要返填符号表,需要动态登陆符号表。常量声明,变量声明,

过程声明、函数声明各写了一个函数进行处理,分别为:

ConstDefine(),varDefine(),FuncProDeclare()。

#### 语句处理

语句处理是程序的核心部分,通过语法分析,查找符号表,生成相应的 P-code 代码,

在解释执行时,根据生成的 P-code 代码解释执行。语句包括赋值语句、条件语句、当循环

语句、过程调用语句、复合语句、读语句、写语句、和空语句,各语句之间也

是相互嵌套,相互调用的。

#### 解释执行

程序的解释执行是通过生成的 P-code 完成的,P-code 指令详见“目标代码说明”部分,

语句处理部分生成的 P-code 指令储存在 codeTable 中,解释执行是循环遍历 codeTable 根据

规定的规则进行执行。

#### 错误处理

错误处理嵌套于声明过程和语句处理部分,包括语法错误和语义错误,在进行声明处理

和语句处理时,如果不符合语法或语义,则报告相应错误,并跳到下一条语句或下一块。

### 4．符号表管理方案

#### 符号表的组织方式

考虑到不同类型的符号所需记录的信息是不完全相同的,符号表的组织方式上采用了一

个折衷的方法,即将大部分共同信息组织成统一格式的符号表(Table),特殊信息另设附表,

二者通过建立一个索引进行连接。

#### 符号表上的操作

符号表上的操作包含插入和检索,对于每一个 block 还有定位和重定位操作。

##### 4.2.1 插入

在进行声明部分的时候,根据常量声明、变量声明、过程声明、函数声明,将相关信息

登入到符号表中。

在插入前需要通过查表判断标识符的作用域。在声明部分读到标识符时,首先检查该标

识符所在程序单元的符号表,如果表中已存在同名的标识符,则说明重复声明了标识符,应

该报错;如果表中不存在同名标识符,则将新声明的标识符插入到符号表中。

##### 4.2.2检索

检索有两个分支,一是在声明时的检索,而是在语句中的检索。

1. 声明部分检索

在声明部分读到标识符时,首先检查该标识符所在程序单元的符号表,如果表中已

存在同名的标识符,则说明重复声明了标识符,应该报错;如果表中不存在同名标识符,

则将新声明的标识符插入到符号表中。

1. 执行语句部分检索

在程序的可执行语句部分,读到标识符时,需要查表以判断该标识符是否已经声明,

首先差标识符所在程序单元的符号表,如果表中存在同名的标识符,则说明该标识符已经声明;如果在本层的符号中未查找到同名的标识符,则再查直接外层的符号表,如果在逐层次向外查找的过程中找到同名标识符,说明该标识符已经声明,返回该标识符在符号表的位置,否则报错。

##### 4.2.3定位与重定位

在编译过程中,识别出是分程序的开始时,就执行定位操作;当遇到分程序结束标志时,就执行重定位操作。每一次进入分程序就设置一个变量保存分程序入口处的符号表下标,实现定位操作,而每一次到达分程序的出口,就需要进行重定位操作,重定位操作需要将符号表中分程序对应的局部变量进行屏蔽,在实现中采用将 level 置为-1 的方法实现。

### 5．存储分配方案

参考 pl0 编译器存储器的设计,设计了两个存储器、一个指令寄存器、三个地址寄存器。

#### P-code 指令存储 code

建立一个 code 数组存放生成的 P-code 指令,大小为 1000,如果指令数量超过 1000,

则溢出,报错。

#### 数据存储器 s

数据区采用动态存储分配,通过建立一个栈式数据存储器动态分配程序的数据空间(栈式动态存储分配),结构设计如下:

|  |
| --- |
| 运算区间 |
| 局部变量 |
| RA |
| DL |
| SL |

SL：静态链,保存直接外层分程序数据区基地址。

DL：动态链,保存原调用过程的数据区的基地址。

RA：返回地址,保存应返回的调用点(下一条指令)的指令地址。

所有的算术运算都在数据堆栈栈顶的两个单元之间进行,并用运算结果取代原来的两个运算对象儿保留在栈顶。栈顶单元的地址放在栈顶地址寄存器中,即栈顶地址寄存器作为栈顶指针永远指向数据栈的栈顶。用另一个指令寄存器存放当前要执行的指令。与之对应,设置一个程序地址寄存器存放下一条要执行的指令地址,每执行一条指令,程序地址寄存器的地址值加 1(数组下标加 1)。此外,再设置一个基址寄存器,用于存放当前的分程序数据区在数据栈中的起始地址。

### 6. 解释执行程序\*

解释执行程序根据存储分配方案设计的数据在栈顶与次栈顶进行操作，具体执行何种操作有生成的P-code进行。

### 7. 四元式设计\*

【对采用的四元式进行详细说明】

### 8. 目标代码生成方案\*

【说明代码生成有关的数据结构、关键算法】

### 9. 优化方案\*

【说明代码优化有关的数据结构、关键算法】

### 10．出错处理

1）对于明显的错误,报告出错误信息并且自动进行自动校正,继续进行编译。

2）在每个语法分析子程序出口处,检查下一个取来的符号是否为该语法成分的合法后继符号。若不是,则报告出错误信息,并且跳读一段源程序,直至去来的符号属于该语法成分的合法后继符集合为止。为防止跳过太多的源程序,设置一个停止符号集合,只要取出的符号属于合法后继符号集合或者停止符号集合,程序就停止跳读。

## 三．操作说明

### 1．运行环境

【说明搭建运行环境的步骤】

### 2．操作步骤

【详细说明操作步骤】

## 四．测试报告

### 1．测试程序及测试结果

【给出提供的测试程序以及每个程序的测试结果，至少5个正确程序，5个错误程序，无需截屏】

### 2．测试结果分析

【说明上述测试程序对语法成分的覆盖情况】

## 五．总结感想

【说明在完成课程设计中的收获、认识和感想】