《编译技术》课程设计文 档

学号：\_\_\_\_\_\_\_\_\_16061182\_\_\_\_\_\_\_\_\_\_\_

姓名：\_\_\_\_\_\_\_\_\_苗萌\_\_\_\_\_\_\_\_\_\_\_

2018 年 11 月 17 日

## 一．需求说明

### 1．文法说明

文法无需改写，符合要求。

＜加法运算符＞ ::= +｜-  
＜乘法运算符＞  ::= \*｜/  
＜关系运算符＞  ::=  <｜<=｜>｜>=｜!=｜==  
＜字母＞   ::= ＿｜a｜．．．｜z｜A｜．．．｜Z  
＜数字＞   ::= ０｜１｜．．．｜９  
＜字符＞    ::=  '＜加法运算符＞'｜'＜乘法运算符＞'｜'＜字母＞'｜'＜数字＞'  
＜字符串＞   ::=  "｛十进制编码为32,33,35-126的ASCII字符｝"  
＜程序＞    ::= ［＜常量说明＞］［＜变量说明＞］{＜有返回值函数定义＞|＜无返回值函数定义＞}＜主函数＞  
＜常量说明＞ ::=  const＜常量定义＞;{ const＜常量定义＞;}  
＜常量定义＞   ::=   int＜标识符＞＝＜整数＞{,＜标识符＞＝＜整数＞}  
                               | char＜标识符＞＝＜字符＞{,＜标识符＞＝＜字符＞}  
＜无符号整数＞  ::= ＜数字＞｛＜数字＞｝  
＜整数＞        ::= ［＋｜－］＜无符号整数＞  
＜标识符＞    ::=  ＜字母＞｛＜字母＞｜＜数字＞｝  
＜声明头部＞   ::=  int＜标识符＞ |char＜标识符＞  
＜变量说明＞  ::= ＜变量定义＞;{＜变量定义＞;}  
＜变量定义＞  ::= ＜类型标识符＞(＜标识符＞|＜标识符＞‘[’＜无符号整数＞‘]’){,(＜标识符＞|＜标识符＞‘[’＜无符号整数＞‘]’ )}  //＜无符号整数＞表示数组元素的个数，其值需大于0  
＜类型标识符＞      ::=  int | char  
＜有返回值函数定义＞  ::=  ＜声明头部＞‘(’＜参数表＞‘)’ ‘{’＜复合语句＞‘}’|＜声明头部＞‘{’＜复合语句＞‘}’  //第一种选择为有参数的情况，第二种选择为无参数的情况  
＜无返回值函数定义＞  ::= void＜标识符＞(’＜参数表＞‘)’‘{’＜复合语句＞‘}’| void＜标识符＞{’＜复合语句＞‘}’//第一种选择为有参数的情况，第二种选择为无参数的情况  
＜复合语句＞   ::=  ［＜常量说明＞］［＜变量说明＞］＜语句列＞  
＜参数表＞    ::=  ＜类型标识符＞＜标识符＞{,＜类型标识符＞＜标识符＞}  
＜主函数＞    ::= void main‘(’‘)’‘{’＜复合语句＞‘}’  
＜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＞｜＜标识符＞‘[’＜表达式＞‘]’|‘(’＜表达式＞‘)’｜＜整数＞|＜字符＞｜＜有返回值函数调用语句＞  
＜语句＞    ::= ＜条件语句＞｜＜循环语句＞| ‘{’＜语句列＞‘}’| ＜有返回值函数调用语句＞;   
                           |＜无返回值函数调用语句＞;｜＜赋值语句＞;｜＜读语句＞;｜＜写语句＞;｜＜空＞;|＜返回语句＞;  
＜赋值语句＞   ::=  ＜标识符＞＝＜表达式＞|＜标识符＞‘[’＜表达式＞‘]’=＜表达式＞  
＜条件语句＞::= if ‘(’＜条件＞‘)’＜语句＞[else＜语句＞]  
＜条件＞    ::=  ＜表达式＞＜关系运算符＞＜表达式＞｜＜表达式＞ //表达式为0条件为假，否则为真  
＜循环语句＞   ::=  while ‘(’＜条件＞‘)’＜语句＞ |for'('＜标识符＞＝＜表达式＞;＜条件＞;＜标识符＞＝＜标识符＞(+|-)＜步长＞')'＜语句＞  
＜步长＞::= ＜无符号整数＞  
＜有返回值函数调用语句＞ ::= ＜标识符＞‘(’＜值参数表＞‘)’|<标识符> //第一种选择为有参数的情况，第二种选择为无参数的情况  
＜无返回值函数调用语句＞ ::= ＜标识符＞‘(’＜值参数表＞‘)’|<标识符> //第一种选择为有参数的情况，第二种选择为无参数的情况  
＜值参数表＞   ::= ＜表达式＞{,＜表达式＞}  
＜语句列＞   ::= ｛＜语句＞｝  
＜读语句＞    ::=  scanf ‘(’＜标识符＞{,＜标识符＞}‘)’  
＜写语句＞    ::= printf ‘(’ ＜字符串＞,＜表达式＞ ‘)’| printf ‘(’＜字符串＞ ‘)’| printf ‘(’＜表达式＞‘)’  
＜返回语句＞   ::=  return[‘(’＜表达式＞‘)’]    
  
附加说明：  
（1）char类型的变量或常量，用字符的ASCII码对应的整数参加运算  
（2）标识符不区分大小写字母  
（3）写语句中，字符串原样输出，单个字符类型的变量或常量输出字符，其他表达式按整型输出   
（4）数组的下标从0开始  
（5）for语句先执行一次循环体中的语句再进行循环变量是否越界的测试

（6）允许++4、--4、+-4、-+4的表示出现，前两种结果为正整数4，后两种结果为负整数4

### 2．目标代码说明

目标代码为MIPS指令，种类如下所示（[$s2]表示以$s2为地址保存的字）

1. syscall，系统调用，不同的$v0代表不同的功能

2、li $s1,2，记录立即数，$s1 = 2

3、la $s1,list1，取地址，$s1 = list1的首地址

4、add $s1,$s2,$s3，加法，$s1 = $s2 + $s3

5、addi $s1,$s2,2，立即数加法，$s1= $s2 + 2

6、sub $s1,$s2,$s3，减法，$s1 = $s2 - $s3

7、subi $s1,$s2,2，立即数减法，$s1 = $s2 - 2

8、mult $s1,$s2，乘法，$hi,$lo = $s1 \* $s2

9、div $s1,$s2，除法，$lo = $s1 / $s2,$hi = $s1 % $s2

10、mfhi $s1，取$hi寄存器的值，$s1 = $hi

11、mflo $s1，取$lo寄存器的值，$s1 = $lo

12、move $s1,$s2，赋值，$s1 = s2

13、lw $s1,0($s2)，取字，$s1 = [$s2]

14、sw $s1,0($s2)，存字，[$s2] = $s1

15、beq $s1,$s2,label，相等则跳转，$s1 == $s2 => goto label

16、bne $s1,$s2,label，不等则跳转，$s1 != $s2 => goto label

17、blt $s1,$s2,label，小于则跳转，$s1 < $s2 => goto label

18、bgt $s1,$s2,label，大于则跳转，$s1 > $s2 => goto label

19、ble $s1,$s2,label，小于等于则跳转，$s1 <= $s2 => goto label

20、bge $s1,$s2,label，大于等于则跳转，$s1 >= $s2 => goto label

21、j label，无条件跳转，goto label

22、jr $ra，跳转到指定位置，pc = $ra

23、jal label，无条件跳转并链接，$ra = pc + 4,goto label

### 3. 优化方案\*

1、常数合并：表达式计算时，若已知部分操作数的值，可以先对这些值进行预处理，之后再与变量一起计算

2、常量替换：在编译的时候把常量替换为数字，之后就不再对其进行取值

3、消除公共子表达式：划分基本块，删除公共子表达式

4、全局寄存器分配：引用计数算法和着色图算法

5、数据流分析：活跃变量分析，定义和使用链

6、窥孔优化：在较短的指令序列中，删除冗余或者低效率的指令，保证最终指令序列简洁而高效

## 二．详细设计

### 1．程序结构

程序分为六个部分：词法分析、语法和语义分析、中间代码生成、目标代码生成、错误处理、优化。

语法和语义分析部分调用词法分析部分，并调用中间代码生成部分生成中间代码，若在此过程中遇到错误，则调用错误处理部分。之后调用目标代码生成部分来生成目标代码，最后调用优化部分来优化目标代码。

### 2．类/方法/函数功能

词法分析：

1、void getch()，读取单个字符

2、void getsym()，读取一个符号

语法分析和语义分析：

1. void deccon()，常量说明子程序
2. void defcon()，常量定义子程序
3. void integer()，整数子程序
4. void dechead()，声明头部子程序
5. void decvar()，变量说明子程序
6. void defvar()，变量定义子程序
7. void deffunct()，有返回值函数定义子程序
8. void deffuncf()，无返回值函数定义子程序
9. void multistate()，复合语句子程序
10. void paralist()，参数表子程序
11. void mainfunc()，主函数子程序
12. void expr()，表达式子程序
13. void item()，项子程序
14. void factor()，因子子程序
15. void state()，语句子程序
16. void fuzhistate()，赋值语句子程序
17. void tiaojianstate()，条件语句子程序
18. void tiaojian()，条件子程序
19. void xunhuanstate()，循环语句子程序
20. void callfunctstate()，有返回值函数调用语句子程序
21. void callfuncfstate()，无返回值函数调用语句子程序
22. void valueparalist()，值参数表子程序
23. void states()，语句列子程序
24. void readstate()，读语句子程序
25. void writestate()，写语句子程序
26. void returnstate()，返回语句子程序

中间代码生成：

1. void getmid()，得到中间代码

优化部分：

1. void conbcon()，常数合并
2. void replcon()，常数替换
3. void delexpr()，删除公共子表达式
4. void allocreg()，分配全局寄存器
5. void datastream()，数据流分析
6. void delcode()，删除无用代码

目标代码生成：

1. void printcode()，根据中间代码得到mips指令

错误处理：

1、void error()，错误处理

### 3．调用依赖关系

语法分析、语义分析函数会调用getsym()读取符号，getsym()会调用getch()读取字符；getsym()、语法和词法分析函数报错时会调用error()。除此之外，其他函数调用关系如下：

1. deccon()调用defcon()
2. defcon()无
3. integer()无
4. dechead()无
5. decvar()调用defvar()
6. defvar()无
7. deffunct()调用dechead()，paralist()，multistate()
8. deffuncf()调用paralist()，multistate()
9. multistate()调用deccon()，decvar()，states()
10. paralist()，无
11. mainfunc()，调用multistate()
12. expr()，调用item()
13. item()，调用factor()
14. factor()，调用expr()，integer()，callfunctstate()
15. state()，调用tiaojianstate()，xunhuanstate()，states()，callfunctstate()，callfuncfstate()，fuzhistate()，readstate()，writestate()，returnstate()
16. fuzhistate()，调用expr()
17. tiaojianstate()，调用tiaojian()，state()
18. tiaojian()，调用expr()
19. xunhuanstate()，调用tiaojian()，state()，expr()
20. callfunctstate()，调用valueparalist()
21. callfuncfstate()，调用valueparalist()
22. valueparalist()，调用expr()
23. states()，调用state()
24. readstate()，无
25. writestate()，调用expr()
26. returnstate()，调用expr()

### 4．符号表管理方案

符号表的数据结构（无序符号表）：

**typedef** struct**{**

char name**[**IDENL**];**//符号id

int type**;**//符号种类,0:const,1:var,2:func,3:para

int value**;**//符号值,如果符号是函数名,则0:函数为int型,1:函数为char型,2:函数为void型

int address**;**//标识符存储地址或者相对地址

int para**;**//函数:表示参数个数,数组:表示数组大小

int level**;**//作用域的嵌套层次

**}**symbol**;**

**typedef** struct**{**

symbol element**[**MAXTAB**];**//符号表

int index**;**//符号表栈顶指针

int totalpre**;**//记录当前符号表拥有的分程序总数

int indexofpre**[**MAXPRE**];**//分程序的索引

**}**tab**;**

符号表管理算法：

用tab来记录符号表，存储不同层次的符号表，使用findsym()函数来判断当前变量是否在符号表中声明过，使用pushsym()来将符号插入符号表，另有一系列函数用来给symbol的域赋值。

### 5．存储分配方案

运行栈按照课本示例设计，如下图所示：

|  |
| --- |
| 局部参数区 |
| 形参数据区（显式参数区）  参数区 |
| prev abp（调用模块基地址） |
| ret addr（下一条指令地址） |
| ret value（返回值） |
| display区 |
| …… |
| …… |
| 主函数变量定义区 |

采用固定寄存器分配方案，共使用三个寄存器，两个是从内存中读入的数，一个是计算结果，然后再把计算结果存入内存。

### 6. 解释执行程序\*

【说明解释执行程序的数据结构，关键算法，及解释执行过程】

### 7. 四元式设计\*

四元式是一种N-元表示，每条指令由4个域组成：

<操作符> , <操作数1> , <操作数2> , <结果>

例如(a + b) \* (c + d) – e可以表示成

+, a, b, t1

+, c, d, t2

\*, t1, t2, t3

-, t3, e, t4

其中<结果>一项一般为临时变量，可以将其存储在寄存器中。

### 8. 目标代码生成方案\*

目标代码（mips指令）是由中间代码（四元式）生成的。目标代码生成的主要任务包括：从中间代码到目标代码的转换过程中进行指令选择，目标代码地址空间的划分，目标体系结构上寄存器和内存单元的分配和指派，目标代码序列的生成等。

### 9. 优化方案\*

其他优化方案要等到实现时再考虑是否采用，因此先只写一个消除公共子表达式：

1. 初始化一个放置 DAG图中间结点的队列。

2. 如果DAG图中还有中间节点未进入队列，则执行步骤 3，否则执行步骤 5。

3. 选取一个尚未进入队列，但其 所有父节点均已进入队列 的中间节点n，将其加入队列；或选取 没有父节点的中间节点，将其加入队列。

4. 如果n的最左子节点符合步骤3的条件，将其加入队列；并沿着当前节点的最左边，循环访问其最左子节点，最左子节点的最左子节点等，将符合步骤3条件的中间节点依次加入队列；如果出现不符合步骤 3条件的最左子节点，执行步骤 2。

5. 将中间节点队列逆序输出，便得到中间节点的计算顺序，将其整理成中间代码序列。

### 10. 出错处理

处理方式：输出错误发生的位置和错误的种类

错误种类如下：

1. 文件打开失败
2. 引用未定义函数名
3. 引用未定义标识符
4. 表达式语法错误
5. 数值应该是整数
6. 缺少;
7. 缺少,
8. 缺少一对()
9. 缺少一对[]
10. 缺少一对{}
11. 缺少(
12. 缺少)
13. 缺少[
14. 缺少]
15. 缺少{
16. 缺少}
17. 数组越界
18. 数组长度为0
19. 函数名冲突
20. 变量名冲突
21. 局部变量名和宏定义冲突
22. 在函数的声明头部部分没有参数名
23. 在常量声明时没有给常量赋值
24. 常量或变量声明时类型不为int或char
25. 常量或变量声明时缺省int或char
26. 非法字符
27. 缺少预定义关键字
28. 除数为0
29. 缺少返回语句
30. 函数返回值类型错误
31. 赋值时等号两边类型不匹配或类型错误
32. 函数调用时传入的参数错误

## 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，感想】

注：【】内的文字为文档模板说明，完成的作业中需去掉。

标\*的章节需根据题目的难度进行取舍。