编 译 实 验

一. 概述

1. 源语言：求n！的极小语言

（文法见后）

〈程序〉→〈分程序〉

〈分程序〉→begin〈说明语句表〉;〈执行语句表〉end

〈说明语句表〉→〈说明语句〉|〈说明语句表〉;〈说明语句〉

〈说明语句〉→〈变量说明〉|〈函数说明〉

〈变量说明〉→integer〈变量〉

〈变量〉→〈标识符〉

〈标识符〉→〈字母〉|〈标识符〉〈字母〉| 〈标识符〉〈数字〉

〈字母〉→a|b|c|d|e|f|g|h|i|j|k|l|m|n|o |p
|q |r|s|t|u|v|w|x|y|z

〈数字〉→0|1|2|3|4|5|6|7|8|9

〈函数说明〉→integer function〈标识符〉(〈参数〉)；〈函数体
〉

〈参数〉→〈变量〉

〈函数体〉→begin〈说明语句表〉;〈执行语句表〉end

〈执行语句表〉→〈执行语句〉|〈执行语句表〉;〈执行语句〉

〈执行语句〉→〈读语句〉|〈写语句〉|〈赋值语句〉|〈条件语句〉

〈读语句〉→read(〈变量〉)

〈写语句〉→write(〈变量〉)

〈赋值语句〉→〈变量〉:=〈算术表达式〉

〈算术表达式〉→〈算术表达式〉-〈项〉|〈项〉

〈项〉→〈项〉*〈因子〉|〈因子〉

〈因子〉→〈变量〉|〈常数〉|〈函数调用〉

〈常数〉→〈无符号整数〉

〈无符号整数〉→〈数字〉|〈无符号整数〉〈数字〉

〈条件语句〉→if〈条件表达式〉then〈执行语句〉else
〈执行语句〉

〈条件表达式〉→〈算术表达式〉〈关系运算符〉〈算术表达式〉

〈关系运算符〉→〈 | 〈= | 〉 | 〉= | = | 〈〉

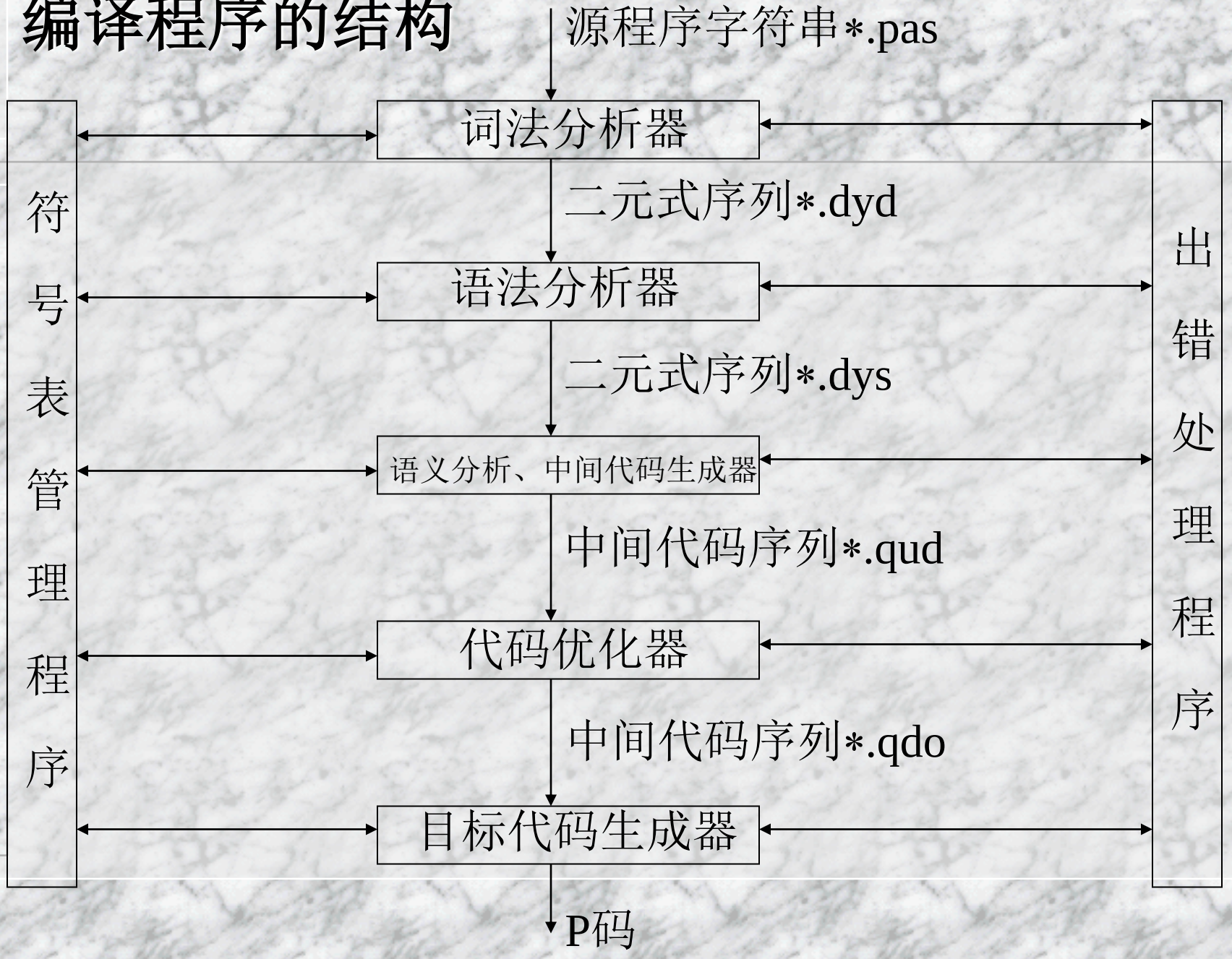
2. 工具语言: Pascal、C或其它

3. 语法分析采用递归下降分析法

4. 目标代码: P码

5. 编译程序的结构

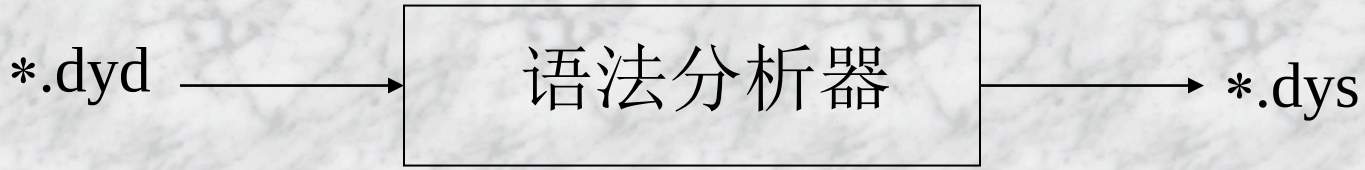
编译程序的结构


源程序字符串*.pas
词法分析器
二元式序列*.dyd
语法分析器
二元式序列*.dys
语义分析、中间代码生成器
中间代码序列*.qud
代码优化器
中间代码序列*.qdo
目标代码生成器
P码
符号表管理程序
出错处理程序

二. 各阶段的输入输出

1. 词法分析

2. 语法分析



同时, 产生文本文件*.var、 *.pro

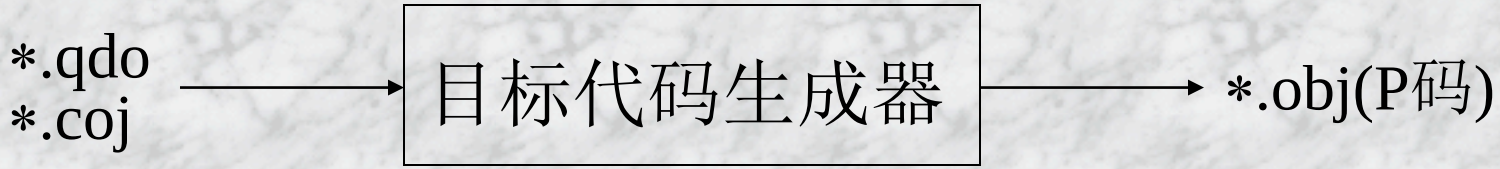
3. 语义分析

同时, 产生二进制文件*.ooj、varfil、profil

4. 优化

同时, 产生二进制文件*.coj

5. 目标代码生成



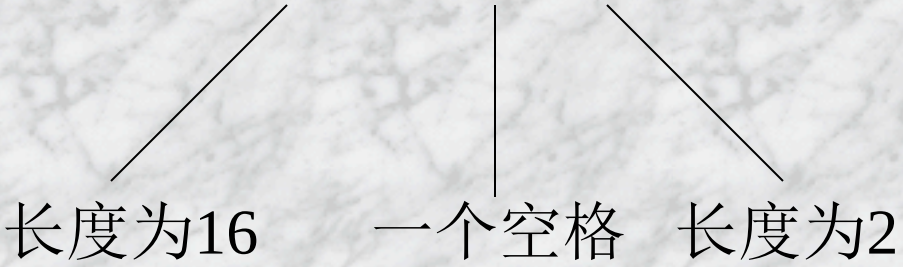
同时, 产生二进制文件pf(目标指令)

三. 数据结构

1. 二元式文件*.dyd

(1)二元式形式:

单词符号␣种别



单词符号与种别对照表

单词符号	种别	单词符号	种别	单词符号	种别
begin	1	end	2	integer	3
if	4	then	5	else	6
function	7	read	8	write	9
标识符	10	常数	11	=	12
<>	13	<=	14	<	15
>=	16	>	17	-	18
*	19	:=	20	(	21
)	22	;	23		

(2)每行后加一

“␣␣␣...␣EOLN␣24”

(3)文件结尾加

“␣␣␣...␣EOF␣25”

2. 错误信息文件*.err

(1)错误信息格式

***LINE:行号␣␣错误性质

(2)注意: 进入每一阶段, 首先打开*.err, 如果无错误, 则*.err为空。

3. *.dys 同*.dyd

4. 变量名表

变量名vname: char(16)

所属过程vproc:char(16)

分类vkind: 0..1(0—变量、1—形参)

变量类型vtype: types

变量层次vlev: int

变量在变量表中的位置vadr: int(相对第一个变量而言)

types=(ints)

5. *过程名表*

*过程名***pname: char(16)**

*过程类型***ptype: types**

*过程层次***plev: int**

*第一个变量在变量表中的位置***fadr: int**

*最后一个变量在变量表中的位置***ladr: int**

6. 四元式表

(oprd, op1, op2, result)

oprd——整数码

op1——第一操作数

op2——第二操作数

result——结果

op1、op2、result可用“值/地址”表示

7. 目标代码——P码

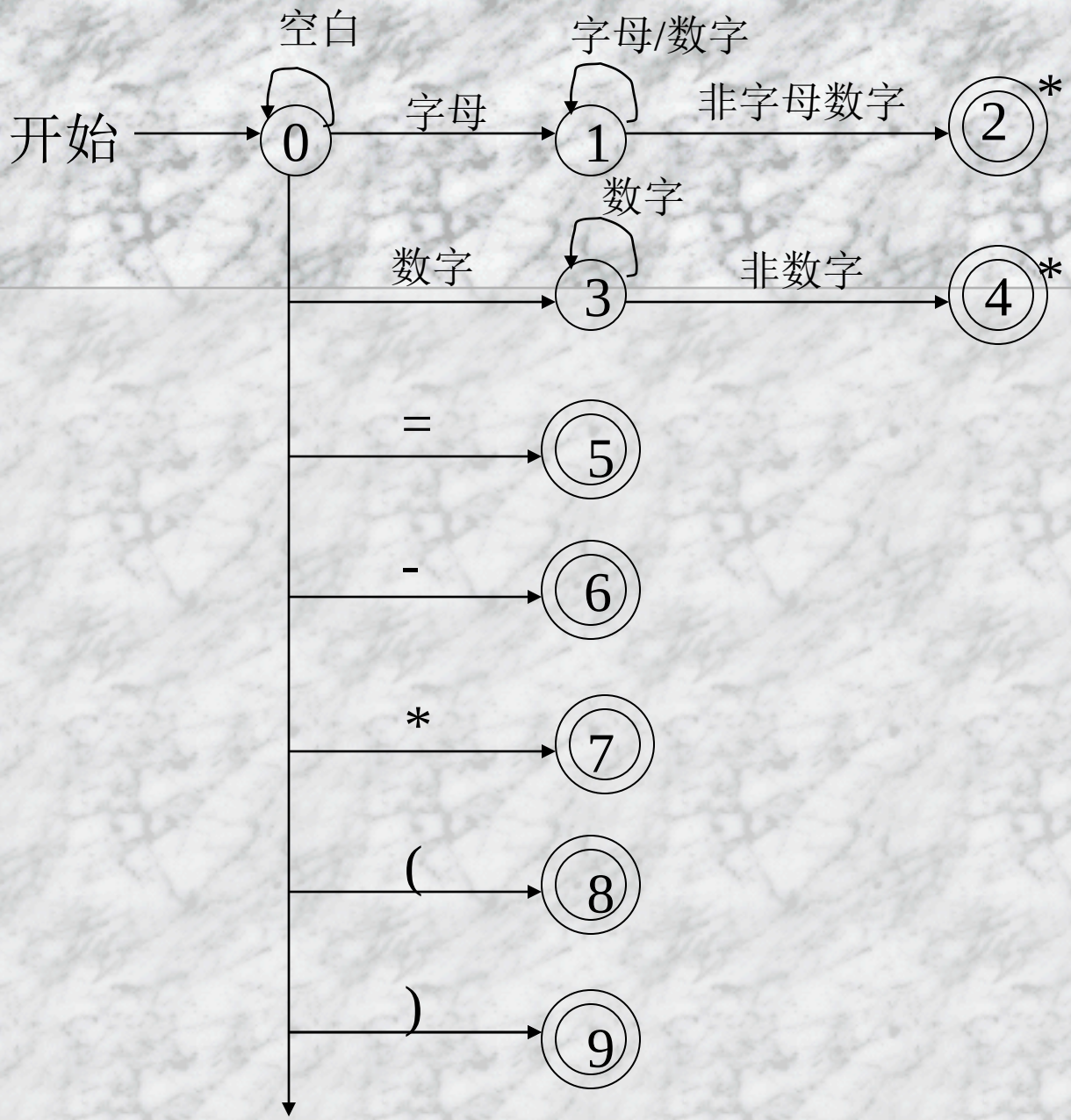
参见目标代码(P码)指令表

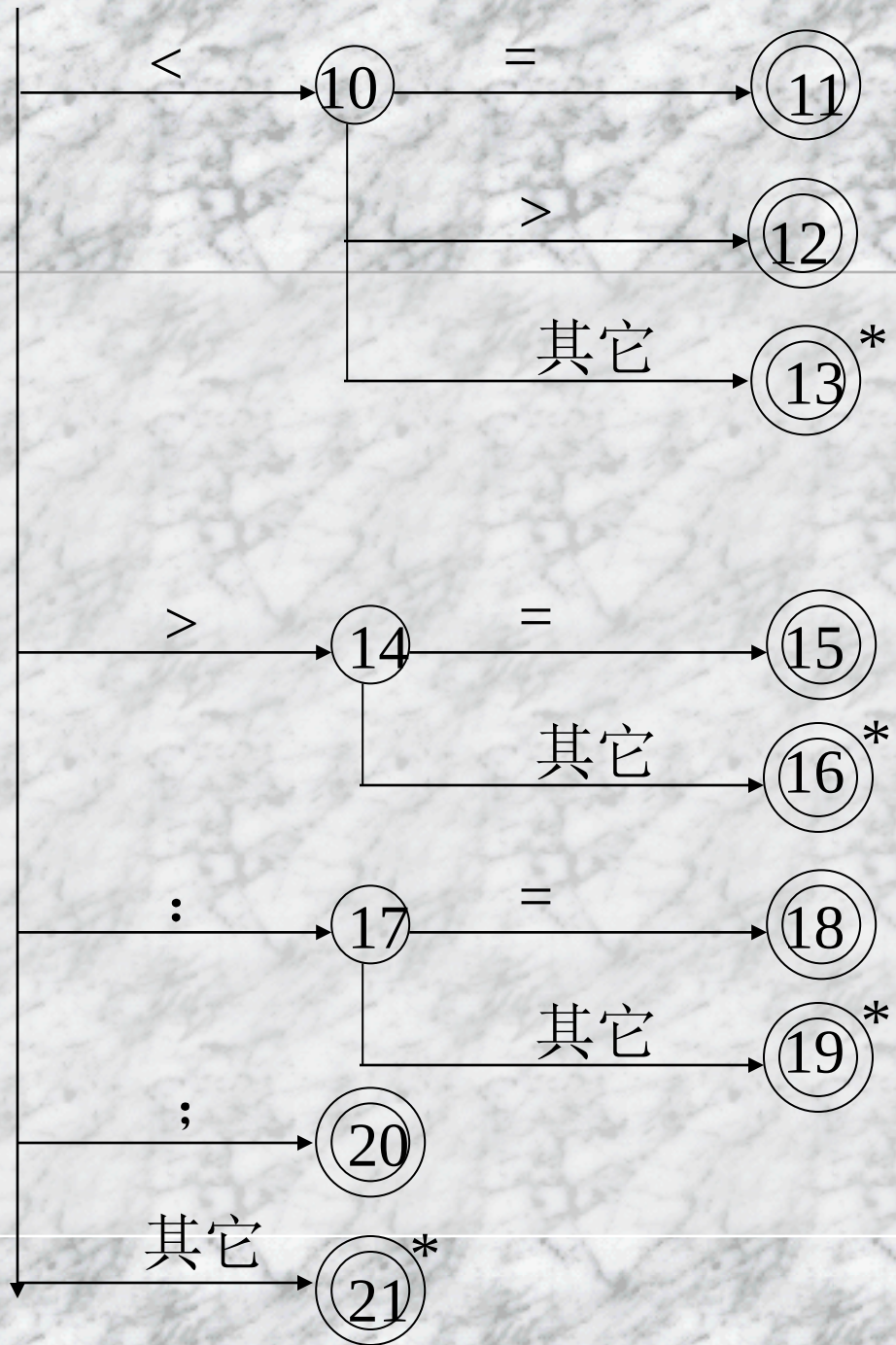
四. 实现提示

1. 多数文件均和源文件同名, 仅扩展名不同。

2. 词法分析时, 注意行尾和文件尾。

3. 词法分析的实现方法——利用状态转换图


空白
字母/数字
开始
0
字母
1
非字母数字
2
*
数字
数字
3
非数字
4
*
=
5
-
6
*
7
(
8
)
9


<
10
=
11
>
12
其它
13
*
>
14
=
15
其它
16
*
:
17
=
18
其它
19
*
;
20
其它
21
*

4.（有过程说明时）设一个总的变量名表，查、填表时注意嵌套。

5. 语法错分类:

(1)缺少符号错;

(2)符号匹配错;

(3)符号无定义或重复定义。

6. 递归下降分析时, 必须先消除左递归。

7. 递归下降分析时, 跟踪符号的位置

当进入某过程时, 其第一个符号必须已经读出; 退出该过程时, 必须读出该语法成分的右界符。

例: G(E) $E \rightarrow E+T \mid T$

$T \rightarrow T*F \mid F$

$F \rightarrow (E) \mid i$

消除左递归: $E \rightarrow T \mid E'$

$E' \rightarrow +TE' \mid \varepsilon$

$T \rightarrow FT'$

$T' \rightarrow *FT' \mid \varepsilon$

$F \rightarrow (E) \mid i$

过程match:匹配单词符号,并读入下一符号

变量lookahead:即将处理但尚未处理的符号

```
procedure match(t:token);
begin
  if lookahead=t then lookahead=nexttoke
  n
        else error
```

```
procedure E;
begin
    T;
    E'
end;
procedure T;
begin
    F;
    T'
end;
```

```
procedure E';
   if lookahead='+' then begin match('+');
                              T;
                              E'
                        end;
procedure T';
   if lookahead='*' then begin match('*');
                              F;
                              T'
                        end;
```

8. 进行自上而下语法制导翻译时，各过程返回的是指针。

五. 实验要求

1. 至少完成词法、语法。

2. 提交: 实验报告

(1) 源程序、测试程序、运行结果

(2) 设计思想、总控算法、主要服务子程序算法、总结(心得、收获、建议等)。

测试程序

```
begin
  integer k;
  integer function F(n);
    begin
      integer n;
      if n<=0 then F:=1
      else F:=n*F(n-1)
    end;
  read(m);
  k:=F(m);
  write(k)
end
```

算符与整数码对照表1

算府	整数码	算符	整数码	算符	整数码
一元负	0	+	1	—	2
*	3	/	4	DIV	5
MOD	6	AND	7	或	8
非	9	>	10	>=	11
<	12	<=	13	<>	14
=	15	赋值	16	无条件转移	17
JNZ(真转移)	18	JZ(假转移)	19	过程调用	20
参数个数	21	程序结束	22	参数传递	23

算符与整数码对照表2

算府	整数码	算符	整数码	算符	整数码
参数返回	24	参数	25	过程返回	26
READ	27	READLN	28	WRITE	29
WRITELN	30	变址存数	31	变址取数	32

其中：22,26无操作数；0,9,17,20,21,23,27,28,29只有一个操作数；16,18,19有两个操作数；1,2,3,4,5,6,7,8,10,11,12,13,14,15,24,25,31,32有三个操作数。

目标代码（P码）指令表1

操作码		参数			功能说明
编码	助记符	T	P	Q	
0	LOD	√	√	√	T—类型，P—层次号，Q—栈内位移 根据P,Q形成栈地址，将其栈内单元的值装载到栈顶单元
1	LDO	√	√	√	T—栈单元类型, P—层次号, Q—栈内位移 以P,Q形成的栈地址单元的内容作为地址，间接取栈单元的值装载到栈顶
2	STR	√	√	√	T—类型，P—层次号，Q—栈内位移 直接存取栈顶单元内容到P,Q指定的栈单元中
3	STO	√	√	√	间接存取栈顶单元的内容，以P,Q形成的栈地址单元为间址单元

目标代码（P码）指令表2

操作码		参数			功能说明
编码	助记符	T	P	Q	
4	LDC	√		√	将类型为T，值为Q的常数装载到栈顶单元
5	NGI				对栈顶单元中的整型数取负
6	NGR				对栈顶单元中的实型数取负
7	ADI				栈顶两单元整型数加操作
8	ADR				栈顶两单元实型数加操作
9	SBI				栈顶两单元整型数减操作

目标代码（P码）指令表3

操作码		参数			功能说明
编码	助记符	T	P	Q	
10	SBR				栈顶两单元实型数减操作
11	MPI				栈顶两单元整型数乘操作
12	MPR				栈顶两单元实型数乘操作
13	DVI				栈顶两单元整型数除操作
14	DVR				栈顶两单元实型数除操作
15	MOD				栈顶两单元整型数取余操作

目标代码（P码）指令表4

操作码		参数			功能说明
编码	助记符	T	P	Q	
16	AND				与操作
17	OR				或操作
18	NOT				非操作
19	GRT	√			大于比较
20	GEQ	√			大于等于比较
21	LES	√			小于比较

目标代码（P码）指令表5

操作码		参数			功能说明
编码	助记符	T	P	Q	
22	LEQ	√			小于等于比较
23	NEQ	√			不等于比较
24	EQ	√			等于比较
25	TJP			√	条件成立转移。若栈顶单元的布尔值为真，则转移至Q表示的目标指令序号执行
26	FJP			√	假转移。若栈顶单元的布尔值为假，则转移至Q表示的目标指令序号执行
27	UJP			√	无条件转移

目标代码（P码）指令表6

操作码		参数			功能说明
编码	助记符	T	P	Q	
28	PROC				转移至过程的起始目标指令处
29	CALL			√	过程调用指令，生成过程的内务管理数据区
30	RET				过程返回，释放栈成分
31	LD				分配一个栈单元
32	LDA	√			取数组元素内容，装入栈顶单元
33	STA	√			将数组元素值存入栈单元

目标代码（P码）指令表7

操作码		参数			功能说明
编码	助记符	T	P	Q	
34	RAD	√			非换行读入数据，装入栈顶
35	RDN	√			换行读入数据，装入栈顶
36	WRI	√			非换行输出栈顶单元内容
37	WRN	√			换行输出栈顶单元内容

实验老师联系方式

陈昆

邮箱：CHENKUN@UESTC.EDU.CN

电话：13880986237