《编译技术》课程设计

文 档

学号：\_\_\_\_\_15231204\_\_\_\_\_\_

姓名：\_\_\_\_\_\_\_周扬\_\_\_\_\_\_\_\_

年 月 日

## 一．需求说明

### 1．文法说明

【说明获取的文法、对文法的改写和扩充】

1.1 原始文法：

＜加法运算符＞ ::= +｜-  
＜乘法运算符＞  ::= \*｜/  
＜关系运算符＞  ::=  <｜<=｜>｜>=｜!=｜==  
＜字母＞   ::= ＿｜a｜．．．｜z｜A｜．．．｜Z  
＜数字＞   ::= ０｜＜非零数字＞  
＜非零数字＞  ::= 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＞  ::= ＜非零数字＞｛＜数字＞｝  
＜整数＞        ::= ［＋｜－］＜无符号整数＞｜０  
＜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＞‘]’) }  
＜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＞‘}’  
＜无返回值函数定义＞  ::= void＜标识符＞‘(’＜参数表＞‘)’‘{’＜复合语句＞‘}’  
＜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＞}| ＜空＞  
＜主函数＞    ::= void main‘(’‘)’ ‘{’＜复合语句＞‘}’  
＜表达式＞    ::= ［＋｜－］＜项＞{＜加法运算符＞＜项＞}  
＜项＞     ::= ＜因子＞{＜乘法运算符＞＜因子＞}  
＜因子＞    ::= ＜标识符＞｜＜标识符＞‘[’＜表达式＞‘]’｜＜整数＞|＜字符＞｜＜有返回值函数调用语句＞|‘(’＜表达式＞‘)’  
＜语句＞    ::= ＜条件语句＞｜＜循环语句＞｜<情况语句>|‘{’＜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＞  
＜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＞   ::=  do＜语句＞while ‘(’＜条件＞‘)’

＜常量＞   ::=  ＜整数＞|＜字符＞

＜情况语句＞  ::=  switch ‘(’＜表达式＞‘)’ ‘{’＜情况表＞ ‘}’  
＜情况表＞   ::=  ＜情况子语句＞{＜情况子语句＞}  
＜情况子语句＞  ::=  case＜常量＞：＜语句＞

＜有返回值函数调用语句＞ ::= ＜标识符＞‘(’＜值参数表＞‘)’  
＜无返回值函数调用语句＞ ::= ＜标识符＞‘(’＜值参数表＞‘)’  
＜值参数表＞   ::= ＜表达式＞{,＜表达式＞}｜＜空＞  
＜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＞‘)’]

文法的附加说明：

（1）char类型的表达式，用字符的ASCII码对应的整数参加运算，在写语句中输出字符

（2）标识符不区分大小写字母

（3）写语句中的字符串原样输出

（4）数组的下标从0开始

（5）情况语句中，switch后面的表达式和case后面的常量只允许出现int和char类型；每个情况子语句执行完毕后，不继续执行后面的情况子语句

1.2 属性翻译文法（待补充）

### 2．目标代码说明

【说明要生成的目标代码指令及含义】

1.1 目标代码：

Mips汇编代码，由于c0文法简单且实现的功能有限，所以一些基础指令足以支撑程序实现。以下是汇编代码举例：

LB 加载字节

LW 加载字

SB 存储字节

SW 存储字节

ADD 符号加

SUB 符号减

MULT 符号乘

DIV 符号除

SLT 小于置一

BEQ 等于转移

BNE 不等转移

BLEZ 小于等于零时转移

BGTZ 大于零转移

BLTZ 小于零转移

BGEZ 大于等于零转移

J 跳转

JAR 跳转并链接

MFHI 读HI寄存器

MFLO 读LO寄存器

MTHI 写HI寄存器

MTLO 写LO寄存器

SYSCALL 系统调用

### 3. 优化方案\*

【说明需要完成的优化方案及其要求】

3.1 方案：

基本块内部的公共子表达式删除（DAG图）；

全局寄存器分配（引用计数或着色算法）；

数据流分析（通过活跃变量分析，或利用定义-使用链建网等方法建立冲突图）；

## 二．详细设计

### 1．程序结构

【从总体上描述程序的结构，文字或图示均可】

程序主文件main.c：包含主函数main

词法分析程序：lexical.c：包含词法分析的主要函数getsym()，以及许多必须的符号处理函数，同时包含打开测试程序的函数和输出词法分析结果的函数（初步的错误处理函数error暂时放在了而其中。）

语法分析程序：syntax

符号表管理：symtable

出错处理：error

语义分析与中间代码生成：

中间代码优化：optimize

目标代码生成：obj

### 2．类/方法/函数功能

【描述各类/方法或函数的功能，以及关键算法】

2.1 词法分析程序：

功能：从源程序的第一个字符开始顺序读字符，一次读一个，根据所读进的字符识别各类字符；对数字常数完成数字字符串到十进制数值的转换，并将其值保留；删去空格、换行、制表等字符；将识别出来的被空白字符分割的单词以二元式的形式输出。

关键算法：将各个种类的字符编码，根据文法中离终结符号较近的底层文法画出单词的状态图，再由这些状态图和词法分析程序算法构建词法分析程序。（算法示例《编译技术》P71）

2.2 语法分析程序：

功能：接受从词法分析程序中得来的单词，按照文法识别出各语法成分，同时进行语法检查，为语义分析和代码生成做准备。

关键算法：利用递归下降分析法，自顶向下地分析语法结构。约定：当调用某个分析子程序时，它所要分析的第一个符号已经读入了symbol。注意，在实现语法分析程序之前，要保证文法不带有左递归与回溯。

2.3 符号表管理

功能：收集、记录和使用源程序中的一些语法符号（标识符）的相关信息，已便检查语义的正确性并辅助生成正确的代码。

关键算法： c0语言是分程序结构语言，因此除了普通的插入删除等符号表操作，还需要具有定位与重定位的能力。考虑用结构指针实现，这样方便函数管理字表。具体设计在符号表管理方案中。

2.4 出错处理

功能：检查出各类错误，定位出错位置，返回错误信息，并跳过错误部分继续编译，尽可能找出所有错误。

2.5 语义分析与中间代码生成

功能：将与上下文有关的信息记录在符号表或其他数据区中，对表达式和赋值语句中的操作数进行一致性检查，类型不一致时要进行类型转换，分析由语法分析所识别的语句的意义，并作出相应的语义处理。

关键算法：在原语法的基础上进行改写，改写为属性翻译文法，再利用自顶向下语法指导翻译，对不同的语句进行不同的语义处理，包括：声明语句、表达式语句、赋值语句、控制语句、输入/输出语句、返回语句。

2.6 中间代码优化：

功能：为目标代码生成提供优化后的中间代码，提高目标程序的运行效率，获得更为紧凑简洁的目标代码。

关键算法：基本块划分算法、基本块的DAG图表示、通过DAG图消除局部公共子表达式、基本块的活跃变量数据流分析。

2.7 目标代码生成

功能：生成目标代码，同时在中间代码优化的基础上进行寄存器分配优化。

关键算法：全局寄存器的图着色算法

### 3．调用依赖关系

【说明各类之间的关系，方法/函数之间的调用关系】

语法分析程序调用词法分析程序取词并识别单词种类。

### 4．符号表管理方案

【说明符号表的数据结构、管理算法】

typedef struct symnode{

string name; //符号名

string kind; //var,const,func..

string type; //char int void

int value; //值

int addr; //地址

int level; //调用层次

bool arr; //是否为数组

int length; //数组的长度，或函数的参数个数

int depth; //该函数运行栈的深度

}\*symele;

管理算法：

C0结构简单，故在初步的符号表设计中省略了pascal-S符号表中的一些内容。

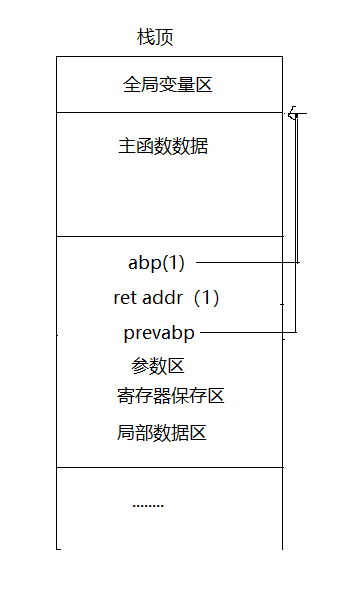
符号表以结构指针存在，符号表的插入删除都以指针操作进行。

### 5．存储分配方案

【说明运行时的存储组织及管理方案，运行栈结构】

由于c0语言中存在对函数的递归调用，因此需要使用动态存储分配进行管理，在进入一个程序模块（函数）时，在运行栈栈顶创建其活动记录。

运行栈结构：



### 6. 四元式设计\*

【对采用的四元式进行详细说明】（待修改）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | 操作符 | 操作数一 | 操作数二 | 结果 | 语义 |
| 0 | add | Src1 | Src2 | Obj | + |
| 1 | sub | Src1 | Src2 | Obj | - |
| 2 | mul | Src1 | Src2 | Obj | \* |
| 3 | div | Src1 | Src2 | Obj | / |
| 4 | big | Src1 | Src2 | Obj | 大于置一 |
| 5 | sml | Src1 | Src2 | Obj | 小于置一 |
| 6 | equ | Src1 | Src2 | Obj | 等于置一 |
| 7 | bge | Src1 | Src2 | Obj | 大于或等于置一 |
| 8 | sme | Src1 | Src2 | Obj | 小于或等于置一 |
| 9 | neq | Src1 | Src2 | Obj | 不等于置一 |
| 10 | asn | Src1 |  | Obj | 赋值 |
| 11 | zej | Src1 |  | Obj | 若为0则跳转 |
| 12 | rd | Src1 |  |  | 读入 |
| 13 | wr | Src1 |  |  | 写出 |
| 14 | lab | Src1 |  |  | 设置标签 |
| 15 | jump |  |  | Obj | 跳转 |
| 16 | func | Src1 |  |  | 函数头部 |
| 17 | para | Src1 | Src2 |  | 函数形参 |
| 18 | call | Src1 |  |  | 函数调用 |
| 19 | push | Src1 |  |  | 函数实参入栈 |
| 20 | ret | Src1 |  | Obj | 有返回值返回 |
| 21 | end | Src1 |  |  | 函数结束 |
| 22 | getar | Src1 | Src2 | Obj | 数组取值 |
| 23 | asnar | Src1 | Src2 | Obj | 数组赋值 |
| 24 | dint | Src1 |  |  | int声明 |
| 25 | dchar | Src1 |  |  | char声明 |
| 26 | scf | Src1 |  | Obj | 输入 |
| 27 | prts | Src1 |  |  | 输出字符串 |
| 28 | prta | Src1 |  |  | 输出变量 |

### 7. 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】

### 8. 出错处理

【说明出错处理方案、错误信息及含义】（待修改与补充）

|  |  |
| --- | --- |
| 编码 | 错误描述 |
| 1 | 单字符赋值中缺少‘ |
| 2 | 不合法的单字符 |
| 3 | 字符串缺少“ |
| 4 | 不等于缺少= |
| 5 | 不合法的字符串字符 |
| 6 | 标识符后缺少，或；或[ |
| 7 | 常量变量函数声明顺序出错 |
| 8 | 无主函数 |
| 9 | []未配对 |
| 10 | （）未配对 |
| 11 | {}未配对 |
| 12 | 无类型标识符关键字 |
| 13 | 不合法表达式 |
| 14 | 复合语句中的常量变量声明顺序错误 |
| 15 | 语句缺少; |
| 16 | 缺少do |
| 17 | 缺少while |
| 18 | 缺少switch |
| 19 | 缺少case |
| 20 | 语句缺少： |
| 21 | 参数个数不匹配 |
| 22 | 参数类型不匹配 |
| 23 | 读语句出错 |
| 24 | 写语句出错 |

## 三．操作说明

### 1．运行环境

【说明搭建运行环境的步骤】

Codeblocks13.12

### 2．操作步骤

【详细说明操作步骤】

（暂时）打开工程文件，运行，输入需要编译的源程序代码文件路径，正确输入后回车，程序开始编译。

## 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】