《编译技术》课程设计文 档

学号：15061169

姓名：彭杰奇

2018年 1月 13日

## 一．需求说明

### 1．文法说明

**获取的文法如下：**

＜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＞‘]’ )}

＜常量＞ ::= ＜整数＞|＜字符＞

＜类型标识符＞ ::= int | char

＜有返回值函数定义＞ ::= ＜声明头部＞‘(’＜参数＞‘)’ ‘{’＜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＞‘)’‘{’＜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＞ ::= ＜参数表＞

＜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＞}

＜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＞;|＜情况语句＞｜＜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＞

＜情况语句＞ ::= switch ‘(’＜表达式＞‘)’ ‘{’＜情况表＞[＜缺省＞] ‘}’

＜情况表＞ ::= ＜情况子语句＞{＜情况子语句＞}

＜情况子语句＞ ::= case＜常量＞：＜语句＞

＜缺省＞ ::= default : ＜语句＞

＜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）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

（5）数组的下标从0开始

**对文法的说明：**

文法是扩充C0文法；变量类型包括整型和字符型，不包括浮点型数字；循环语句为while循环，不包括do-while和for循环；条件语句为if-else条件判断；switch语句需要实现default；函数定义与调用部分，无参数时函数名之后不加括号；详细语法部分可见上述文法

文法中变量定义和函数定义存在FIRST集冲突的问题，但是并没有改写文法而是在实现编译器的过程中进行了判断处理

### 2．目标代码说明

生成的目标代码为能够在mars4.5上执行的mips 32位汇编码 根据扩充C0文法功能实现选取了需要使用的mips指令以及部分mars支持的伪指令

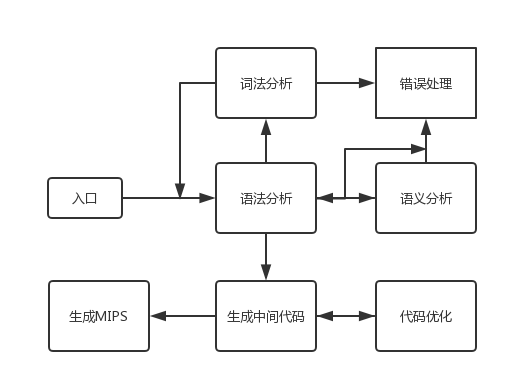
|  |  |
| --- | --- |
| 指令 | 含义 |
| add | 对两个数求和，结果保存在寄存器中 |
| addi | 对一个数和一个立即数求和，结果保存在寄存器中 |
| sub | 对两个数求差，结果保存在寄存器中 |
| subi | 对一个数和一个立即数求差，结果保存在寄存器中 |
| sll | 对一个数做逻辑左移（代替乘法） |
| mul | 对两个数求积，低32位保存到寄存器中 |
| div | 对两个数做除法，商保存在lo中，余数保存在hi中 |
| mflo | 取出lo寄存器中的值到寄存器中 |
| j | 跳转到目标标签 |
| jal | 跳转到目标标签并链接 |
| jr | 跳转到寄存器指向的目标地址 |
| beq | 如果两个数值相等就跳转 |
| bne | 如果两个数值不相等就跳转 |
| blt | 如果小于就跳转 |
| ble | 如果小于等于就跳转 |
| bgt | 如果大于就跳转 |
| bge | 如果大于等于就跳转 |
| li | 将一个立即数存入寄存器 |
| la | 取得目标地址，将该地址存入寄存器 |
| move | 将一个寄存器中的值存入另一个寄存器 |
| syscall | 系统调用，用来实现读和写 |
| sw | 将寄存器中的值存入目标地址 |
| lw | 从目标地址取出32位数存到寄存器中 |

### 3. 优化方案\*

见第二部分的优化说明

## 二．详细设计

### 1．程序结构



### 2．类/方法/函数功能

|  |  |
| --- | --- |
| 函数 | 功能 |
| void getch(); | //读下一个字符  从文件中读字符 |
| void searchSym(); | //查保留字表  读取的标识符需要查保留字表 |
| void getsym(); | //获取下一个symbol  根据单词状态图读取字符并确定symbol |
| void dec\_constant(); | //常量说明  处理常量说明语句并填写符号表 |
| void dec\_variableTofunction(); | //变量说明  处理变量说明语句并填写符号表，因为变量说明之后是函数定义，如果有函数定义的话不可避免会读到函数的头部，所以要转入函数定义 |
| void dec\_function(); | //函数定义  处理有返回值或无返回值的函数定义，识别声明头部和标识符并填写符号表和函数表 对于有参数的函数需要调用参数列表函数处理并填表，对于 { 后的内容调用复合语句处理函数处理 |
| void parameterlist(); | //参数列表  处理有参数的函数定义中的参数，识别类型和标识符并填写符号表 |
| void mainfunc(); | //主函数  处理程序入口函数，检查单词读取顺序为 void、main、(、)、{ ，之后调用复合语句处理函数处理，最后读取 }并检查后面还有没有多余字符 |
| void compound\_statement(); | //复合语句  处理复合语句，依次调用常量说明处理函数、变量说明处理函数、语句处理函数 |
| void statement\_list(); | //语句列  处理语句列，语句列由0个或多个语句组成，循环调用语句处理函数 |
| void statement(); | //语句  处理各类语句或语句列，根据symbol调用条件语句(if-else)处理函数，循环语句(while)处理函数，函数调用语句(有返回值或无返回值)处理函数，赋值语句处理函数，读(scanf)语句处理函数，写(printf)语句处理函数，情况语句(switch)处理函数，返回(return)语句处理函数或处理语句列 |
| void expresion(); | //表达式  处理表达式，调用项处理函数，循环处理项之间的加法运算 |
| void term(); | //项  处理项，调用因子处理函数，循环处理因子之间的乘法运算 |
| void factor(); | //因子  处理因子，需要查符号表，调用表达式处理函数或函数调用处理函数(有返回值函数) |
| void statement\_ifelse(); | //条件语句  处理条件语句，先处理if 和括号内的条件，生成跳转的label，调用语句处理函数处理if后的语句，处理else，无条件跳转生成跳转label，调用语句处理函数处理elese后面的语句 |
| void statement\_while(); | //循环语句  处理循环语句，先处理while和括号内的条件，生成跳转的label，调用语句处理函数处理while后的语句 |
| void statement\_switch(); | //情况语句  处理switch-case情况语句，生成各种情况的跳转label，调用语句处理函数处理case 和default后面的语句 |
| void functioncall(); | //函数调用  处理函数调用语句，读取函数标识符和参数，查符号表和函数参数表 |
| void statement\_assign(); | //赋值语句  处理赋值语句，调用表达式处理语句处理赋值语句的右部 |
| void statement\_read(); | //读语句  处理读语句，生成中间代码 |
| void statement\_write(); | //写语句  处理写语句，生成中间代码 |
| void statement\_return(); | //返回语句  处理return，生成跳转中间代码 |
| void search\_table(char name[NAME\_L],int field\_flag,int def\_flag); | //loc  在语法分析过程中查找符号表  field\_flag表示函数定义时查表  def\_flag表示常量和变量定义时查表 |
| void enter\_table(char name[NAME\_L], object o, types typ); | 填符号表  将name对应的标识符信息登录到符号表  o表示种类，typ表示类型 |
| void record\_quadruple(operator\_quadruple op, char operand1[OPERAND\_L], char operand2[OPERAND\_L], char result[OPERAND\_L]); | 生成四元式  op表示四元式操作符  operand1表示操作数1  operand2表示操作数2  result表示结果 |
| int search\_ifGlobal(char name[NAME\_L]); | 在生成mips时查找name对应的标识符是否在全局作用域中 |
| int search\_table2(char name[NAME\_L]); | 在生成misp时查找name对应的标识符在符号表中的位置 |
| void generate\_code(quadruple q\_i); | 根据四元式生成对应的mips |
| void generate\_mips(); | 生成全部mips汇编码 |
| void error(int errortype); | 输出错误信息  包括错误类型，错误发生行数和错误行代码以及错误位置指示 |
| void skip(); | 跳读 |
| print\_table(); | 输出符号表信息 |
| void print\_Qlist() | 输出生成的四元式 |
| print\_display() | 输出display表中信息 |
| print\_mips(); | 将生成的mips写入文件中 |

### 3．调用依赖关系

**词法分析**

|  |
| --- |
| getsym() |
| getch() |
| searchSym() |
| error(int errortype) |

**语法分析**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| dec\_constant() | dec\_variabletofunction() | dec\_function() | mainfunc() | function\_call() |
| getsym() | getsym() | getsym() | getsym() | getsym() |
| deal\_int\_constant() | deal\_int\_variable() | mainfunc() | getch() | expression() |
| deal\_char\_constant() | deal\_char\_variable() | parameterlist() |  | statement() |
| enter\_table() | enter\_table() | enter\_table() | enter\_table() | enter\_table() |
| error() | error() | error() | error() | error() |
| skip() | skip() | skip() | skip() | skip() |
| generate\_tempvar() | generate\_tempvar**()** | generate\_tempvar**()** | generate\_tempvar**()** | generate\_tempvar**()** |
| record\_quadruple() | record\_quadruple**()** | record\_quadruple**()** | record\_quadruple**()** | record\_quadruple**()** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| statement\_ifelse() | statement\_while() | statement\_switch() | statement\_read() | statement\_write() |
| expression() | expression() | expression() | expression() | expression() |
| statement() | statement() | statement() | search\_table() | search\_table() |
|  |  | deal\_case() |  |  |
|  |  | deal\_default() |  |  |
| error() | error() | error() | error() | error() |
| skip() | skip() | skip() | skip() | skip() |
| generate\_tempvar() | generate\_tempvar**()** | generate\_tempvar**()** | generate\_tempvar**()** | generate\_tempvar**()** |
| record\_quadruple() | record\_quadruple**()** | record\_quadruple**()** | record\_quadruple**()** | record\_quadruple**()** |

|  |  |  |  |
| --- | --- | --- | --- |
| statement\_assign() | statement\_return() | statement\_list() | compound\_statement() |
| expression() | expression() | statement() | dec\_constant() |
| statement() | getsym() |  | dec\_variabletofunction() |
| getsym() |  |  | statement\_list() |
|  |  |  |  |
| error() | error() |  | error() |
| skip() | skip() |  | skip() |
| generate\_tempvar() | generate\_tempvar**()** |  | generate\_tempvar**()** |
| record\_quadruple() | record\_quadruple**()** |  | record\_quadruple**()** |

|  |  |  |
| --- | --- | --- |
| expression() | term() | factor() |
| getsym() | getsym() | getsym() |
| term() | factor() | expression() |
| generate\_tempvar() | generate\_tempvar**()** | generate\_tempvar**()** |
| record\_quadruple() | record\_quadruple**()** | record\_quadruple**()** |

|  |  |
| --- | --- |
| statement\_list() | statement\_ifelse(), statement\_while(),statement\_switch(),  statement\_read(),statement\_write(),statement\_return(),  statement\_assign(),function\_call(),search\_table(),getsym(),  error(int errortype) ,skip(), |

mips

|  |  |
| --- | --- |
| generate\_code() | generate\_mips() |
| push\_stack() | generate\_code() |
| search\_ifGlobal() |  |
| get\_reg() |  |
| search\_table2() |  |

### 4．符号表管理方案

**数据结构：**

单条符号表中的记录:

typedef struct{

char name[NAME\_L]; //名称

object obj; //种类

types type; //类型

int val;//常量的值

int arr\_size;//数组大小

int para\_num;//函数参数个数

int link; //上一条记录在符号表中的位置

int offset;//mips需要 变量相对偏移地址（第几个变量）

}table;

符号表数组:

table tables[MAX\_T];

分程序索引表

int display[MAX\_T];

**管理算法：**

在变量定义时先查符号表，在同层作用域中有无已定义的同名变量或同名的本层函数名，如果有则报错，没有则登录符号表；在函数定义时查符号表全局作用域，看有无已定义的同名函数或变量，如果有则报错，没有则登录符号表；登录符号表的基本信息包括名字，种类和类型以及相对偏移地址，对于数组来说需要之后登录数组大小并修改偏移地址，对于函数来说需要之后登录函数的参数个数，对于常量来说需要之后登录常量的值

查找符号表可以利用link属性遍历以区别作用域，不能直接使用tables的index遍历查找

### 5．存储分配方案

**存储结构和管理方案:**

**运行栈**

调用函数时先向运行栈中保存上一个函数的fp寄存器中的值（运行栈地址），再保存ra寄存器中的值（返回地址），再顺序压入函数的参数，函数定义的变量

**全局变量**

$gp保存了全局变量保存字段的起始地址

**函数参数与保存现场**

$k0模拟栈，负责保存现场以及在调用函数时保存函数实参，进入函数后保存现场时的各个寄存器的值保存在$k0模拟的栈中

**运行栈结构**

|  |
| --- |
| 运行栈 |
| 函数f的局部变量 |
| 函数f的参数 |
| 函数f的 $fp |
| 返回地址 $ra |
| 上一个函数的 $fp |
| 全局变量 $gp |

### 6. 解释执行程序\*

无

### 7. 四元式设计\*

|  |  |
| --- | --- |
| 四元式结构 | 说明 |
| dec\_const int/char, name, val; | 声明常量 |
| dec\_val int/char, name, ; | 声明变量 |
| dec\_Arr int/char, name, arr\_size; | 声明数组变量 |
| dec\_func , , name; | 声明函数 |
| dec\_para int/char, name, ; | 声明参数（形参） |
| dec\_label , , lab; | 生成标签lab |
| op\_add arg1,arg2,res; | res = arg1+arg2 |
| op\_sub arg1,arg2,res; | res = arg1-arg2 |
| op\_mult arg1,arg2,res; | res = arg1\*arg2 |
| op\_div arg1,arg2,res; | res = arg1/arg2 |
| op\_assignNum , tempvar, val; | 将val赋给临时变量tempvar |
| op\_assignVal (tempvar1), name, tempvar; | 将临时变量tempvar赋给标识符name  arg1不为空则代表赋给name[tempvar1] |
| op\_getVal (tempvar1), name, tempvar; | 将标识符name的值赋给临时变量tempvar  arg1不为空则是使用name[tempvar1]赋值 |
| op\_getFuncVal , name, tempvar; | 将函数name返回值赋给临时变量tempvar |
| op\_para , , tempvar; | 将tempvar作为参数 |
| op\_j , , lab; | 无条件跳转至lab |
| op\_jal , , name; | 跳转到函数name |
| op\_beq arg1,arg2,lab; | 如果arg1==arg2 跳转至lab |
| op\_bne arg1,arg2,lab; | 如果arg1!=arg2 跳转至lab |
| op\_blt arg1,arg2,lab; | 如果arg1<arg2 跳转至lab |
| op\_ble arg1,arg2,lab; | 如果arg1<=arg2 跳转至lab |
| op\_bgt arg1,arg2,lab; | 如果arg1>arg2 跳转至lab |
| op\_bge arg1,arg2,lab; | 如果arg1>=arg2 跳转至lab |
| sys\_read , , name; | 读取变量name的值 |
| sys\_write 0/1/ , , tmepvar/str | 输出str或者输出tempvar |
| sys\_return , , temp\_var/; | 返回语句 返回值为空或者tempvar |
| sys\_exit , , ; | 强制退出 |

### 8. 目标代码生成方案\*

根据四元式生成对应的mips汇编码

**数据结构：**

临时寄存器以及使用过的寄存器由数组保存，生成的mips保存在数组中

**关键算法：**

对于加减乘除以及跳转的四元式在生成mips时和mips相对应；

对于函数调用四元式，生成目标代码时需要保存返回地址和上一个运行栈地址，保存现场，从内存中取出参数再存入运行栈内存中；

对于常量和变量声明四元式需要压栈操作；

对于参数四元式需要将实参值保存到内存中；

对于赋值和取数操作四元式，需要再查符号表获取偏移量来计算内存地址，再取数或者存数；

对于读写操作以及强制退出四元式需要系统调用；

### 9. 优化方案\*

**基本块内部的公共子表达式删除（DAG图）**

**数据结构：**

分别构造结构体表示DAG图的节点和节点表元素

typedef struct {

int parents[MAX\_L];

int left\_child;

int right\_child;

int index;

int parents\_index;

operator\_quadruple op;

int used;

}node;

typedef struct{

char var\_name[NAME\_L];

int var\_node;

int used\_node[MAX\_L];

}table\_node;

**算法：**

遍历四元式数组，按照四元式操作符类型（跳转操作符和lable操作符）确定基本块；

对于基本运算操作符（包括赋值）建立DAG图，根据启发式算法导出DAG图得到新的四元式序列；

对于一般的操作符需要根据DAG图替换四元式中的临时变量，并根据情况临时导出四元式序列并把自己写入新的四元式数组

**全局寄存器分配（引用计数或着色算法）**

**引用计数法**

对变量赋予不同的权值，使用次数更多的变量赋予更高的权值，对于一般的变量引用，每引用一次权值+1，对于处于循环体中的变量权值+10

因为最后时间不足，能力有限，优化没有完成

### 10. 出错处理

错误处理方案:

出错之后调用error(int errortype)函数，传入对应错误类型；error()函数会提示错误类型和错误发生位置（行号和列的位置）；

根据不同的错误类型会执行跳读代码或者停止编译或者忽略错误等操作；

|  |  |
| --- | --- |
| 错误类型 | 含义 |
| MISSING\_CHAR | 缺少右单引号 |
| MISSING\_STR | 缺少右双引号 |
| MISSING\_CASE | 缺少case |
| MISSING\_EQL | 缺少等号 |
| MISSING\_SEMICOLON | 缺少分号 |
| MISSING\_LPAREN | 缺少左括号 |
| MISSING\_RPAREN | 缺少右括号 |
| MISSING\_LBRACE | 缺少左中括号 |
| MISSING\_RBRACE | 缺少右中括号 |
| MISSING\_LBRACK | 缺少左大括号 |
| MISSING\_RBRACK | 缺少右大括号 |
| MISSING\_COLON | 缺少冒号 |
| MISSING\_COMMA | 缺少逗号 |
| MISSING\_MAIN | 缺少主函数 |
| OUT\_OF\_RANGE\_ARRAY | 数组越界 |
| OUT\_OF\_RANGE\_TABLE | 符号表越界 |
| OUT\_OF\_RANGE\_STR | 字符串长度越界 |
| OUT\_OF\_RANGE\_INT | 整数大小超过上限 |
| INVALID\_CHAR | 非法字符 |
| INVALID\_STR | 非法字符串 |
| INVALID\_NAME | 非法命名 |
| INVALID\_CASE | 非法情况子语句 |
| INVALID\_ASSIGN | 非法赋值 |
| INVALID\_FUNCTION\_DEC | 非法的函数声明 |
| INVALID\_VARIABLE\_DEC | 非法的变量声明 |
| INVALID\_FUNCTION\_HEAD | 非法函数头部 |
| INVALID\_READ | 非法读数据 |
| INVALID\_PARAMETER | 非法参数 |
| SHOULD\_BE\_INT\_OR\_CHAR | 需要是int或者char |
| SHOULD\_BE\_IDENTIFIER | 需要是标识符 |
| SHOULD\_BE\_INTEGER\_USINT | 需要是整数或者无符号整数 |
| SHOULD\_BE\_CHARACTER | 需要是字符 |
| SHOULD\_BE\_ASSIGNSYM | 需要是赋值符号 |
| SHOULD\_BE\_SCANF | 需要是scanf |
| SHOULD\_BE\_PRINTF | 需要是printf |
| SHOULD\_BE\_IFSYM | 需要是if |
| SHOULD\_BE\_ELSESYM | 需要是else |
| SHOULD\_BE\_WHILESYM | 需要是while |
| SHOULD\_BE\_SWITCHSYM | 需要是switch |
| SHOULD\_BE\_PARAMETER | 需要是参数 |
| SHOULD\_BE\_ARRAY | 需要是数组 |
| CANNOT\_BE\_VOID\_FUNCTION | 不能是无返回值函数 |
| CANNOT\_BE\_ARRAY | 不能是数组 |
| EMPTY\_READ | 读入为空 |
| DUPLICATE\_DEF | 重复定义 |
| NOT\_DEFINE | 没有定义 |
| RETURN\_ERROR | 返回值错误 |
| EXTRA\_CHARACTER\_AFTER\_MAIN | 在main函数之后还有多余代码 |
| DUPLICATE\_CASE | case后的值重复 |

## 三．操作说明

### 1．运行环境

CodeBlocks 16.01

Mars 4.5

### 2．操作步骤

使用CodeBlocks打开编译程序源代码，编译运行，输入待编译文件的相对地址；

程序运行会在编译程序源代码目录下生成15061169\_mips.asm文件

使用Mars4.5打开15061169\_mips.asm文件，设置Mars的Memory Configuration为default 并且不勾选delayed branching；运行程序

## 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】

测试程序1

const char const\_c = 'a';

const int sss=5,ddd=-5;

int x,y,r;

int arr[10];

char str[5];

void test1{

const int a = 2;

const char b = 'a';

int f0,f1;

int farr[21];

f0 = a; printf(" f0: ",f0);

x = f0; printf(" x: ",x);

f1 = x+f0\*2; printf(" f1: ",f1);

y = f1\*f0/(x+3); printf(" y: ",y);

arr[1]=1;

arr[2]=y;

arr[3]=x\*(a-3);

arr[4]=b\*3+2;

arr[5]='d'-3;

farr[1]='b'; printf(" farr[1]: ",farr[1]);

farr[2]='a'+'b'; printf(" farr[2]: ",farr[2]);

farr[3]='a'\*3; printf(" farr[3]: ",farr[3]);

f0 = arr[3]; printf(" f0: ",f0);

f1 = farr[2]; printf(" f1: ",f1);

f0 = arr[2]+f0; printf(" f0: ",f0);

f1 = arr[3]-farr[2]; printf(" f1: ",f1);

farr[1] = 5;

printf(" arr: ");

while(farr[1]){

printf(" ",arr[farr[1]]);

farr[1] = farr[1]-1;

}

}

void main(){

test1;

}

**测试结果**

**输入：无**

**输出：**

**f0: 2 x: 2 f1: 6 y: 2 farr[1]: 98 farr[2]: 195 farr[3]: 291 f0: -2 f1: 195 f0: 0 f1: -197 arr: 97 293 -2 2 1**

测试程序2

void main(){}

**测试结果**

**输入：无**

**输出：无**

测试程序3

const int a = 1;

const int b = 2;

const char c = 'c';

int x,y;

char sx,sy;

void fun7{

int x,y,z;

char c1,c2;

scanf(x);

scanf(c1,c2);

scanf(y,z);

}

void fun6(int a1,int a2,int a3,int a4,int a5){

int arr[5];

int intarr[10];

int i;

i=0;

while(i<10){

intarr[i]=i+'a';

i=i+1;

}

arr[0]=a1;arr[1]=a2;arr[2]=a3;arr[3]=a4;arr[4]=a5;

printf("arr[1]=",arr[1]);

printf("a2=",a2);

switch(arr[1]){

case 1:printf("arr[2]=",arr[2]);

case 2:printf("arr[3]=",arr[3]);

case 4:;

case 3:printf("arr[4]=",arr[4]);

default:printf(" default ");

}

}

void fun5{

const char ch = 'r';

char charr[10];

int f0,f1,f2;

int i,j;

i = 10;

while(i>0){

printf("i= ",i);

printf(", ");

i = i-1;

}

while(i<5){

j = 3;

while(j){

printf("i=",i);

printf(", j=",j);

printf(", ");

if(i\*j>5) printf("fun5 i\*j=",i\*j);

else;

printf(", ");

j = j-1;

}

i = i+1;

}

}

void fun4{

const char ch = 'r';

char charr[10];

int f0,f1,f2;

f0 = 1;

if(f0>2){

f0 = 3;

printf("fun4 f0>2 ");

}else{

f0 = 0;

printf("fun4 f0<=2 ");

}

if(f0){

printf("fun4 f0!=0 ");

}

else{

f1 = 3;

printf("fun4 f0==0 ");

}

if(f1>=3){

f2 = 2;

printf("fun4 f2==2 ");

if(f2!=2){

}else{

f2 = ch;

printf("fun4 f2 = ",f2);

printf(", ");

printf("fun4 ch = ",ch);

printf(", ");

}

}

else{}

}

int fun3(int f1,int f2){

const int x = 2;

return(f1\*f2-2);

}

int fun2(int f1,int f2,int f3){

printf(", fun2-f1=",f1);

printf(", fun2-f2=",f2);

printf(", fun2-f3=",f3);

return((f1+f2)/f3-fun3(f1-f2,f3));

}

void fun(int t,char tc){

const int x = 3;

int a;

int f0,f1,f2,f3;

f2 = t;

printf(f2);

printf(", ");

printf(tc);

printf(", ");

a = x+b;

f0 = a+x;

f1 = a+f0;

printf(a);

printf(", ");

printf(f0);

printf(", ");

printf(f1);

y = a;

f2 = y\*b+t;

f3 = f2/2-2;

printf(", ",f2);

printf(", ",f3);

printf(", fun2 = ",fun2(f2,f3,3));

}

void main(){

int t1,t2;

int arr[10];

fun7;

fun5;

fun4;

t2 = 3+b;

arr[t2] = 4;

printf(arr[5]);

printf(", ");

fun(3,'f');

t1= y+a\*t2;

printf(", ",t1);

}

**测试结果**

**输入：**

**2**

**ab3**

**4**

**输出：**

**i= 10, i= 9, i= 8, i= 7, i= 6, i= 5, i= 4, i= 3, i= 2, i= 1, i=0, j=3, , i=0, j=2, , i=0, j=1, , i=1, j=3, , i=1, j=2, , i=1, j=1, , i=2, j=3, fun5 i\*j=6, i=2, j=2, , i=2, j=1, , i=3, j=3, fun5 i\*j=9, i=3, j=2, fun5 i\*j=6, i=3, j=1, , i=4, j=3, fun5 i\*j=12, i=4, j=2, fun5 i\*j=8, i=4, j=1, , fun4 f0<=2 fun4 f0==0 fun4 f2==2 fun4 f2 = 114, fun4 ch = r, 4, 3, f, 5, 8, 13, 13, 4, fun2 = , fun2-f1=13, fun2-f2=4, fun2-f3=3-20, 10**

**输入：**

**12**

**rt-5**

**43**

**输出：**

**i= 10, i= 9, i= 8, i= 7, i= 6, i= 5, i= 4, i= 3, i= 2, i= 1, i=0, j=3, , i=0, j=2, , i=0, j=1, , i=1, j=3, , i=1, j=2, , i=1, j=1, , i=2, j=3, fun5 i\*j=6, i=2, j=2, , i=2, j=1, , i=3, j=3, fun5 i\*j=9, i=3, j=2, fun5 i\*j=6, i=3, j=1, , i=4, j=3, fun5 i\*j=12, i=4, j=2, fun5 i\*j=8, i=4, j=1, , fun4 f0<=2 fun4 f0==0 fun4 f2==2 fun4 f2 = 114, fun4 ch = r, 4, 3, f, 5, 8, 13, 13, 4, fun2 = , fun2-f1=13, fun2-f2=4, fun2-f3=3-20, 10**

测试程序4

const int c1 = 1;

const int c2 = 3;

int fun2(int a,int b,int x){

switch(x){

case 1:;

case 2:{

switch(c2){

case 1:printf(" c2=1 ");

case 2:printf(" c2=2 ");

default:printf(" c2=? ");

}

}

case 3:{printf(" x= ",x);printf(" ");}

}

return (a\*c2-b\*c1);

}

int fun1(int a,int b,int c){

return (a\*2/b+c);

}

void main(){

printf(fun1(6,fun2(3,2,2),3));

printf(fun1(fun2(1,fun1(2,3,4),3),2,2));

}

**测试结果**

**输入：无**

**输出： c2=? 4 x= 3 0**

测试程序5

const int max\_l = 10;

const int min\_l = -1;

const char const\_c = 'a';

const int sss=5,ddd=-5;

int x,y,r;

int arr[10];

char str[5];

int p,q;

int Fibonacci(int n){

int f1,f2;

if(n==1)

return(1);

else{

if(n<=0)

return(0);

else{

f1 = Fibonacci(n-1);

f2 = Fibonacci(n-2);

return(f1+f2);

}

}

}

int judgemax(int a,int b){

if(a<b)

return(0);

else

return(1);

}

int getnum{

if(x>0)

return(+1\*sss\*2-6);

else

return(ddd\*2+8);

}

void input{

int p;char q;

int f1,f2,f3;

char f4,f5,f6;

int fiarr[21];

char fcarr[21];

;;;{};{}

scanf(p,q);

scanf(f1,f2);

scanf(f3);

scanf(f4,f5);

scanf(f6);

printf(" p: ",p);printf(" q: ",q);

printf(" f: ",f1);printf(",",f2);printf(",",f3);printf(",",f4);printf(",",f5);printf(",",f6);

}

void output(int a,char c){

const int const\_i = 1;

int f,t;

f = getnum;

t = judgemax(f,const\_i);

if(t){

printf(" a=",a);

printf(",");

printf(" f=",f);

printf("; ");

}

else{

printf(" c=",c);

printf(",");

printf(" f=",f);

printf("; ");

}

}

void test1{

const int a = 2;

const char b = 'a';

int f0,f1;

int farr[21];

f0 = a; printf(" f0: ",f0);

x = f0; printf(" x: ",x);

f1 = x+f0\*2; printf(" f1: ",f1);

y = f1\*f0/(x+3); printf(" y: ",y);

arr[1]=1;

arr[2]=y;

arr[3]=x\*(a-3);

arr[4]=b\*3+2;

arr[5]='d'-3;

farr[1]='b'; printf(" farr[1]: ",farr[1]);

farr[2]='a'+'b'; printf(" farr[2]: ",farr[2]);

farr[3]='a'\*3; printf(" farr[3]: ",farr[3]);

f0 = arr[3]; printf(" f0: ",f0);

f1 = farr[2]; printf(" f1: ",f1);

f0 = arr[2]+f0; printf(" f0: ",f0);

f1 = arr[3]-farr[2]; printf(" f1: ",f1);

farr[1] = 5;

printf(" arr: ");

while(farr[1]){

printf(" ",arr[farr[1]]);

farr[1] = farr[1]-1;

}

}

char test2(int p){

int test;

if(p>5) p = 5;

else;

while(p<10){

p = p+2;

test = p;

while(test>0){

test = test-3;

}

}

switch(x){

case 1:x = 2;

case 2:{x = 3;}

case 3:;

case 4:x = x+5;

default:;

}

switch(test){

case 2:{test=-2;x=+2;}

}

printf(" test = ",test);

printf(" x = ",x);

return('x');

}

void fulltest{

const int h = +1;

int i,j;

char ch;

printf(" x = ",x);

i=0;

while(i<max\_l){

arr[i] = i;

if(i<5){

ch=str[i];

}else{}

if(i<3){

x = x\*(i+1);

}

else{

x = x-i;

}

output(arr[i],ch);

i = i+1;

}

j=+3;

if(x>0) y = x+1;

else y = -x+1;

ch='a';

printf(" x = ",x);

while(j>min\_l){

switch(ch){

case 'a': {

ch='b';

y=y/j;

}

case 'b': {

ch='c';

y=y+j;

}

default: y=y-j;

}

printf(" y = ",y);

j = j-1;

}

printf(" h = ",h);

}

void main(){

const int ff = -5;

int i;

char ch;

printf(" please input x: ");

scanf(x);

r = x;

if(x<6)

printf("Fibonacci(x) = ",Fibonacci(x));

else

printf("Fibonacci(5) = ",Fibonacci(5));

printf(" please input str[len==5]: ");

i = 0; while(i<5){

scanf(ch);

str[i] = ch;

i = i+1;

}

printf(" test function input(p,q,f1,f2,f3,f4,f5,f6): ");

input;

printf(" test function test1, test2, fulltest: ");

test1;

q = 3; ch = test2(q);

fulltest;

printf(" last = ",getnum+Fibonacci(-ff+x));

return;

}

**测试结果**

**输入：**

**（please input x:）**

**3**

**（please input str[len==5]:）**

**abdse**

**（test function input(p,q,f1,f2,f3,f4,f5,f6):）**

**2**

**a3**

**4**

**5**

**art**

**输出：**

**Fibonacci(x) = 2 p: 2 q: a f: 3,4,5,a,r,t test function test1, test2, fulltest: f0: 2 x: 2 f1: 6 y: 2 farr[1]: 98 farr[2]: 195 farr[3]: 291 f0: -2 f1: 195 f0: 0 f1: -197 arr: 97 293 -2 2 1 test = -1 x = 3 x = 3 a=0, f=4; a=1, f=4; a=2, f=4; a=3, f=4; a=4, f=4; a=5, f=4; c=e, f=-2; c=e, f=-2; c=e, f=-2; c=e, f=-2; x = -24 y = 8 y = 10 y = 9 y = 9 h = 1 last = -2**

**输入：**

**（please input x:）**

**8**

**（please input str[len==5]:）**

**rusdr**

**（test function input(p,q,f1,f2,f3,f4,f5,f6):）**

**-12**

**f32**

**12**

**-5**

**sdf**

**输出：**

**Fibonacci(5) = 5 p: -12 q: f f: 32,12,-5,s,d,f test function test1, test2, fulltest: f0: 2 x: 2 f1: 6 y: 2 farr[1]: 98 farr[2]: 195 farr[3]: 291 f0: -2 f1: 195 f0: 0 f1: -197 arr: 97 293 -2 2 1 test = -1 x = 3 x = 3 a=0, f=4; a=1, f=4; a=2, f=4; a=3, f=4; a=4, f=4; a=5, f=4; c=r, f=-2; c=r, f=-2; c=r, f=-2; c=r, f=-2; x = -24 y = 8 y = 10 y = 9 y = 9 h = 1 last = -2**

错误测试程序1

const int a = 1;

const char c = 'a';

const char d;

char ch;

int x = 1;

int y;

int b;

int arr[10];

char str[5];

void s(int a){

const int xx=1;

const char cc='a';

int xxx,yyy;

scanf(xxx);

printf(yyy);

printf("nice");

printf("okok",y);

return(253);

x = 12;

y = -(x+2);

if(x<2){

a=3;

}else{

b = 4;

}

}

int haha{

int i;

i=1;

while(a>2){

i=i+1;

}

switch(x){

case 1:arr[1]=2;

case 'a':str[2]='a';

default:arr[2]=3;

}

}

void main(){

s(a);

haha;

}

**测试结果**

**编译未通过**

**Error[28]: there should be an assignment [=]**

**line[3]: const char d;**

**^**

**Error[31]: invalid variable declaration [invalid]**

**line[5]: int x =**

**^**

**Error[43]: function does not have return value [no return]**

**line[19]: return(253);**

**^**

**Error[26]: there should be an integer or unsigned integer [+1]**

**line[36]: case 'a':**

**^**

错误测试程序2

const int ca = 1;

const char cc = 'c';

void fun7{

int x,y,z;

char c1,c2;

scanf(x);

scanf(y,z);

printf(x);

}

void fun8(){

if('a'){

printf("it is 'a'");

}else{}

}

void main(){

int a;

char c,c2;

printf("hello ok");

printf(" world");

fun8;

return;

fun7;

a = c+c2;

c = cc;

c = cc+c2;

c = c+c2;

c = c+1;

}

**测试结果**

**编译未通过**

**Error[24]: there should be symbol int or char [int]**

**line[11]: void fun8(){**

**^**

**Error[27]: there should be a character ['a']**

**line[27]: c = cc+c2;**

**^**

**Error[27]: there should be a character ['a']**

**line[28]: c = c+c2;**

**^**

**Error[27]: there should be a character ['a']**

**line[29]: c = c+1;**

**^**

错误测试程序3

void fun6(int a1,int a2,int a3,int a4,int a5){

int arr[5];

arr[0]=a1;arr[1]=a2;arr[2]=a3;arr[3]=a4;arr[4]=a5;

switch(arr[1]){

case 1:printf("arr[2]=",arr[2]);

case 2:printf("arr[3]=",arr[3]);

case 4:;

case 3:printf("arr[4]=",arr[4]);

default:printf(" default ");

}

;;;;;;;{};;{;;};;;

}

void fun7{

int a,b;

a = 1;

if(a<2){

return;

}else{

}

}

void main(){

fun7;

fun6(1,2,3,4);

}

**测试结果**

**编译未通过**

**Error[47]: there should be parameter [par]**

**line[25]: fun6(1,2,3,4);**

**^**

错误测试程序4

int x,y;

void fun{}

void main(){}

int a;

**测试结果**

**编译未通过**

**Error[33]: there are extra characters after main function [extra]**

**line[4]: i**

**^**

错误测试程序5

const int a =+2;

const int b=-3;

int h;

char arr[12];

int fun(int s){

const int a=3;

int x;

x = 1;

a = 4;

if(x<1){

x=2;

}else{

x=3;

}

return(x);

}

int fun1(int s){

int x;

x = 2;

switch(x){

case 1:x = 3;

case 2:{x=4;}

case 3:;

case 4:x=x+5;

default:;

}

return(x);

}

void main(){

const int a=3;

int y;

x = 4;

arr[fun(1)\*1]=1;

y = arr[arr[fun(1)];

y = a+fun(1);

}

**测试结果**

**编译未通过**

**Error[23]: cannot assign constants [no=]**

**line[11]: a**

**^**

**Error[48]: not define [no def]**

**line[35]: x**

**^**

**Error[27]: there should be a character ['a']**

**line[36]: arr[fun(1)\*1]=1;**

**^**

**Error[9]: miss right brack []]**

**line[37]: y = arr[arr[fun(1)];**

**^**

### 2．测试结果分析

测试程序1 测试了各种情况下的变量以及数组元素的相互赋值，并使用数组元素作为while循环的循环条件输出数组元素

测试程序2 测试了最简单结构的程序是否可以编译

测试程序3 测试了基本常量和变量定义，各种类型的函数定义，while循环以及循环嵌套，if-else条件语句以及if-else的嵌套，switch语句以及多种case情况和有无default的情况，输入和输出语句，空语句测试，变量定义域的测试

测试程序4 测试了有返回值函数调用作为函数参数传参，复杂的嵌套调用，switch语句的嵌套调用

测试程序5 测试了函数的递归调用，并且测试了复杂的输入，对以上所有测试的语法分支都再进行了全面的测试

错误测试程序1 测试了常量和变量声明时出现的错误，以及无返回值函数出现带返回值的return语句和case后的常量与switch中的不匹配的错误 总共4处错误

错误测试程序2 测试了无参数函数定义时有多余括号的错误，并且测试了对字符型变量赋值不为字符型的错误（根据论坛中老师的说法自行定义）总共4处错误

错误测试程序3 测试了函数调用时传参的错误 总共1处错误

错误测试程序4 测试了在main函数结束之后还存在代码段的错误 总共1处错误

错误测试程序5 测试了对常量赋值，变量未定义便使用，向字符型变量赋值整数，括号匹配不正确的错误 总共4处错误

## 五．总结感想

首先必须要说，编译课设对我理解编译理论课上学习到的知识帮助非常大：比如说语法制导翻译以及语义分析部分在自己亲自去设计编译器和写代码之前认识是比较模糊的，当自己完成和符号表相关的函数，设计了四元式，在语法分析的过程中插入这些函数时对这一部分也就清晰了；再比如对运行栈的理解，在理论课上觉得并没有什么太难的地方，但是在实际实现编译器的时候，一开始实在对运行栈有些理解不能，直到生成mips汇编代码的时候才回过神来，之后生成mips时对运行栈的设计和内存管理操作的思考都加深了我对运行栈这一部分的理解；当然更不用说词法分析语法分析等等基本部分了

然后要说编译课设加深了我对计算机语言的认识，完成将扩充C0文法定义的语言分析并翻译为中间代码再生成汇编码的过程，再加上编译课最后一节课讲的自编译与自展以及之前学过的优化，让我对一段程序从编译到执行的“生命过程”更感兴趣，脑海中也总是会对许多类C语言的编译有大概的映像；再结合自己写过的代码运行情况，对一些编译信息和运行快慢有不同的理解

同时编译大作业的完成也让我又重新熟悉了许久未使用的C语言和C++库，让我对更加生疏的mips操作熟练起来；再一个要感慨的是编译课设的作业进度设计还是比较合理的，对我们按时完成最终作业有很大帮助，不仅是合理安排时间达到进度，课设每周任务的区分也能帮助我们分别设计编译器的各部分；另外特别感谢的是老师提供的共享测试集的方案，也感谢许多无私奉献提供和不断改进测试程序的同学，他们的测试程序还是帮助自己发现了不少bug的，另外助教在检查的时候也非常有耐心，积极帮助我们分析问题所在，让我们能够加深理解，一路前进

美中不足的就是优化部分因为最后几周的各种大作业DDL和好几门考试而被挤压没有时间，尝试写DAG图和引用计数时没有完成，另外C0文法其实是有FIRST集冲突的，不过也没有去改写文法，只是在实现编译器的时候判断了；这个编译器又是自己完成的一个比较大的工程，自己也做了一点结构上的调整尽力去使结构清晰合理，其中付出的努力和这段经历对自己之后学习和写代码都是有很大帮助的；最后希望编译课程越来越好，也希望同学们能学到更多知识