**《编译技术》课程设计**

**需求说明**

# 引言

## 编写目的

本需求说明旨在对整个c0编译器做一说明，是开发编译器的依据。

## 参考文献

[1]Compilers Principles,Techniques, & Tools. V.Aho, S.Lam, Ravi Sethi, D.Ullman. 北京,机械工业出版社(第2版)

[2]编译原理及编译程序构造. 高仲仪, 金茂忠. 北京,北京航空航天大学出版社(第2版)

# 需求概述

## 目标

实现一个扩充C0文法的编译器，具体要求如下：

文法： 扩充C0文法  
优化： 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)

其它优化自选

代码生成时合理利用临时寄存器(临时寄存器池)，并能生成较高质量的目标代码

中间代码：四元式

目标码： X86汇编

## 功能

以文本文件的形式输入一个C语言的源程序，编译并生成该源程序的一个X86汇编码的文本文件，该文件可以在masm32上成功运行并得到正确结果。

# 需求规定

## 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＞   ::=  "｛<合法字符>｝"  
                             //字符串中可以出现所有合法的可打印字符集中的字符  
＜程序＞    ::= ［＜常量说明部分＞］［＜变量说明部分＞］{＜有返回值函数定义部分＞|＜无返回值函数定义部分＞}＜主函数＞  
＜常量说明部分＞ ::=  const＜常量定义＞;{ ＜常量定义＞;}  
＜常量定义＞   ::=   int＜标识符＞＝＜整数＞{,＜标识符＞＝＜整数＞}  
                                    | float＜标识符＞＝＜实数＞{,＜标识符＞＝＜实数＞}  
                                    | char＜标识符＞＝＜字符＞{,＜标识符＞＝＜字符＞}  
＜整数＞    ::= ［＋｜－］＜非零数字＞｛＜数字＞｝｜０  
＜实数＞    ::= ［＋｜－］<整数>[.<整数>]  
＜标识符＞    ::=  ＜字母＞｛＜字母＞｜＜数字＞｝  
＜声明头部＞   ::=  int＜标识符＞ |float ＜标识符＞|char＜标识符＞  
＜变量说明部分＞  ::= <变量定义>;{<变量定义>;}  
＜变量定义＞  ::= ＜类型标识符＞＜标识符＞{,＜标识符＞}  
＜类型标识符＞      ::=  int | float | char  
＜有返回值函数定义部分＞  ::=  ＜声明头部＞‘(’ ＜参数＞ ‘)’  ‘{’ ＜复合语句＞ ‘}’  
＜无返回值函数定义部分＞  ::= void<标识符>‘(’ <参数> ‘)’  ‘{’  <复合语句> ‘}’  
＜复合语句＞   ::=  ［＜常量说明部分＞］［＜变量说明部分＞］＜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＞}| <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＞｜‘(’＜表达式＞‘)’｜＜整数＞｜＜有返回值函数调用语句＞|<实数>|＜字符＞           
＜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＞;｜;｜＜返回语句＞；  
＜赋值语句＞   ::=  ＜标识符＞＝＜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＞    ::=  ＜非零数字＞｛＜数字＞｝    
＜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＞｛＜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

## 语法图

程序

常量说明

常量说明

有返回值函数定义

无返回值函数定义

主函数

主函数

void

main

(

)

复合语句

{

}

常量说明

const

char

int

float

id

character

id

id

integer

real

=

=

=

,

,

,

;

变量说明

char/int/float

id

,

;

有返回值函数定义

char/int/float

id

;

;

参数表

复合语句

无返回值函数定义

void

id

;

;

参数表

复合语句

复合语句

常量说明

常量说明

语句列

语句

if

条件

语句

else

(

)

语句

while

条件

(

)

语句

for

id

(

;

语句

=

表达式

条件

;

id

=

+

-

integer

)

id

=

表达式

(

值参数表

)

scanf

printf

return

(

id

)

,

(

表达式

(

string

)

,

表达式

)

(

语句列

)

;

;

;

;

;

语句列

语句

表达式

项

+

-

+

-

项

因子

\*

/

因子

id

(

character

integer

real

表达式

)

(

)

值参数表

参数表

char/int/float

id

,

值参数表

表达式

,