第十二章
PL/0
代 码 生 成

广东工业大学计算机学院

PL/0编译程序的目标代码结构

- PL/0编译程序所产生的目标代码是一个假想栈式计算机的汇编语言，可称为类PCODE指令代码，它不依赖任何实际计算机
- 此类栈式机没有累加器和通用寄存器，有一个栈式存储器，有四个控制寄存器（指令寄存器 I，指令地址寄存器 P，栈顶寄存器 T和基址寄存器 B），算逻运算都在栈顶进行。（这部分内容涉及到PL/0解释程序）

- **1.** 栈顶寄存器(指针)**T**：由于每个过程当它被运行时，给它分配的数据空间(下边称数据段)可分成两部分。
  - 静态部分：包括变量存放区和三个联系单元(**SL**、**DL**和**RA**)。
  - 动态部分：作为临时工作单元和累加器用。需要时随时分配，用完后立即释放。
- **2.** 基址寄存器(指针)**B**：指向每个过程被调用时，在数据区**S**中给它分配的数据段起始地址，也称基地址。（第**10**章的**SP**）
- **3.** 程序地址寄存器 **P**：指向下一条要执行的指令
- **4.** 指令寄存器 **I**：存放正在执行的指令

PL/0编译程序的目标代码结构

- 指令格式

f	l	a

f 功能码

l 层次差（标识符引用层减去定义层）

a 根据不同的指令有所区别

八条目标指令：

PL/0编译程序的目标代码结构和代码生成

- 指令 “LIT 0 A”
  - 立即数存入栈顶，即置T 所指存储单元的值为A
  - T 加 1
- **LOD**：将变量放到栈顶。**a**域为变量在所说明层中的相对位置，**l**为调用层与说明层的层差值。
- **STO**：将栈顶的内容送入某变量单元中。**a**和**l**域的含意同**LOD**指令。
- **CAL**：调用过程的指令。**a**为被调用过程的目标程序入口地址，**l**为层差。
- **INT**：为被调用的过程(或主程序)在运行栈中开辟数据区。**a**域为开辟的单元个数。
- **JMP**：无条件转移指令，**a**为转向地址。
- **JPC**：条件转移指令，当栈顶的布尔值为非真时，转向**a**域的地址，否则顺序执行。
- **OPR**：关系运算和算术运算指令。将栈顶和次栈顶的内容进行运算，结果存放在次栈顶。此外还可以是读写等特殊功能的指令，具体操作由**a**域值给出。（**P23**）

PL/0编译程序的代码生成

- 编译程序的目标代码生成
  - 在分析程序体时生成的
  - 当分析程序体中的每个语句时，当语法正确则调用目标代码生成过程以生成与PL/O语句等价功能的目标代码，直到编译正常结束。除了过程说明部分外，变量和常量的说明都不产生目标代码。
  - 由过程GEN完成，GEN有三个参数，分别代表目标代码的功能码，层差和位移量 **P417**
    - 例如：gen(opr,0,16) 从命令行读入一个输入到栈顶

PL/0编译程序的代码生成

gen(sto,lev-level,adr)

- 将栈顶内容送到变量单元中，

- **lev**：当前处理的过程层次
- **level**：被引用变量或过程所在层次
- **adr:** 变量在过程中的相对位置（即第**9**章讲的**DX**）
- **CX**：为目标代码**code**数组的下标指针（CX为指令的指针，由**0**开始顺序增加。）

PL/0编译程序的代码生成

- 相关定义

Code: array[0..cxmax] of instruction; p414

fct=(lit, opr, lod, sto, cal, int, jmp, jpc) P413

Instruction= packec record P413

f: fct;

l: 0..levmax

a: 0..amax;

end;

- CODE为一维数组，数组元素为记录型数据。每一个记录就是一条目标指令。

PL/0编译程序的代码生成

- *实际上目标代码的顺序是内层过程的排在前边，主程序的目标代码在最后。*
- 下面我们给出一个PL/0源程序和对应的目标程序的清单。

```
const   a=10;
var     b,c;
procedure p;
  begin
    c:=b+a;
  end;
begin
  read(b);
  while b#0 do
  begin
    call p;
    write(2*c);
    read(b);
  end
end.
```

```
( 0) jmp 0 8   转向主程序入口
( 1) jmp 0 2   转向过程p入口
( 2) int 0 3   过程p入口,为过程p开辟空间
( 3) lod 1 3   取变量b的值到栈顶
( 4) lit 0 10  取常数10到栈顶
( 5) opr 0 2   次栈顶与栈顶相加
( 6) sto 1 4   栈顶值送变量c中
( 7) opr 0 0   退栈并返回调用点(16)
( 8) int 0 5   主程序入口开辟5个栈空间
( 9) opr 0 16  从命令行读入值置于栈顶
(10) sto 0 3   将栈顶值存入变量b中
(11) lod 0 3   将变量b的值取至栈顶
(12) lit 0 0   将常数值0进栈
(13) opr 0 9   次栈顶与栈顶是否不等
(14) jpc 0 24  等时转(24)（条件不满足转）
(15) cal 0 2   调用过程p
(16) lit 0 2   常数值2进栈
(17) lod 0 4   将变量c的值取至栈顶
(18) opr 0 4   次栈顶与栈顶相乘(2*c)
(19) opr 0 14  栈顶值输出至屏幕
(20) opr 0 15  换行
(21) opr 0 16  从命令行读取值到栈顶
(22) sto 0 3   栈顶值送变量b中
(23) jmp 0 11  无条件转到循环入口(11)
(24) opr 0 0   结束退栈
```

PL/0编译程序的代码生成

- 对于过程(分程序)的代码生成

  在**block**入口处生成一条**(jmp,0,0)**指令，作为过程体入口指令，该指令的第**3**区域的**'0'**需分析到过程体入口时才返填。

- 对分程序体入口的处理（见程序文本**P424**页**block** 的过程体）

```
begin (*block*)
  dx:=3;
  tx0:=tx;
```

- （*保留当前**table**表指针值,实际为过程名在**table**表中的位置*）

```
table[tx].adr:=cx;（*保留当前code指针值到过程名的adr域*）
gen(jmp,0,0);
```

（*上例的前二行！*）

- 其实上述内容最重要的是地址的回填（见下几页**PPT)**

录过程在code的入口到table中的adr域如下表所示：

```
CONST A=35, B=49;
VAR C, D, E;
PROCEDURE P;
VAR G
```

tx0:=tx;

table[tx].adr:=cx;

code(cx)

(0) jmp 0 0

CX → (1) jmp 0 0

tx0 → tx

NAME: A	KIND: CONSTANT	VAL: 35		
NAME: B	KIND: CONSTANT	VAL: 49		
NAME: C	KIND: VARIABLE	LEVEL: LEV	ADR: DX	
NAME: D	KIND: VARIABLE	LEVEL: LEV	ADR: DX+1	
NAME: E	KIND: VARIABLE	LEVEL: LEV	ADR: DX+2	
NAME: P	KIND: PROCEDUR	LEVEL: LEV	ADR: 1	SIZE: 4
NAME: G	KIND: VARIABLE	LEVEL: LEV+1	ADR: DX	
.....	.....	.....	.....	
名字	类型	层次/值	地址	存储空间

（*生成转向过程体入口的指令，该指令的地址为cx已保留在过程名的adr域，真正的过程体入口地址，等生成过程体入口的指令时，将过程体入口返填（jmp,0,0）的第3区域，同时填到table[tx0].adr 中*)

table表格管理

code[table[tx0].adr].a:=cx;

```
CONST A=35, B=49;
VAR C, D, E;
PROCEDURE P;
VAR G
```

过程的入口地址填写在code和table中

table[tx0].adr=cx

```
(0)  jmp  0  0
(1)  jmp  0  0
      ...
(cx) int  0  4
```

	名字	类型	层次/值	地址	存储空间
	NAME: A	KIND: CONSTANT	VAL: 35		
	NAME: B	KIND: CONSTANT	VAL: 49		
	NAME: C	KIND: VARIABLE	LEVEL: LEV	ADR: DX	
	NAME: D	KIND: VARIABLE	LEVEL: LEV	ADR: DX+1	
	NAME: E	KIND: VARIABLE	LEVEL: LEV	ADR: DX+2	
tx0 →	NAME: P	KIND: PROCEDUR	LEVEL: LEV	ADR: 1 cx	SIZE: 4
	NAME: G	KIND: VARIABLE	LEVEL: LEV+1	ADR: DX	
tx →	.....	.....	.....	.....	

过程体入口时的处理 table表格管理

code[table[tx0].adr].a:=cx;（cx为过程入口地址，填写在code中）

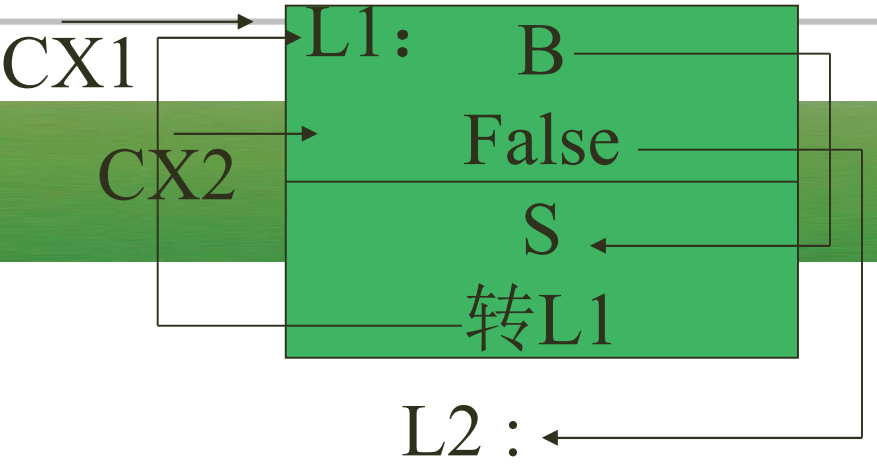
```
with table[tx0] do
  begin
    adr:=cx; （table[tx0].adr=cx）
    size:=dx; （table[tx0].size＝dx）
  end;
```

请特别注意dx、 tx、 cx的作用和如何处理信息之间的连接关系。

While B Do S

switch (SYM) {

……



CASE WHILESYM:

CX1=CX; GetSym(); //保留L1的地址

CONDITION(SymSetAdd(DOSYM,FSYS),LEV,TX);

//生成B的代码

CX2=CX; GEN(JPC,0,0);//条件跳转，L2待定，

if (SYM==DOSYM) GetSym();

else Error(18);

STATEMENT(FSYS,LEV,TX);//生成S的代码

GEN(JMP,0,CX1); //无条件跳转， 转到L1

CODE[CX2].A=CX; //回填L2

break;

- 附录
- 当源程序经过语法分析，如果未发现错误时，由编译程序调用解释程序,对目标代码开始进行解释执行。
- **PL/0**解释程序

PL/0解释程序

PL/0解释程序的存储分配

- 存储区：
  - 数组CODE存放目标程序
  - 运行时的数据区S
- S是由解释程序定义的一维整型数组
- 由于PL/0语言的目标程序是一种假想的栈式计算机的汇编语言，现仍用Pascal语言解释执行

PL/0解释程序

- 解释程序定义了4个寄存器：
- ①I　指令寄存器:存放当前正在解释的一条目标指令
- ②P　程序地址寄存器:指向下一条要执行的目标程序的
- 　地址（相当目标程序CODE数组的下标）

③T　栈顶寄存器。指向当前栈中最新分配的单元（T也是数组S的下标）。

每个过程当它被调用时，给它分配的数据空间（下边称数据段）可分成两部分：

- 静态部分：包括变量存放区和三个联系单元
- 动态部分：作为临时工作单元和累加器用。需要时随时分配，用完后立即释放

④B　基址寄存器。指向当前执行过程的在数据区S中给该过程分配的数据段起始地址，称基地址

- 当过程被调用时，在栈顶分配三个联系单元，这三个联系单元存放的内容分别为：

①SL　静态链：指向定义该过程的直接外过程（或主程序）运行时最新数据段的基地址

②DL　动态链：指向调用该过程前正在运行过程的数据段基地址

③RA　返回地址：记录调用该过程目标程序的断点，即当时的程序地址寄存器P的值。也就是调用过程指令的下一条指令的地址

❖PL/0编译程序给每个过程定义的变量在数据段内分配的相对位置是从3开始顺序增加。前面的三个单元为上面指出的联系单元

※具体的过程调用和结束，对上述寄存器及3个联系单元的填写和恢复由下列目标指令完成

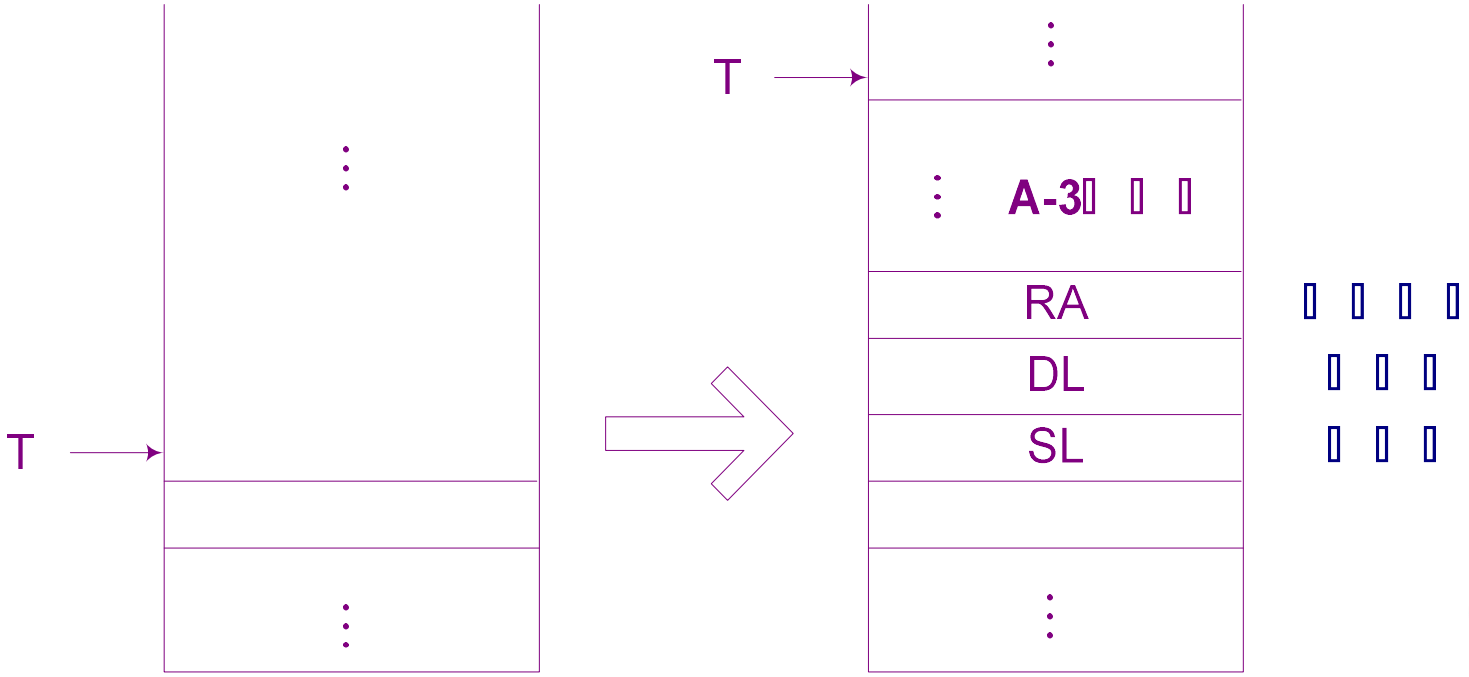
（1）INT 0 A

（2）OPR 0 0

（3）CAL L A

PL/O编译程序的目标代码结构

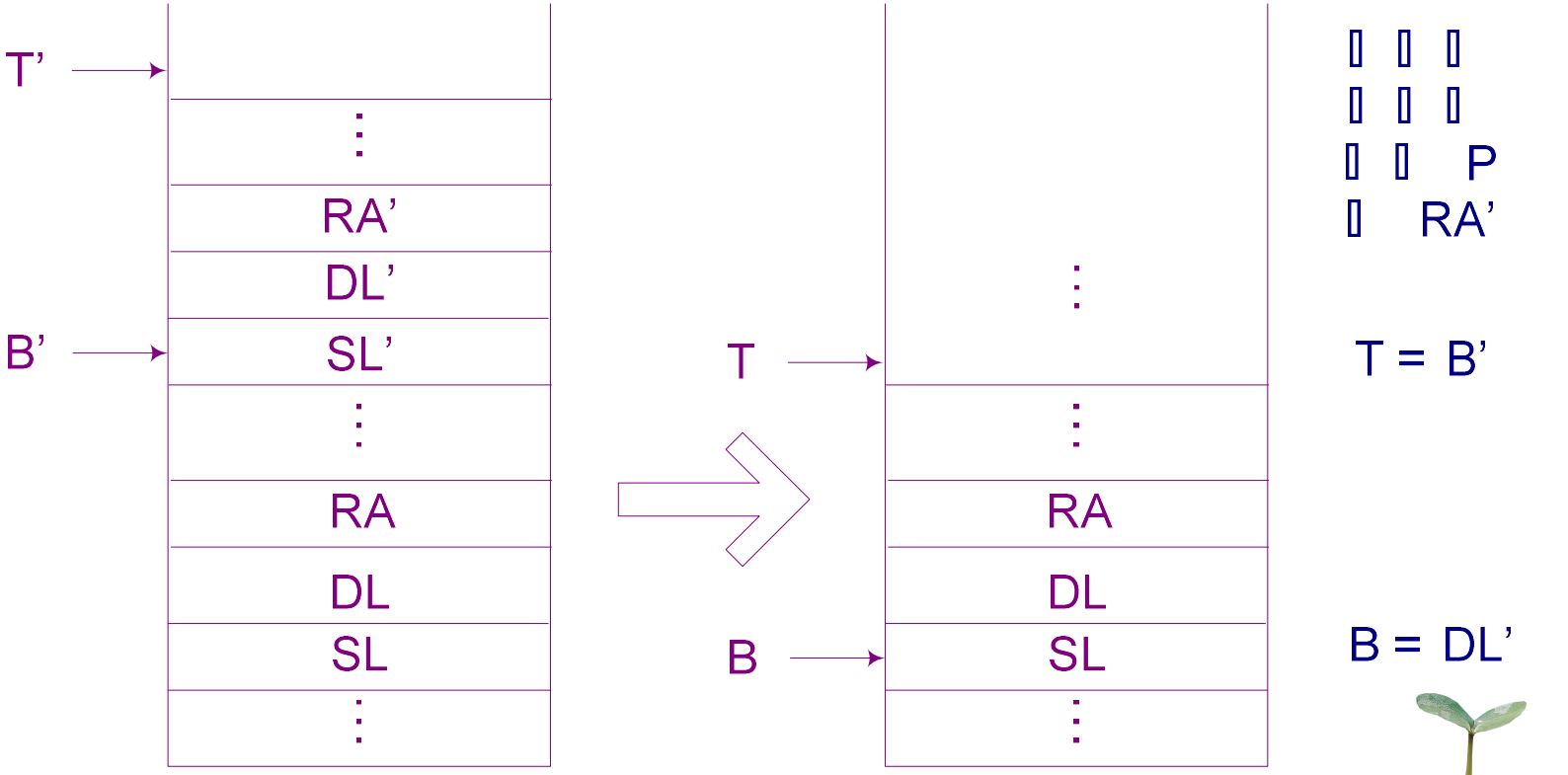
- **指令 “INT 0 A”**
  - **在栈顶开辟 A 个存储单元，服务于被调用的过程**
  - **A 等于该过程的局部变量数加 3**
  - **3 个特殊的联系单元**



类P-code虚拟机

✧ 指令 “OPR 0 0”

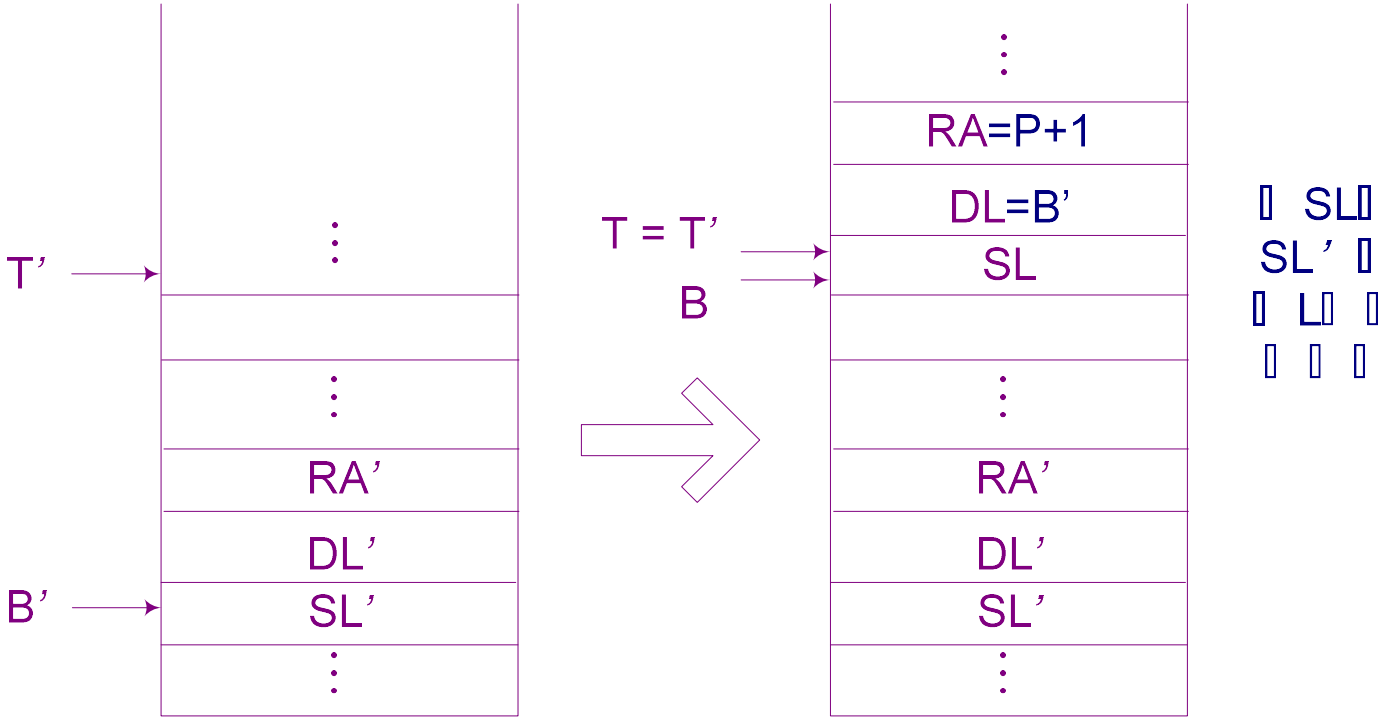
- 过程调用结束后,返回调用点并退栈
- 重置基址寄存器和栈顶寄存器



类P-code虚拟机

✧ 指令 “CAL L A”

- 调用地址为 A 的过程（置指令地址寄存器为A)
- L 为调用过程与被调用过程的层差
- 设置被调用过程的3 个联系单元



类P-code虚拟机

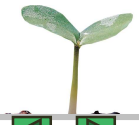
- **指令 “LIT 0 A”**
  - 立即数存入栈顶，即置T 所指存储单元的值为A
  - T 加 1
- **指令 “LOD L A”**
  - 将层差为L、偏移量为A的存储单元的值取到栈顶
  - T 加 1
- **指令 “STO L A”**
  - T 减 1
  - 将栈顶的值存入层差为L、偏移量为A的存储单元
- **注：层差为L、偏移量为A的存储单元，即沿当前层静态链SL开始向前第L层的SL作为基址，加上A，即为该单元的地址**

类P-code虚拟机

- **指令** “OPR 0 1”
  - **求栈顶元素的相反数，结果值留在栈顶**
- **指令** “OPR 0 6”
  - **栈顶元素的奇偶判断，若为奇数，结果为1；若为偶数，结果为0；结果值留在栈顶**

类P-code虚拟机

- 指令 “OPR 0 2”
  - 次栈顶与栈顶的值相加，结果存入次栈顶
  - T 减 1
- 指令 “OPR 0 3”
  - 次栈顶的值减去栈顶的值，结果存入次栈顶
  - T 减 1
- 指令 “OPR 0 4”
  - 次栈顶的值乘以栈顶的值，结果存入次栈顶
  - T 减 1
- 指令 “OPR 0 5”
  - 次栈顶的值除以栈顶的值，结果存入次栈顶
  - T 减 1



类P-code虚拟机

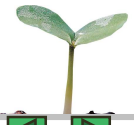
- ✧ 指令 “OPR 0 8” 比较次栈顶与栈顶是否相等
  - 若相等，结果为0；存结果至次栈顶；T 减1
- ✧ 指令 “OPR 0 9” 比较次栈顶与栈顶是否不相等
  - 若不相等，结果为0；存结果至次栈顶；T 减1
- ✧ 指令 “OPR 0 10” 比较次栈顶是否小于栈顶
  - 若小于，结果为0；存结果至次栈顶；T 减1
- ✧ 指令 “OPR 0 11” 比较次栈顶是否大于等于栈顶
  - 若大于等于，结果为0；存结果至次栈顶；T 减1
- ✧ 指令 “OPR 0 12” 比较次栈顶是否大于栈顶
  - 若大于，结果为0；存结果至次栈顶；T 减1
- ✧ 指令 “OPR 0 13” 比较次栈顶是否小于等于栈顶
  - 若小于等于，结果为0；存结果至次栈顶；T 减1

类P-code虚拟机

- **指令** “JMP 0 A”
  - **无条件转移至地址 A，即置指令地址寄存器为A**
- **指令** “JPC 0 A”
  - **条件转移指令**
  - **若栈顶为 0，则转移至地址 A，即置指令地址寄存器为A**

类P-code虚拟机

- **指令** “OPR 0 14”
  - 栈顶的值输出至控制台屏幕
  - T 减 1
- **指令** “OPR 0 15”
  - 控制台屏幕输出一个换行
- **指令** “OPR 0 16”
  - 从控制台读入一行输入，置入栈顶
  - T 加 1



类P-code虚拟机

✧ 类P-code 解释程序

– 数据结构

运行栈 int s[stacksize]

指令寄存器 struct instruction

```
{
    enum fct  f;   /*操作码*/
    int l;    /*引用层与声明层的层差*/
    int a;   /*因不同的f各异*/
} i
```

指令地址寄存器 int p;

基址寄存器 int b;

栈顶寄存器 int t;

虚拟机代码段 struct instruction code[cxmax];

类P-code虚拟机

✧ 类P-code 解释程序

– 处理流程

（1）**初始化** p=b=t=0； s[0]=s[1]=s[2]=0；

（2）**取指令到指令寄存器** i=code[p]； p++；

（3）**分析并解释执行指令** i

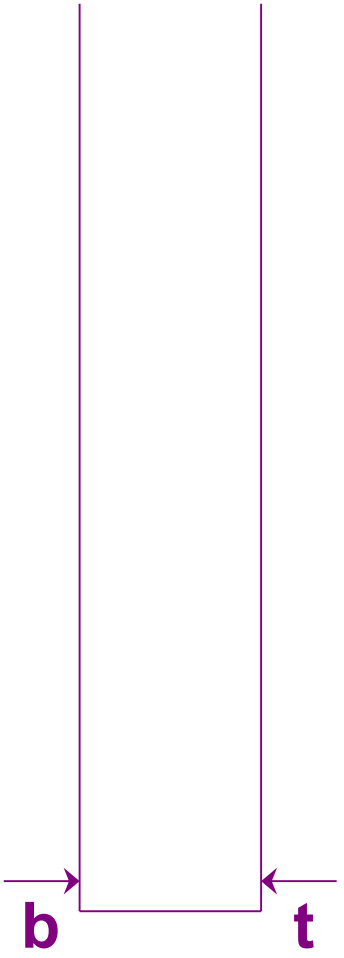
（4）**若程序未结束（**p != 0**），转**（2）

（5）**返回**

类P-code虚拟机

p →

```
( 0) jmp 0 8      转向主程序入口
( 1) jmp 0 2      转向过程p入口
( 2) int 0 3      过程p入口,为过程p开辟空间
( 3) lod 1 3      取变量b的值到栈顶
( 4) lit 0 10     取常数10到栈顶
( 5) opr 0 2      次栈顶与栈顶相加
( 6) sto 1 4      栈顶值送变量c中
( 7) opr 0 0      退栈并返回调用点(16)
( 8) int 0 5      主程序入口开辟5个栈空间
( 9) opr 0 16     从命令行读入值置于栈顶
(10) sto 0 3      将栈顶值存入变量b中
(11) lod 0 3      将变量b的值取至栈顶
(12) lit 0 0      将常数值0进栈
(13) opr 0 9      次栈顶与栈顶是否不等
(14) jpc 0 24     相等时转(24)（条件不满足转)
(15) cal 0 2      调用过程p
(16) lit 0 2      常数值2进栈
(17) lod 0 4      将变量c的值取至栈顶
(18) opr 0 4      次栈顶与栈顶相乘(2*c)
(19) opr 0 14     栈顶值输出至屏幕
(20) opr 0 15     换行
(21) opr 0 16     从命令行读取值到栈顶
(22) sto 0 3      栈顶值送变量b中
(23) jmp 0 11     无条件转到循环入口(11)
(24) opr 0 0      结束退栈
```



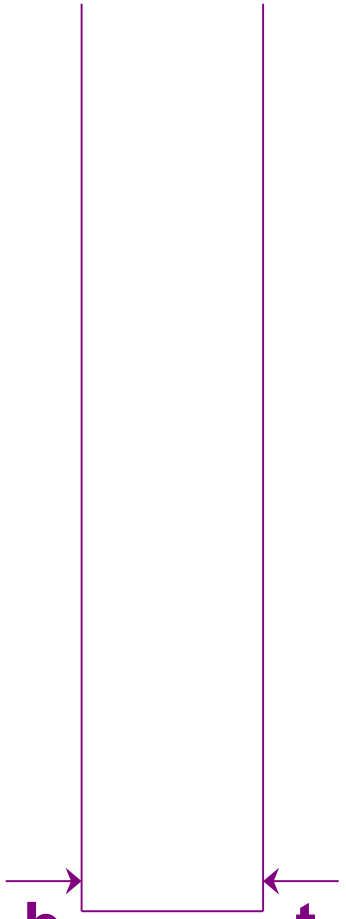
✧ 例

```
const  a=10;
var    b,c;
procedure p;
  begin
    c:=b+a;
  end;
begin
  read(b);
  while b#0 do
  begin
    call p;
    write(2*c);
    read(b);
  end
end.
```

类P-code虚拟机

```
( 0) jmp 0 8      转向主程序入口
( 1) jmp 0 2      转向过程p入口
( 2) int 0 3      过程p入口,为过程p开辟空间
( 3) lod 1 3      取变量b的值到栈顶
( 4) lit 0 10     取常数10到栈顶
( 5) opr 0 2      次栈顶与栈顶相加
( 6) sto 1 4      栈顶值送变量c中
( 7) opr 0 0      退栈并返回调用点(16)
( 8) int 0 5      主程序入口开辟5个栈空间
( 9) opr 0 16     从命令行读入值置于栈顶
(10) sto 0 3      将栈顶值存入变量b中
(11) lod 0 3      将变量b的值取至栈顶
(12) lit 0 0      将常数值0进栈
(13) opr 0 9      次栈顶与栈顶是否不等
(14) jpc 0 24     相等时转(24)（条件不满足转)
(15) cal 0 2      调用过程p
(16) lit 0 2      常数值2进栈
(17) lod 0 4      将变量c的值取至栈顶
(18) opr 0 4      次栈顶与栈顶相乘(2*c)
(19) opr 0 14     栈顶值输出至屏幕
(20) opr 0 15     换行
(21) opr 0 16     从命令行读取值到栈顶
(22) sto 0 3      栈顶值送变量b中
(23) jmp 0 11     无条件转到循环入口(11)
(24) opr 0 0      结束退栈
```

p →



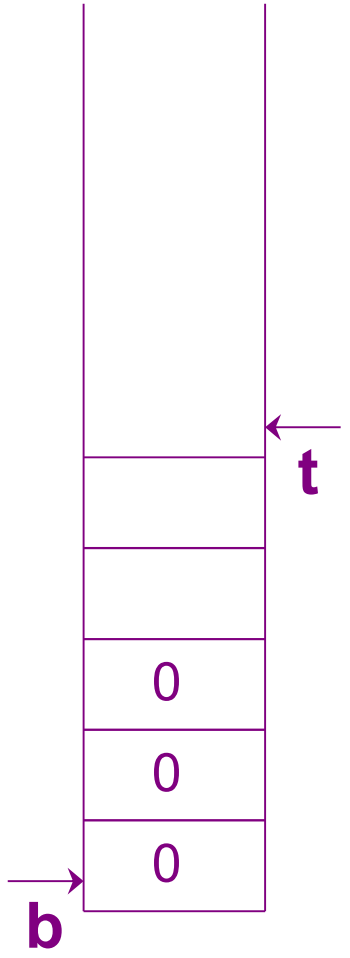
✧ 例

```
const  a=10;
var    b,c;
procedure p;
  begin
    c:=b+a;
  end;
begin
  read(b);
  while b#0 do
  begin
    call p;
    write(2*c);
    read(b);
  end
end.
```

类P-code虚拟机

```
( 0) jmp 0 8      转向主程序入口
( 1) jmp 0 2      转向过程p入口
( 2) int 0 3      过程p入口,为过程p开辟空间
( 3) lod 1 3      取变量b的值到栈顶
( 4) lit 0 10     取常数10到栈顶
( 5) opr 0 2      次栈顶与栈顶相加
( 6) sto 1 4      栈顶值送变量c中
( 7) opr 0 0      退栈并返回调用点(16)
( 8) int 0 5      主程序入口开辟5个栈空间
( 9) opr 0 16     从命令行读入值置于栈顶
(10) sto 0 3      将栈顶值存入变量b中
(11) lod 0 3      将变量b的值取至栈顶
(12) lit 0 0      将常数值0进栈
(13) opr 0 9      次栈顶与栈顶是否不等
(14) jpc 0 24     相等时转(24)（条件不满足转)
(15) cal 0 2      调用过程p
(16) lit 0 2      常数值2进栈
(17) lod 0 4      将变量c的值取至栈顶
(18) opr 0 4      次栈顶与栈顶相乘(2*c)
(19) opr 0 14     栈顶值输出至屏幕
(20) opr 0 15     换行
(21) opr 0 16     从命令行读取值到栈顶
(22) sto 0 3      栈顶值送变量b中
(23) jmp 0 11     无条件转到循环入口(11)
(24) opr 0 0      结束退栈
```

p →



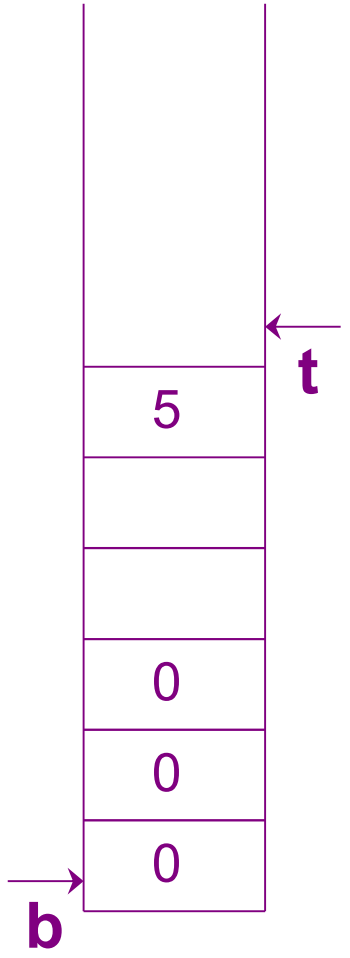
✧ 例

```
const  a=10;
var    b,c;
procedure p;
  begin
    c:=b+a;
  end;
begin
  read(b);
  while b#0 do
  begin
    call p;
    write(2*c);
    read(b);
  end
end.
```

类P-code虚拟机

```
( 0) jmp 0 8      转向主程序入口
( 1) jmp 0 2      转向过程p入口
( 2) int 0 3      过程p入口,为过程p开辟空间
( 3) lod 1 3      取变量b的值到栈顶
( 4) lit 0 10     取常数10到栈顶
( 5) opr 0 2      次栈顶与栈顶相加
( 6) sto 1 4      栈顶值送变量c中
( 7) opr 0 0      退栈并返回调用点(16)
( 8) int 0 5      主程序入口开辟5个栈空间
( 9) opr 0 16     从命令行读入值置于栈顶
(10) sto 0 3      将栈顶值存入变量b中
(11) lod 0 3      将变量b的值取至栈顶
(12) lit 0 0      将常数值0进栈
(13) opr 0 9      次栈顶与栈顶是否不等
(14) jpc 0 24     相等时转(24)（条件不满足转）
(15) cal 0 2      调用过程p
(16) lit 0 2      常数值2进栈
(17) lod 0 4      将变量c的值取至栈顶
(18) opr 0 4      次栈顶与栈顶相乘(2*c)
(19) opr 0 14     栈顶值输出至屏幕
(20) opr 0 15     换行
(21) opr 0 16     从命令行读取值到栈顶
(22) sto 0 3      栈顶值送变量b中
(23) jmp 0 11     无条件转到循环入口(11)
(24) opr 0 0      结束退栈
```

p →



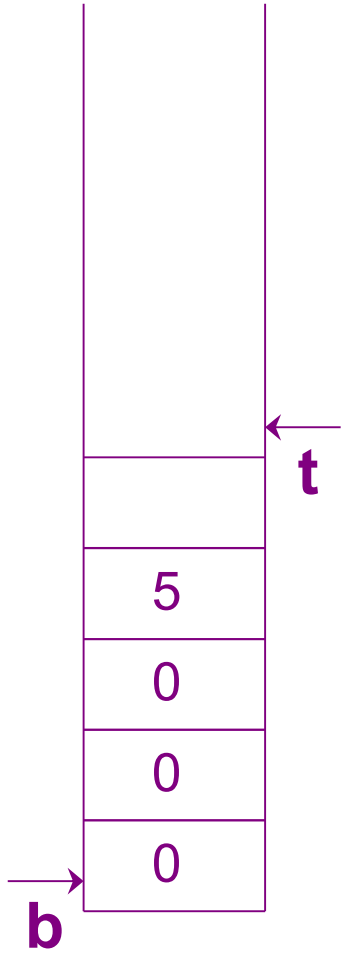
✧ 例

```
const  a=10;
var    b,c;
procedure p;
  begin
    c:=b+a;
  end;
begin
  read(b);
  while b#0 do
  begin
    call p;
    write(2*c);
    read(b);
  end
end.
```

类P-code虚拟机

```
( 0) jmp 0 8      转向主程序入口
( 1) jmp 0 2      转向过程p入口
( 2) int 0 3      过程p入口,为过程p开辟空间
( 3) lod 1 3      取变量b的值到栈顶
( 4) lit 0 10     取常数10到栈顶
( 5) opr 0 2      次栈顶与栈顶相加
( 6) sto 1 4      栈顶值送变量c中
( 7) opr 0 0      退栈并返回调用点(16)
( 8) int 0 5      主程序入口开辟5个栈空间
( 9) opr 0 16     从命令行读入值置于栈顶
(10) sto 0 3      将栈顶值存入变量b中
(11) lod 0 3      将变量b的值取至栈顶
(12) lit 0 0      将常数值0进栈
(13) opr 0 9      次栈顶与栈顶是否不等
(14) jpc 0 24     相等时转(24)（条件不满足转)
(15) cal 0 2      调用过程p
(16) lit 0 2      常数值2进栈
(17) lod 0 4      将变量c的值取至栈顶
(18) opr 0 4      次栈顶与栈顶相乘(2*c)
(19) opr 0 14     栈顶值输出至屏幕
(20) opr 0 15     换行
(21) opr 0 16     从命令行读取值到栈顶
(22) sto 0 3      栈顶值送变量b中
(23) jmp 0 11     无条件转到循环入口(11)
(24) opr 0 0      结束退栈
```

p →



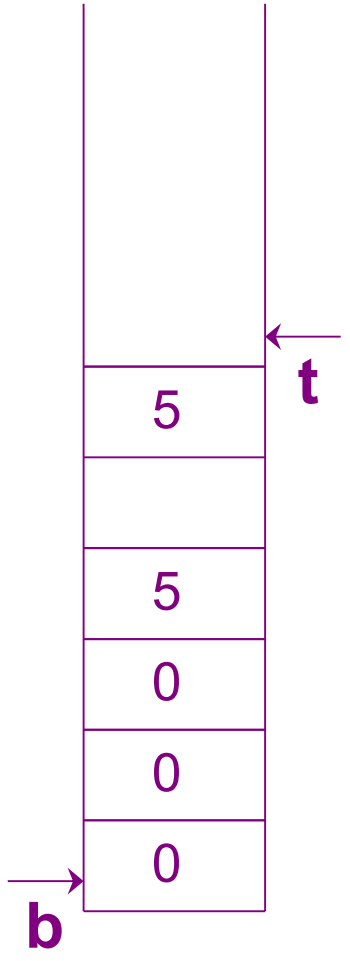
✧ 例

```
const  a=10;
var    b,c;
procedure p;
  begin
    c:=b+a;
  end;
begin
  read(b);
  while b#0 do
  begin
    call p;
    write(2*c);
    read(b);
  end
end.
```

类P-code虚拟机

```
( 0) jmp 0 8      转向主程序入口
( 1) jmp 0 2      转向过程p入口
( 2) int 0 3      过程p入口,为过程p开辟空间
( 3) lod 1 3      取变量b的值到栈顶
( 4) lit 0 10     取常数10到栈顶
( 5) opr 0 2      次栈顶与栈顶相加
( 6) sto 1 4      栈顶值送变量c中
( 7) opr 0 0      退栈并返回调用点(16)
( 8) int 0 5      主程序入口开辟5个栈空间
( 9) opr 0 16     从命令行读入值置于栈顶
(10) sto 0 3      将栈顶值存入变量b中
(11) lod 0 3      将变量b的值取至栈顶
(12) lit 0 0      将常数值0进栈
(13) opr 0 9      次栈顶与栈顶是否不等
(14) jpc 0 24     相等时转(24)（条件不满足转)
(15) cal 0 2      调用过程p
(16) lit 0 2      常数值2进栈
(17) lod 0 4      将变量c的值取至栈顶
(18) opr 0 4      次栈顶与栈顶相乘(2*c)
(19) opr 0 14     栈顶值输出至屏幕
(20) opr 0 15     换行
(21) opr 0 16     从命令行读取值到栈顶
(22) sto 0 3      栈顶值送变量b中
(23) jmp 0 11     无条件转到循环入口(11)
(24) opr 0 0      结束退栈
```

p →



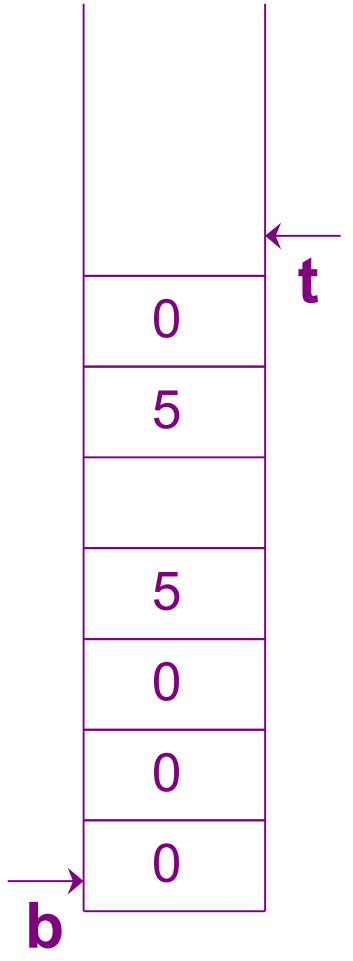
✧ 例

```
const  a=10;
var    b,c;
procedure p;
  begin
    c:=b+a;
  end;
begin
  read(b);
  while b#0 do
  begin
    call p;
    write(2*c);
    read(b);
  end
end.
```

类P-code虚拟机

```
( 0) jmp 0 8      转向主程序入口
( 1) jmp 0 2      转向过程p入口
( 2) int 0 3      过程p入口,为过程p开辟空间
( 3) lod 1 3      取变量b的值到栈顶
( 4) lit 0 10     取常数10到栈顶
( 5) opr 0 2      次栈顶与栈顶相加
( 6) sto 1 4      栈顶值送变量c中
( 7) opr 0 0      退栈并返回调用点(16)
( 8) int 0 5      主程序入口开辟5个栈空间
( 9) opr 0 16     从命令行读入值置于栈顶
(10) sto 0 3      将栈顶值存入变量b中
(11) lod 0 3      将变量b的值取至栈顶
(12) lit 0 0      将常数值0进栈
(13) opr 0 9      次栈顶与栈顶是否不等
(14) jpc 0 24     相等时转(24)（条件不满足转)
(15) cal 0 2      调用过程p
(16) lit 0 2      常数值2进栈
(17) lod 0 4      将变量c的值取至栈顶
(18) opr 0 4      次栈顶与栈顶相乘(2*c)
(19) opr 0 14     栈顶值输出至屏幕
(20) opr 0 15     换行
(21) opr 0 16     从命令行读取值到栈顶
(22) sto 0 3      栈顶值送变量b中
(23) jmp 0 11     无条件转到循环入口(11)
(24) opr 0 0      结束退栈
```

p →



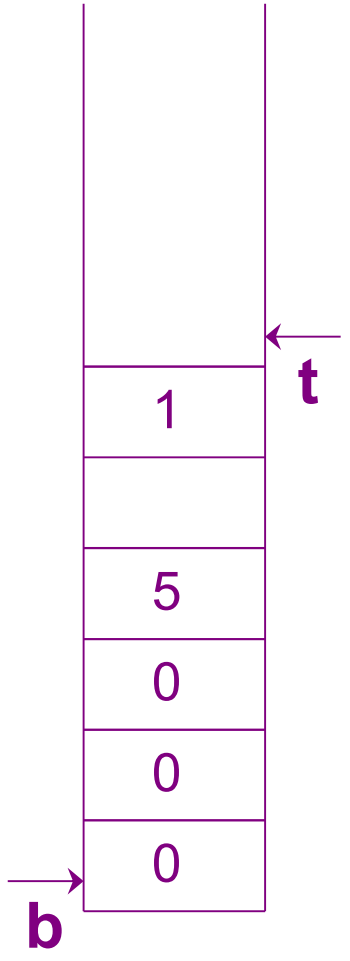
✧ 例

```
const  a=10;
var    b,c;
procedure p;
  begin
    c:=b+a;
  end;
begin
  read(b);
  while b#0 do
  begin
    call p;
    write(2*c);
    read(b);
  end
end.
```

类P-code虚拟机

```
( 0) jmp 0 8      转向主程序入口
( 1) jmp 0 2      转向过程p入口
( 2) int 0 3      过程p入口,为过程p开辟空间
( 3) lod 1 3      取变量b的值到栈顶
( 4) lit 0 10     取常数10到栈顶
( 5) opr 0 2      次栈顶与栈顶相加
( 6) sto 1 4      栈顶值送变量c中
( 7) opr 0 0      退栈并返回调用点(16)
( 8) int 0 5      主程序入口开辟5个栈空间
( 9) opr 0 16     从命令行读入值置于栈顶
(10) sto 0 3      将栈顶值存入变量b中
(11) lod 0 3      将变量b的值取至栈顶
(12) lit 0 0      将常数值0进栈
(13) opr 0 9      次栈顶与栈顶是否不等
(14) jpc 0 24     相等时转(24)（条件不满足转)
(15) cal 0 2      调用过程p
(16) lit 0 2      常数值2进栈
(17) lod 0 4      将变量c的值取至栈顶
(18) opr 0 4      次栈顶与栈顶相乘(2*c)
(19) opr 0 14     栈顶值输出至屏幕
(20) opr 0 15     换行
(21) opr 0 16     从命令行读取值到栈顶
(22) sto 0 3      栈顶值送变量b中
(23) jmp 0 11     无条件转到循环入口(11)
(24) opr 0 0      结束退栈
```

p →



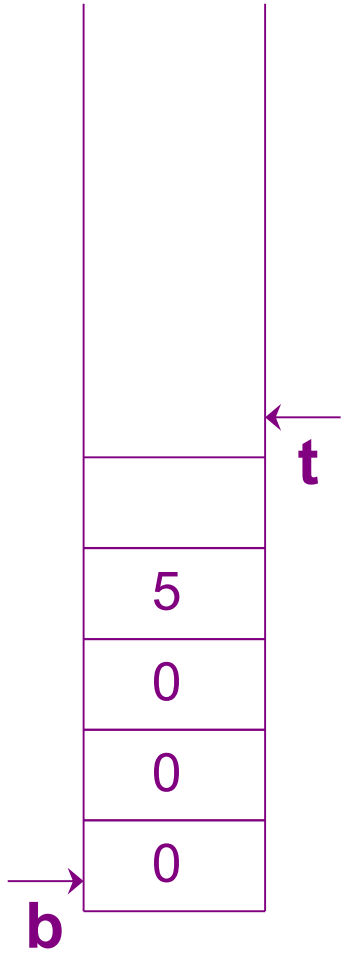
✧ 例

```
const  a=10;
var    b,c;
procedure p;
  begin
    c:=b+a;
  end;
begin
  read(b);
  while b#0 do
  begin
    call p;
    write(2*c);
    read(b);
  end
end.
```

类P-code虚拟机

```
( 0) jmp 0 8      转向主程序入口
( 1) jmp 0 2      转向过程p入口
( 2) int 0 3      过程p入口,为过程p开辟空间
( 3) lod 1 3      取变量b的值到栈顶
( 4) lit 0 10     取常数10到栈顶
( 5) opr 0 2      次栈顶与栈顶相加
( 6) sto 1 4      栈顶值送变量c中
( 7) opr 0 0      退栈并返回调用点(16)
( 8) int 0 5      主程序入口开辟5个栈空间
( 9) opr 0 16     从命令行读入值置于栈顶
(10) sto 0 3      将栈顶值存入变量b中
(11) lod 0 3      将变量b的值取至栈顶
(12) lit 0 0      将常数值0进栈
(13) opr 0 9      次栈顶与栈顶是否不等
(14) jpc 0 24     相等时转(24)（条件不满足转)
(15) cal 0 2      调用过程p
(16) lit 0 2      常数值2进栈
(17) lod 0 4      将变量c的值取至栈顶
(18) opr 0 4      次栈顶与栈顶相乘(2*c)
(19) opr 0 14     栈顶值输出至屏幕
(20) opr 0 15     换行
(21) opr 0 16     从命令行读取值到栈顶
(22) sto 0 3      栈顶值送变量b中
(23) jmp 0 11     无条件转到循环入口(11)
(24) opr 0 0      结束退栈
```

p →



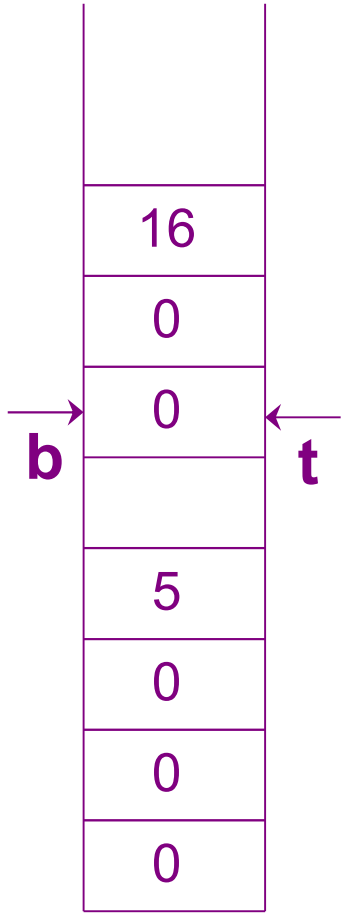
✧ 例

```
const  a=10;
var    b,c;
procedure p;
  begin
    c:=b+a;
  end;
begin
  read(b);
  while b#0 do
  begin
    call p;
    write(2*c);
    read(b);
  end
end.
```

类P-code虚拟机

```
( 0) jmp 0 8      转向主程序入口
( 1) jmp 0 2      转向过程p入口
( 2) int 0 3      过程p入口,为过程p开辟空间
( 3) lod 1 3      取变量b的值到栈顶
( 4) lit 0 10     取常数10到栈顶
( 5) opr 0 2      次栈顶与栈顶相加
( 6) sto 1 4      栈顶值送变量c中
( 7) opr 0 0      退栈并返回调用点(16)
( 8) int 0 5      主程序入口开辟5个栈空间
( 9) opr 0 16     从命令行读入值置于栈顶
(10) sto 0 3      将栈顶值存入变量b中
(11) lod 0 3      将变量b的值取至栈顶
(12) lit 0 0      将常数值0进栈
(13) opr 0 9      次栈顶与栈顶是否不等
(14) jpc 0 24     相等时转(24)（条件不满足转)
(15) cal 0 2      调用过程p
(16) lit 0 2      常数值2进栈
(17) lod 0 4      将变量c的值取至栈顶
(18) opr 0 4      次栈顶与栈顶相乘(2*c)
(19) opr 0 14     栈顶值输出至屏幕
(20) opr 0 15     换行
(21) opr 0 16     从命令行读取值到栈顶
(22) sto 0 3      栈顶值送变量b中
(23) jmp 0 11     无条件转到循环入口(11)
(24) opr 0 0      结束退栈
```

p →



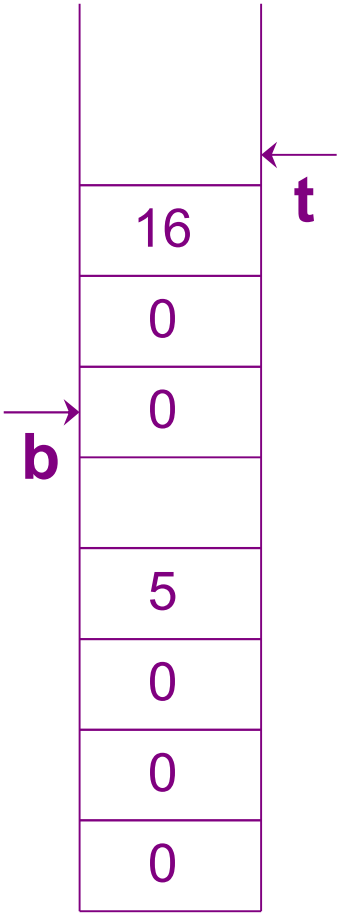
✧ 例

```
const  a=10;
var    b,c;
procedure p;
  begin
    c:=b+a;
  end;
begin
  read(b);
  while b#0 do
  begin
    call p;
    write(2*c);
    read(b);
  end
end.
```

类P-code虚拟机

```
( 0) jmp 0 8      转向主程序入口
( 1) jmp 0 2      转向过程p入口
( 2) int 0 3      过程p入口,为过程p开辟空间
( 3) lod 1 3      取变量b的值到栈顶
( 4) lit 0 10     取常数10到栈顶
( 5) opr 0 2      次栈顶与栈顶相加
( 6) sto 1 4      栈顶值送变量c中
( 7) opr 0 0      退栈并返回调用点(16)
( 8) int 0 5      主程序入口开辟5个栈空间
( 9) opr 0 16     从命令行读入值置于栈顶
(10) sto 0 3      将栈顶值存入变量b中
(11) lod 0 3      将变量b的值取至栈顶
(12) lit 0 0      将常数值0进栈
(13) opr 0 9      次栈顶与栈顶是否不等
(14) jpc 0 24     相等时转(24)（条件不满足转)
(15) cal 0 2      调用过程p
(16) lit 0 2      常数值2进栈
(17) lod 0 4      将变量c的值取至栈顶
(18) opr 0 4      次栈顶与栈顶相乘(2*c)
(19) opr 0 14     栈顶值输出至屏幕
(20) opr 0 15     换行
(21) opr 0 16     从命令行读取值到栈顶
(22) sto 0 3      栈顶值送变量b中
(23) jmp 0 11     无条件转到循环入口(11)
(24) opr 0 0      结束退栈
```

p →



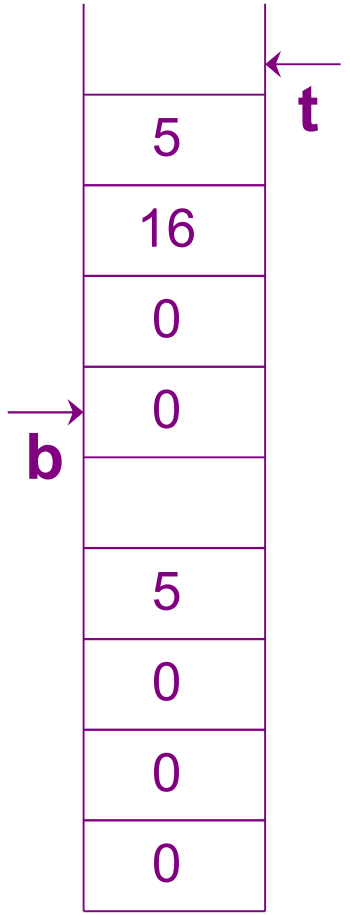
✧ 例

```
const  a=10;
var    b,c;
procedure p;
  begin
    c:=b+a;
  end;
begin
  read(b);
  while b#0 do
  begin
    call p;
    write(2*c);
    read(b);
  end
end.
```

类P-code虚拟机

```
( 0) jmp 0 8      转向主程序入口
( 1) jmp 0 2      转向过程p入口
( 2) int 0 3      过程p入口,为过程p开辟空间
( 3) lod 1 3      取变量b的值到栈顶
( 4) lit 0 10     取常数10到栈顶
( 5) opr 0 2      次栈顶与栈顶相加
( 6) sto 1 4      栈顶值送变量c中
( 7) opr 0 0      退栈并返回调用点(16)
( 8) int 0 5      主程序入口开辟5个栈空间
( 9) opr 0 16     从命令行读入值置于栈顶
(10) sto 0 3      将栈顶值存入变量b中
(11) lod 0 3      将变量b的值取至栈顶
(12) lit 0 0      将常数值0进栈
(13) opr 0 9      次栈顶与栈顶是否不等
(14) jpc 0 24     相等时转(24)（条件不满足转)
(15) cal 0 2      调用过程p
(16) lit 0 2      常数值2进栈
(17) lod 0 4      将变量c的值取至栈顶
(18) opr 0 4      次栈顶与栈顶相乘(2*c)
(19) opr 0 14     栈顶值输出至屏幕
(20) opr 0 15     换行
(21) opr 0 16     从命令行读取值到栈顶
(22) sto 0 3      栈顶值送变量b中
(23) jmp 0 11     无条件转到循环入口(11)
(24) opr 0 0      结束退栈
```

p →



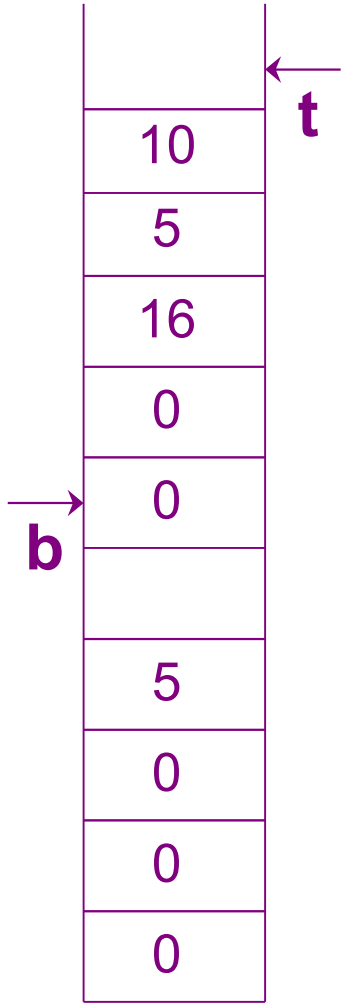
✧ 例

```
const  a=10;
var    b,c;
procedure p;
  begin
    c:=b+a;
  end;
begin
  read(b);
  while b#0 do
  begin
    call p;
    write(2*c);
    read(b);
  end
end.
```

类P-code虚拟机

```
( 0) jmp 0 8      转向主程序入口
( 1) jmp 0 2      转向过程p入口
( 2) int 0 3      过程p入口,为过程p开辟空间
( 3) lod 1 3      取变量b的值到栈顶
( 4) lit 0 10     取常数10到栈顶
( 5) opr 0 2      次栈顶与栈顶相加
( 6) sto 1 4      栈顶值送变量c中
( 7) opr 0 0      退栈并返回调用点(16)
( 8) int 0 5      主程序入口开辟5个栈空间
( 9) opr 0 16     从命令行读入值置于栈顶
(10) sto 0 3      将栈顶值存入变量b中
(11) lod 0 3      将变量b的值取至栈顶
(12) lit 0 0      将常数值0进栈
(13) opr 0 9      次栈顶与栈顶是否不等
(14) jpc 0 24     相等时转(24)（条件不满足转)
(15) cal 0 2      调用过程p
(16) lit 0 2      常数值2进栈
(17) lod 0 4      将变量c的值取至栈顶
(18) opr 0 4      次栈顶与栈顶相乘(2*c)
(19) opr 0 14     栈顶值输出至屏幕
(20) opr 0 15     换行
(21) opr 0 16     从命令行读取值到栈顶
(22) sto 0 3      栈顶值送变量b中
(23) jmp 0 11     无条件转到循环入口(11)
(24) opr 0 0      结束退栈
```

p →



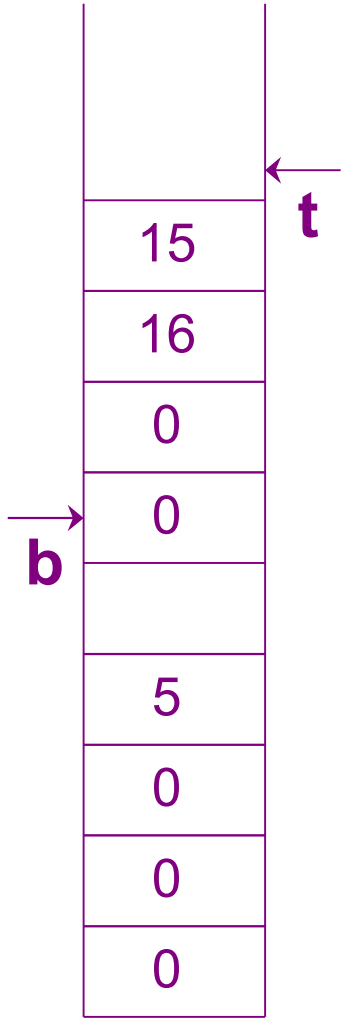
✧ 例

```
const  a=10;
var    b,c;
procedure p;
  begin
    c:=b+a;
  end;
begin
  read(b);
  while b#0 do
  begin
    call p;
    write(2*c);
    read(b);
  end
end.
```

类P-code虚拟机

```
( 0) jmp 0 8     转向主程序入口
( 1) jmp 0 2     转向过程p入口
( 2) int 0 3     过程p入口,为过程p开辟空间
( 3) lod 1 3     取变量b的值到栈顶
( 4) lit 0 10    取常数10到栈顶
( 5) opr 0 2     次栈顶与栈顶相加
( 6) sto 1 4     栈顶值送变量c中
( 7) opr 0 0     退栈并返回调用点(16)
( 8) int 0 5     主程序入口开辟5个栈空间
( 9) opr 0 16    从命令行读入值置于栈顶
(10) sto 0 3     将栈顶值存入变量b中
(11) lod 0 3     将变量b的值取至栈顶
(12) lit 0 0     将常数值0进栈
(13) opr 0 9     次栈顶与栈顶是否不等
(14) jpc 0 24    相等时转(24)（条件不满足转)
(15) cal 0 2     调用过程p
(16) lit 0 2     常数值2进栈
(17) lod 0 4     将变量c的值取至栈顶
(18) opr 0 4     次栈顶与栈顶相乘(2*c)
(19) opr 0 14    栈顶值输出至屏幕
(20) opr 0 15    换行
(21) opr 0 16    从命令行读取值到栈顶
(22) sto 0 3     栈顶值送变量b中
(23) jmp 0 11    无条件转到循环入口(11)
(24) opr 0 0     结束退栈
```

p →



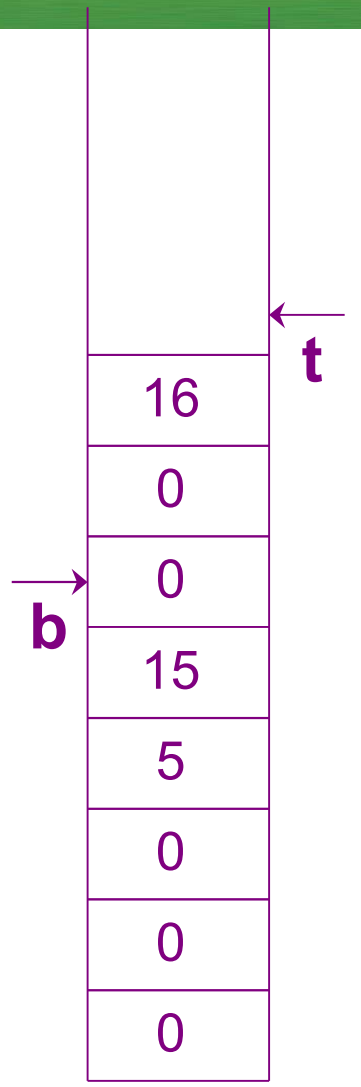
✧ 例

```
const  a=10;
var    b,c;
procedure p;
  begin
    c:=b+a;
  end;
begin
  read(b);
  while b#0 do
  begin
    call p;
    write(2*c);
    read(b);
  end
end.
```

类P-code虚拟机

```
( 0) jmp 0 8      转向主程序入口
( 1) jmp 0 2      转向过程p入口
( 2) int 0 3      过程p入口,为过程p开辟空间
( 3) lod 1 3      取变量b的值到栈顶
( 4) lit 0 10     取常数10到栈顶
( 5) opr 0 2      次栈顶与栈顶相加
( 6) sto 1 4      栈顶值送变量c中
( 7) opr 0 0      退栈并返回调用点(16)
( 8) int 0 5      主程序入口开辟5个栈空间
( 9) opr 0 16     从命令行读入值置于栈顶
(10) sto 0 3      将栈顶值存入变量b中
(11) lod 0 3      将变量b的值取至栈顶
(12) lit 0 0      将常数值0进栈
(13) opr 0 9      次栈顶与栈顶是否不等
(14) jpc 0 24     相等时转(24)（条件不满足转)
(15) cal 0 2      调用过程p
(16) lit 0 2      常数值2进栈
(17) lod 0 4      将变量c的值取至栈顶
(18) opr 0 4      次栈顶与栈顶相乘(2*c)
(19) opr 0 14     栈顶值输出至屏幕
(20) opr 0 15     换行
(21) opr 0 16     从命令行读取值到栈顶
(22) sto 0 3      栈顶值送变量b中
(23) jmp 0 11     无条件转到循环入口(11)
(24) opr 0 0      结束退栈
```

p →



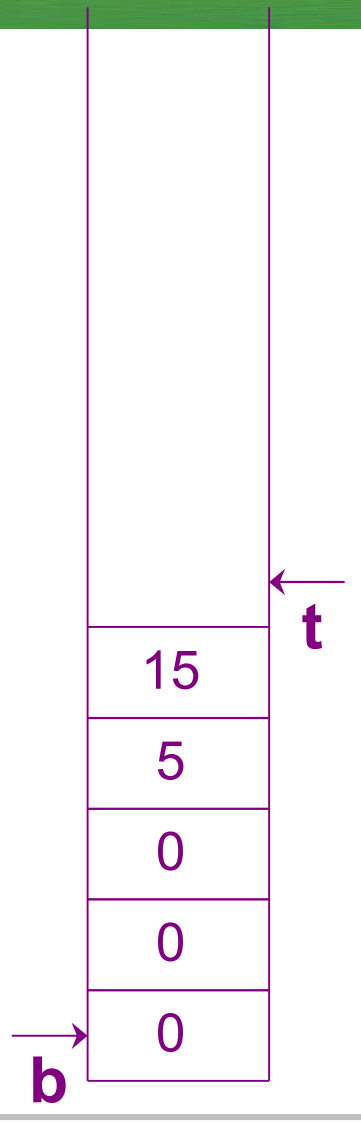
✧ 例

```
const  a=10;
var    b,c;
procedure p;
  begin
    c:=b+a;
  end;
begin
  read(b);
  while b#0 do
  begin
    call p;
    write(2*c);
    read(b);
  end
end.
```

类P-code虚拟机

```
( 0) jmp 0 8      转向主程序入口
( 1) jmp 0 2      转向过程p入口
( 2) int 0 3      过程p入口,为过程p开辟空间
( 3) lod 1 3      取变量b的值到栈顶
( 4) lit 0 10     取常数10到栈顶
( 5) opr 0 2      次栈顶与栈顶相加
( 6) sto 1 4      栈顶值送变量c中
( 7) opr 0 0      退栈并返回调用点(16)
( 8) int 0 5      主程序入口开辟5个栈空间
( 9) opr 0 16     从命令行读入值置于栈顶
(10) sto 0 3      将栈顶值存入变量b中
(11) lod 0 3      将变量b的值取至栈顶
(12) lit 0 0      将常数值0进栈
(13) opr 0 9      次栈顶与栈顶是否不等
(14) jpc 0 24     相等时转(24)（条件不满足转）
(15) cal 0 2      调用过程p
(16) lit 0 2      常数值2进栈
(17) lod 0 4      将变量c的值取至栈顶
(18) opr 0 4      次栈顶与栈顶相乘(2*c)
(19) opr 0 14     栈顶值输出至屏幕
(20) opr 0 15     换行
(21) opr 0 16     从命令行读取值到栈顶
(22) sto 0 3      栈顶值送变量b中
(23) jmp 0 11     无条件转到循环入口(11)
(24) opr 0 0      结束退栈
```

p →



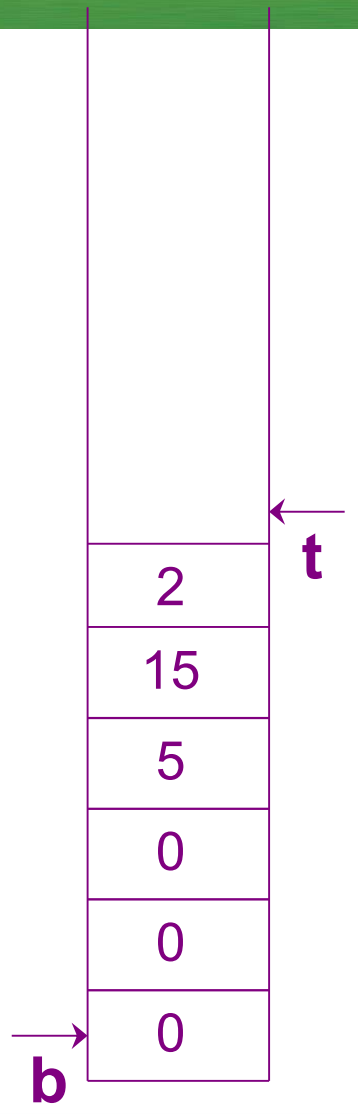
✧ 例

```
const  a=10;
var    b,c;
procedure p;
  begin
    c:=b+a;
  end;
begin
  read(b);
  while b#0 do
  begin
    call p;
    write(2*c);
    read(b);
  end
end.
```

类P-code虚拟机

```
( 0) jmp 0 8      转向主程序入口
( 1) jmp 0 2      转向过程p入口
( 2) int 0 3      过程p入口,为过程p开辟空间
( 3) lod 1 3      取变量b的值到栈顶
( 4) lit 0 10     取常数10到栈顶
( 5) opr 0 2      次栈顶与栈顶相加
( 6) sto 1 4      栈顶值送变量c中
( 7) opr 0 0      退栈并返回调用点(16)
( 8) int 0 5      主程序入口开辟5个栈空间
( 9) opr 0 16     从命令行读入值置于栈顶
(10) sto 0 3      将栈顶值存入变量b中
(11) lod 0 3      将变量b的值取至栈顶
(12) lit 0 0      将常数值0进栈
(13) opr 0 9      次栈顶与栈顶是否不等
(14) jpc 0 24     相等时转(24)（条件不满足转)
(15) cal 0 2      调用过程p
(16) lit 0 2      常数值2进栈
(17) lod 0 4      将变量c的值取至栈顶
(18) opr 0 4      次栈顶与栈顶相乘(2*c)
(19) opr 0 14     栈顶值输出至屏幕
(20) opr 0 15     换行
(21) opr 0 16     从命令行读取值到栈顶
(22) sto 0 3      栈顶值送变量b中
(23) jmp 0 11     无条件转到循环入口(11)
(24) opr 0 0      结束退栈
```

p →



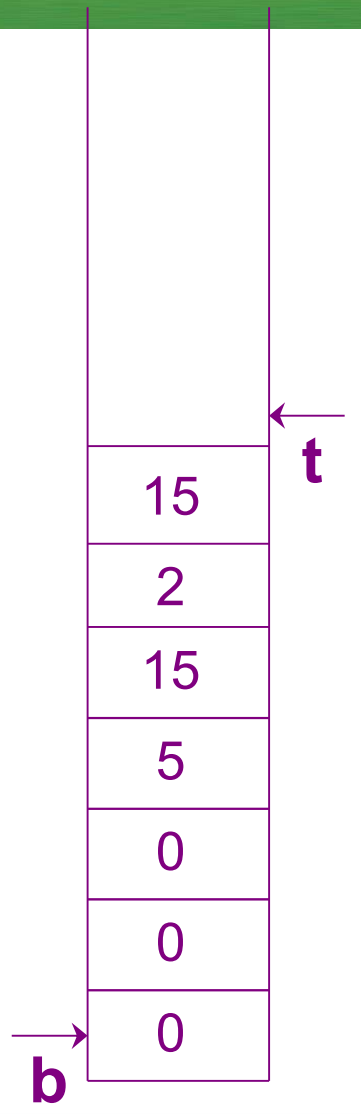
✧ 例

```
const  a=10;
var    b,c;
procedure p;
  begin
    c:=b+a;
  end;
begin
  read(b);
  while b#0 do
  begin
    call p;
    write(2*c);
    read(b);
  end
end.
```

类P-code虚拟机

```
( 0) jmp 0 8      转向主程序入口
( 1) jmp 0 2      转向过程p入口
( 2) int 0 3      过程p入口,为过程p开辟空间
( 3) lod 1 3      取变量b的值到栈顶
( 4) lit 0 10     取常数10到栈顶
( 5) opr 0 2      次栈顶与栈顶相加
( 6) sto 1 4      栈顶值送变量c中
( 7) opr 0 0      退栈并返回调用点(16)
( 8) int 0 5      主程序入口开辟5个栈空间
( 9) opr 0 16     从命令行读入值置于栈顶
(10) sto 0 3      将栈顶值存入变量b中
(11) lod 0 3      将变量b的值取至栈顶
(12) lit 0 0      将常数值0进栈
(13) opr 0 9      次栈顶与栈顶是否不等
(14) jpc 0 24     相等时转(24)（条件不满足转)
(15) cal 0 2      调用过程p
(16) lit 0 2      常数值2进栈
(17) lod 0 4      将变量c的值取至栈顶
(18) opr 0 4      次栈顶与栈顶相乘(2*c)
(19) opr 0 14     栈顶值输出至屏幕
(20) opr 0 15     换行
(21) opr 0 16     从命令行读取值到栈顶
(22) sto 0 3      栈顶值送变量b中
(23) jmp 0 11     无条件转到循环入口(11)
(24) opr 0 0      结束退栈
```

p →



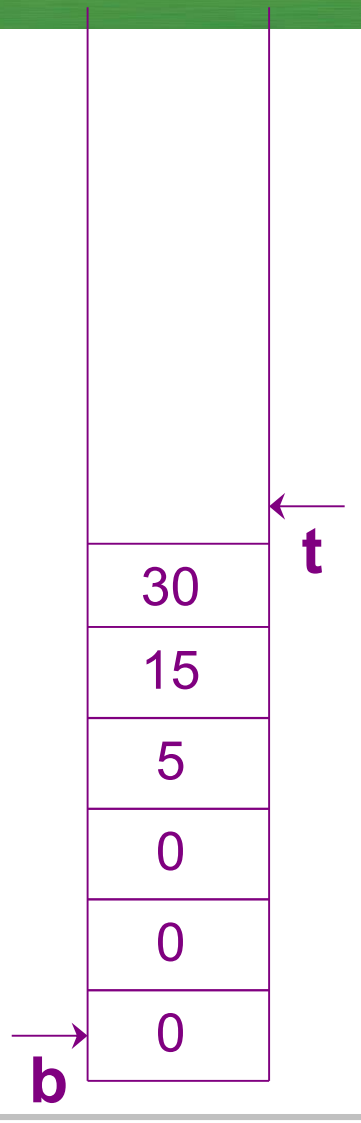
✧ 例

```
const  a=10;
var    b,c;
procedure p;
  begin
    c:=b+a;
  end;
begin
  read(b);
  while b#0 do
  begin
    call p;
    write(2*c);
    read(b);
  end
end.
```

类P-code虚拟机

```
( 0) jmp 0 8      转向主程序入口
( 1) jmp 0 2      转向过程p入口
( 2) int 0 3      过程p入口,为过程p开辟空间
( 3) lod 1 3      取变量b的值到栈顶
( 4) lit 0 10     取常数10到栈顶
( 5) opr 0 2      次栈顶与栈顶相加
( 6) sto 1 4      栈顶值送变量c中
( 7) opr 0 0      退栈并返回调用点(16)
( 8) int 0 5      主程序入口开辟5个栈空间
( 9) opr 0 16     从命令行读入值置于栈顶
(10) sto 0 3      将栈顶值存入变量b中
(11) lod 0 3      将变量b的值取至栈顶
(12) lit 0 0      将常数值0进栈
(13) opr 0 9      次栈顶与栈顶是否不等
(14) jpc 0 24     相等时转(24)（条件不满足转)
(15) cal 0 2      调用过程p
(16) lit 0 2      常数值2进栈
(17) lod 0 4      将变量c的值取至栈顶
(18) opr 0 4      次栈顶与栈顶相乘(2*c)
(19) opr 0 14     栈顶值输出至屏幕
(20) opr 0 15     换行
(21) opr 0 16     从命令行读取值到栈顶
(22) sto 0 3      栈顶值送变量b中
(23) jmp 0 11     无条件转到循环入口(11)
(24) opr 0 0      结束退栈
```

p →



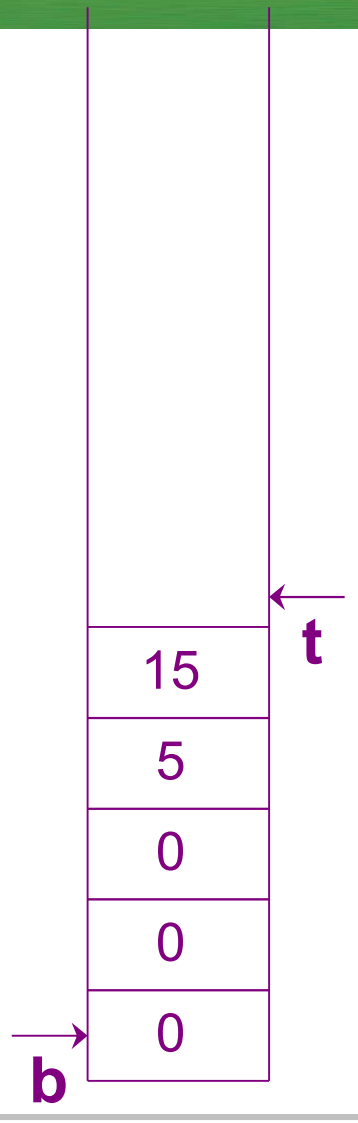
✧ 例

```
const  a=10;
var    b,c;
procedure p;
  begin
    c:=b+a;
  end;
begin
  read(b);
  while b#0 do
  begin
    call p;
    write(2*c);
    read(b);
  end
end.
```

类P-code虚拟机

```
( 0) jmp 0 8      转向主程序入口
( 1) jmp 0 2      转向过程p入口
( 2) int 0 3      过程p入口,为过程p开辟空间
( 3) lod 1 3      取变量b的值到栈顶
( 4) lit 0 10     取常数10到栈顶
( 5) opr 0 2      次栈顶与栈顶相加
( 6) sto 1 4      栈顶值送变量c中
( 7) opr 0 0      退栈并返回调用点(16)
( 8) int 0 5      主程序入口开辟5个栈空间
( 9) opr 0 16     从命令行读入值置于栈顶
(10) sto 0 3      将栈顶值存入变量b中
(11) lod 0 3      将变量b的值取至栈顶
(12) lit 0 0      将常数值0进栈
(13) opr 0 9      次栈顶与栈顶是否不等
(14) jpc 0 24     相等时转(24)（条件不满足转)
(15) cal 0 2      调用过程p
(16) lit 0 2      常数值2进栈
(17) lod 0 4      将变量c的值取至栈顶
(18) opr 0 4      次栈顶与栈顶相乘(2*c)
(19) opr 0 14     栈顶值输出至屏幕
(20) opr 0 15     换行
(21) opr 0 16     从命令行读取值到栈顶
(22) sto 0 3      栈顶值送变量b中
(23) jmp 0 11     无条件转到循环入口(11)
(24) opr 0 0      结束退栈
```

p →



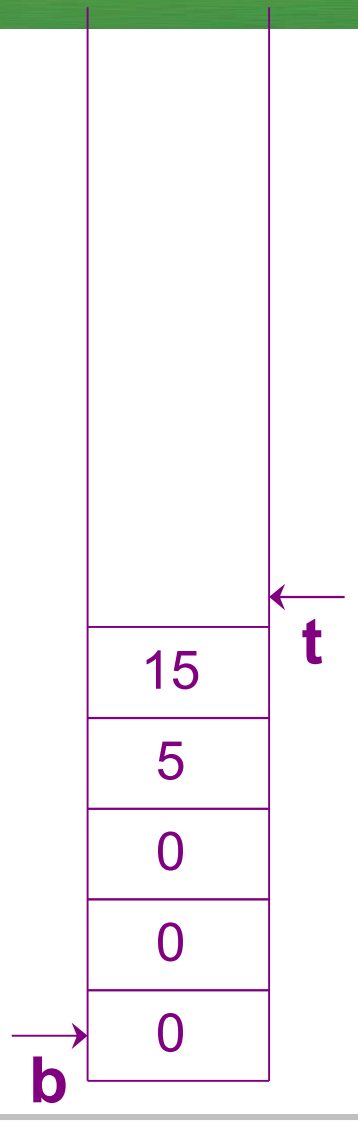
✧ 例

```
const  a=10;
var    b,c;
procedure p;
  begin
    c:=b+a;
  end;
begin
  read(b);
  while b#0 do
  begin
    call p;
    write(2*c);
    read(b);
  end
end.
```

类P-code虚拟机

```
( 0) jmp 0 8      转向主程序入口
( 1) jmp 0 2      转向过程p入口
( 2) int 0 3      过程p入口,为过程p开辟空间
( 3) lod 1 3      取变量b的值到栈顶
( 4) lit 0 10     取常数10到栈顶
( 5) opr 0 2      次栈顶与栈顶相加
( 6) sto 1 4      栈顶值送变量c中
( 7) opr 0 0      退栈并返回调用点(16)
( 8) int 0 5      主程序入口开辟5个栈空间
( 9) opr 0 16     从命令行读入值置于栈顶
(10) sto 0 3      将栈顶值存入变量b中
(11) lod 0 3      将变量b的值取至栈顶
(12) lit 0 0      将常数值0进栈
(13) opr 0 9      次栈顶与栈顶是否不等
(14) jpc 0 24     相等时转(24)（条件不满足转）
(15) cal 0 2      调用过程p
(16) lit 0 2      常数值2进栈
(17) lod 0 4      将变量c的值取至栈顶
(18) opr 0 4      次栈顶与栈顶相乘(2*c)
(19) opr 0 14     栈顶值输出至屏幕
(20) opr 0 15     换行
(21) opr 0 16     从命令行读取值到栈顶
(22) sto 0 3      栈顶值送变量b中
(23) jmp 0 11     无条件转到循环入口(11)
(24) opr 0 0      结束退栈
```

p →



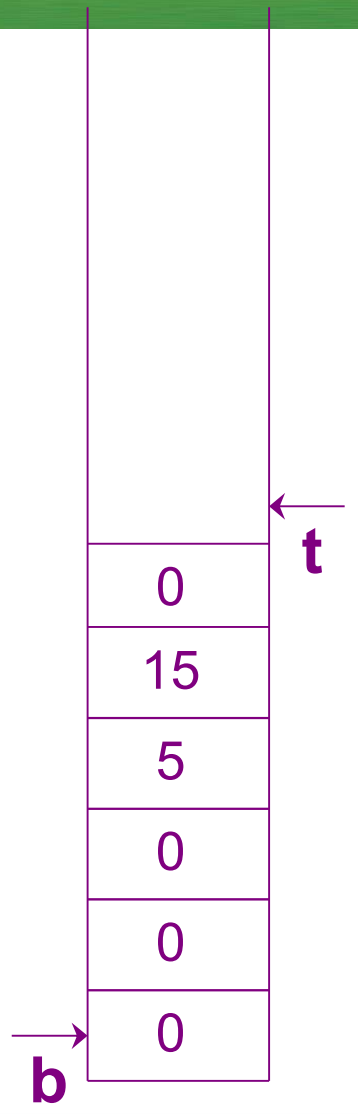
✧ 例

```
const  a=10;
var    b,c;
procedure p;
  begin
    c:=b+a;
  end;
begin
  read(b);
  while b#0 do
  begin
    call p;
    write(2*c);
    read(b);
  end
end.
```

类P-code虚拟机

```
( 0) jmp 0 8      转向主程序入口
( 1) jmp 0 2      转向过程p入口
( 2) int 0 3      过程p入口,为过程p开辟空间
( 3) lod 1 3      取变量b的值到栈顶
( 4) lit 0 10     取常数10到栈顶
( 5) opr 0 2      次栈顶与栈顶相加
( 6) sto 1 4      栈顶值送变量c中
( 7) opr 0 0      退栈并返回调用点(16)
( 8) int 0 5      主程序入口开辟5个栈空间
( 9) opr 0 16     从命令行读入值置于栈顶
(10) sto 0 3      将栈顶值存入变量b中
(11) lod 0 3      将变量b的值取至栈顶
(12) lit 0 0      将常数值0进栈
(13) opr 0 9      次栈顶与栈顶是否不等
(14) jpc 0 24     相等时转(24)（条件不满足转)
(15) cal 0 2      调用过程p
(16) lit 0 2      常数值2进栈
(17) lod 0 4      将变量c的值取至栈顶
(18) opr 0 4      次栈顶与栈顶相乘(2*c)
(19) opr 0 14     栈顶值输出至屏幕
(20) opr 0 15     换行
(21) opr 0 16     从命令行读取值到栈顶
(22) sto 0 3      栈顶值送变量b中
(23) jmp 0 11     无条件转到循环入口(11)
(24) opr 0 0      结束退栈
```

p →



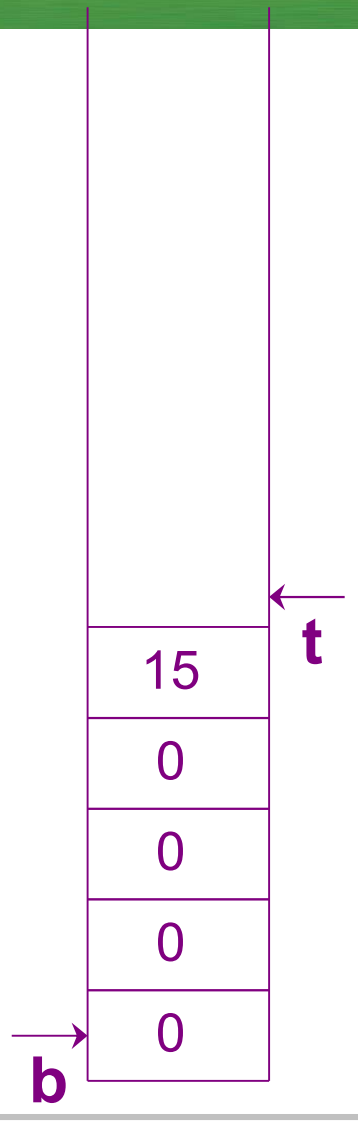
✧ 例

```
const  a=10;
var    b,c;
procedure p;
  begin
    c:=b+a;
  end;
begin
  read(b);
  while b#0 do
  begin
    call p;
    write(2*c);
    read(b);
  end
end.
```

类P-code虚拟机

```
( 0) jmp 0 8      转向主程序入口
( 1) jmp 0 2      转向过程p入口
( 2) int 0 3      过程p入口,为过程p开辟空间
( 3) lod 1 3      取变量b的值到栈顶
( 4) lit 0 10     取常数10到栈顶
( 5) opr 0 2      次栈顶与栈顶相加
( 6) sto 1 4      栈顶值送变量c中
( 7) opr 0 0      退栈并返回调用点(16)
( 8) int 0 5      主程序入口开辟5个栈空间
( 9) opr 0 16     从命令行读入值置于栈顶
(10) sto 0 3      将栈顶值存入变量b中
(11) lod 0 3      将变量b的值取至栈顶
(12) lit 0 0      将常数值0进栈
(13) opr 0 9      次栈顶与栈顶是否不等
(14) jpc 0 24     相等时转(24)（条件不满足转)
(15) cal 0 2      调用过程p
(16) lit 0 2      常数值2进栈
(17) lod 0 4      将变量c的值取至栈顶
(18) opr 0 4      次栈顶与栈顶相乘(2*c)
(19) opr 0 14     栈顶值输出至屏幕
(20) opr 0 15     换行
(21) opr 0 16     从命令行读取值到栈顶
(22) sto 0 3      栈顶值送变量b中
(23) jmp 0 11     无条件转到循环入口(11)
(24) opr 0 0      结束退栈
```

p →



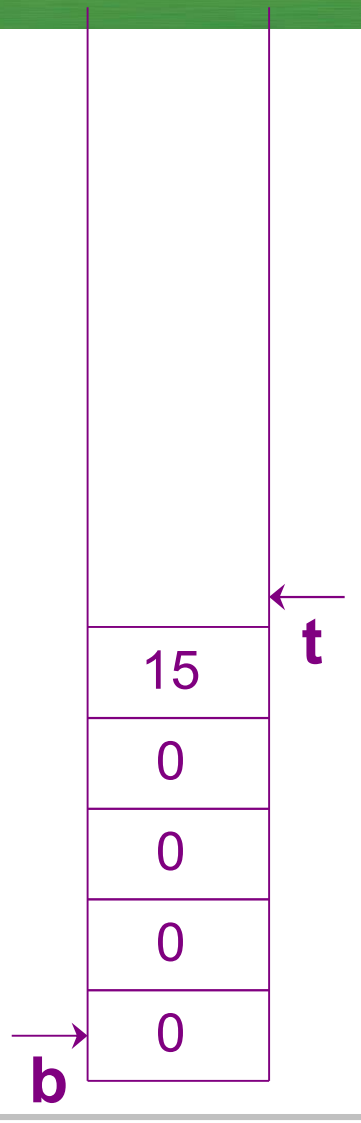
✧ 例

```
const  a=10;
var    b,c;
procedure p;
  begin
    c:=b+a;
  end;
begin
  read(b);
  while b#0 do
  begin
    call p;
    write(2*c);
    read(b);
  end
end.
```

类P-code虚拟机

```
( 0) jmp 0 8      转向主程序入口
( 1) jmp 0 2      转向过程p入口
( 2) int 0 3      过程p入口,为过程p开辟空间
( 3) lod 1 3      取变量b的值到栈顶
( 4) lit 0 10     取常数10到栈顶
( 5) opr 0 2      次栈顶与栈顶相加
( 6) sto 1 4      栈顶值送变量c中
( 7) opr 0 0      退栈并返回调用点(16)
( 8) int 0 5      主程序入口开辟5个栈空间
( 9) opr 0 16     从命令行读入值置于栈顶
(10) sto 0 3      将栈顶值存入变量b中
(11) lod 0 3      将变量b的值取至栈顶
(12) lit 0 0      将常数值0进栈
(13) opr 0 9      次栈顶与栈顶是否不等
(14) jpc 0 24     相等时转(24)（条件不满足转）
(15) cal 0 2      调用过程p
(16) lit 0 2      常数值2进栈
(17) lod 0 4      将变量c的值取至栈顶
(18) opr 0 4      次栈顶与栈顶相乘(2*c)
(19) opr 0 14     栈顶值输出至屏幕
(20) opr 0 15     换行
(21) opr 0 16     从命令行读取值到栈顶
(22) sto 0 3      栈顶值送变量b中
(23) jmp 0 11     无条件转到循环入口(11)
(24) opr 0 0      结束退栈
```

p →



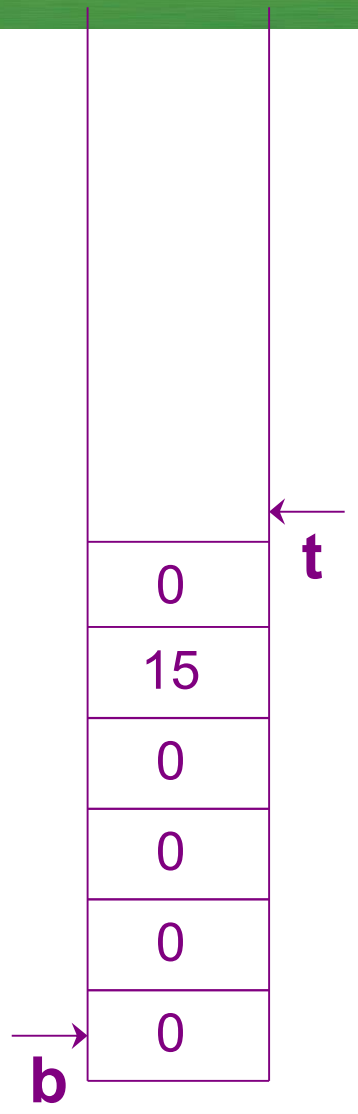
✧ 例

```
const  a=10;
var    b,c;
procedure p;
  begin
    c:=b+a;
  end;
begin
  read(b);
  while b#0 do
  begin
    call p;
    write(2*c);
    read(b);
  end
end.
```

类P-code虚拟机

```
( 0) jmp 0 8      转向主程序入口
( 1) jmp 0 2      转向过程p入口
( 2) int 0 3      过程p入口,为过程p开辟空间
( 3) lod 1 3      取变量b的值到栈顶
( 4) lit 0 10     取常数10到栈顶
( 5) opr 0 2      次栈顶与栈顶相加
( 6) sto 1 4      栈顶值送变量c中
( 7) opr 0 0      退栈并返回调用点(16)
( 8) int 0 5      主程序入口开辟5个栈空间
( 9) opr 0 16     从命令行读入值置于栈顶
(10) sto 0 3      将栈顶值存入变量b中
(11) lod 0 3      将变量b的值取至栈顶
(12) lit 0 0      将常数值0进栈
(13) opr 0 9      次栈顶与栈顶是否不等
(14) jpc 0 24     相等时转(24)（条件不满足转）
(15) cal 0 2      调用过程p
(16) lit 0 2      常数值2进栈
(17) lod 0 4      将变量c的值取至栈顶
(18) opr 0 4      次栈顶与栈顶相乘(2*c)
(19) opr 0 14     栈顶值输出至屏幕
(20) opr 0 15     换行
(21) opr 0 16     从命令行读取值到栈顶
(22) sto 0 3      栈顶值送变量b中
(23) jmp 0 11     无条件转到循环入口(11)
(24) opr 0 0      结束退栈
```

p →



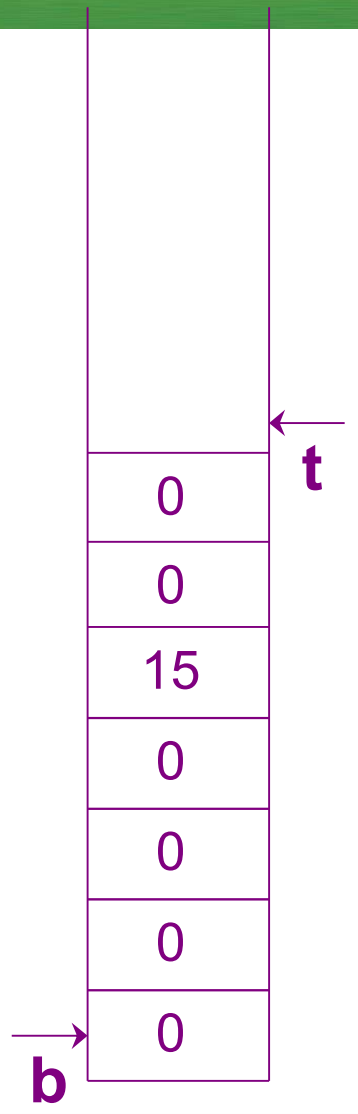
✧ 例

```
const  a=10;
var    b,c;
procedure p;
  begin
    c:=b+a;
  end;
begin
  read(b);
  while b#0 do
  begin
    call p;
    write(2*c);
    read(b);
  end
end.
```

类P-code虚拟机

```
( 0) jmp 0 8      转向主程序入口
( 1) jmp 0 2      转向过程p入口
( 2) int 0 3      过程p入口,为过程p开辟空间
( 3) lod 1 3      取变量b的值到栈顶
( 4) lit 0 10     取常数10到栈顶
( 5) opr 0 2      次栈顶与栈顶相加
( 6) sto 1 4      栈顶值送变量c中
( 7) opr 0 0      退栈并返回调用点(16)
( 8) int 0 5      主程序入口开辟5个栈空间
( 9) opr 0 16     从命令行读入值置于栈顶
(10) sto 0 3      将栈顶值存入变量b中
(11) lod 0 3      将变量b的值取至栈顶
(12) lit 0 0      将常数值0进栈
(13) opr 0 9      次栈顶与栈顶是否不等
(14) jpc 0 24     相等时转(24)（条件不满足转）
(15) cal 0 2      调用过程p
(16) lit 0 2      常数值2进栈
(17) lod 0 4      将变量c的值取至栈顶
(18) opr 0 4      次栈顶与栈顶相乘(2*c)
(19) opr 0 14     栈顶值输出至屏幕
(20) opr 0 15     换行
(21) opr 0 16     从命令行读取值到栈顶
(22) sto 0 3      栈顶值送变量b中
(23) jmp 0 11     无条件转到循环入口(11)
(24) opr 0 0      结束退栈
```

p →



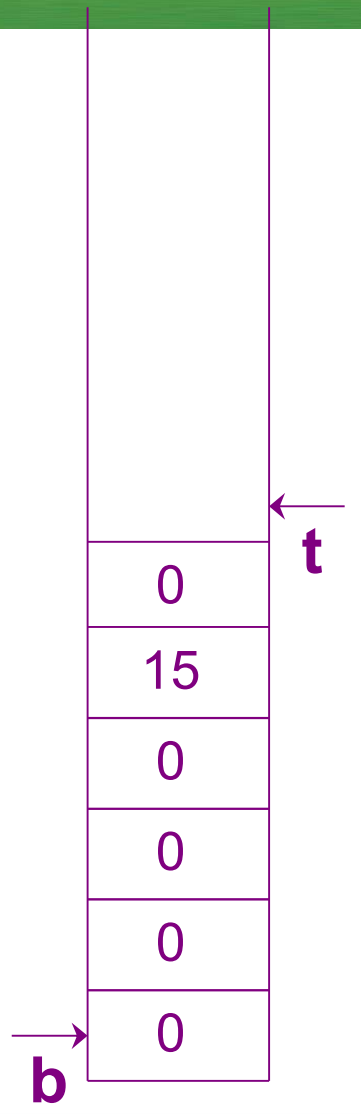
✧ 例

```
const  a=10;
var    b,c;
procedure p;
  begin
    c:=b+a;
  end;
begin
  read(b);
  while b#0 do
  begin
    call p;
    write(2*c);
    read(b);
  end
end.
```

类P-code虚拟机

```
( 0) jmp 0 8      转向主程序入口
( 1) jmp 0 2      转向过程p入口
( 2) int 0 3      过程p入口,为过程p开辟空间
( 3) lod 1 3      取变量b的值到栈顶
( 4) lit 0 10     取常数10到栈顶
( 5) opr 0 2      次栈顶与栈顶相加
( 6) sto 1 4      栈顶值送变量c中
( 7) opr 0 0      退栈并返回调用点(16)
( 8) int 0 5      主程序入口开辟5个栈空间
( 9) opr 0 16     从命令行读入值置于栈顶
(10) sto 0 3      将栈顶值存入变量b中
(11) lod 0 3      将变量b的值取至栈顶
(12) lit 0 0      将常数值0进栈
(13) opr 0 9      次栈顶与栈顶是否不等
(14) jpc 0 24     相等时转(24)（条件不满足转）
(15) cal 0 2      调用过程p
(16) lit 0 2      常数值2进栈
(17) lod 0 4      将变量c的值取至栈顶
(18) opr 0 4      次栈顶与栈顶相乘(2*c)
(19) opr 0 14     栈顶值输出至屏幕
(20) opr 0 15     换行
(21) opr 0 16     从命令行读取值到栈顶
(22) sto 0 3      栈顶值送变量b中
(23) jmp 0 11     无条件转到循环入口(11)
(24) opr 0 0      结束退栈
```

p →



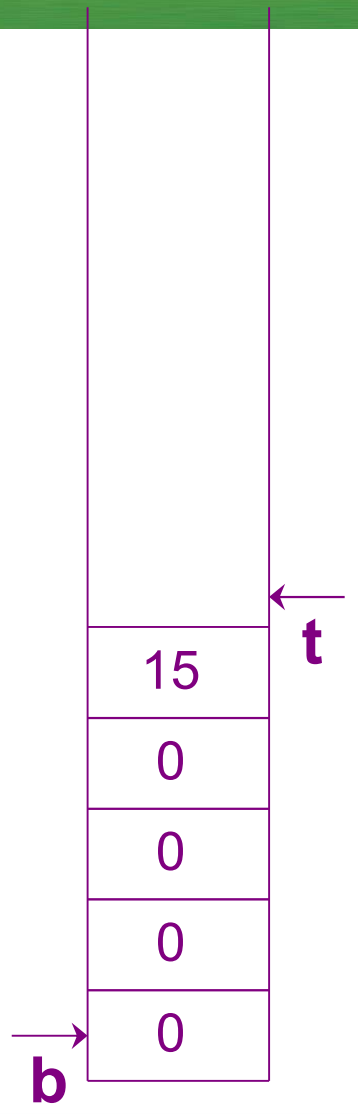
✧ 例

```
const  a=10;
var    b,c;
procedure p;
  begin
    c:=b+a;
  end;
begin
  read(b);
  while b#0 do
  begin
    call p;
    write(2*c);
    read(b);
  end
end.
```

类P-code虚拟机

```
( 0) jmp 0 8      转向主程序入口
( 1) jmp 0 2      转向过程p入口
( 2) int 0 3      过程p入口,为过程p开辟空间
( 3) lod 1 3      取变量b的值到栈顶
( 4) lit 0 10     取常数10到栈顶
( 5) opr 0 2      次栈顶与栈顶相加
( 6) sto 1 4      栈顶值送变量c中
( 7) opr 0 0      退栈并返回调用点(16)
( 8) int 0 5      主程序入口开辟5个栈空间
( 9) opr 0 16     从命令行读入值置于栈顶
(10) sto 0 3      将栈顶值存入变量b中
(11) lod 0 3      将变量b的值取至栈顶
(12) lit 0 0      将常数值0进栈
(13) opr 0 9      次栈顶与栈顶是否不等
(14) jpc 0 24     相等时转(24)（条件不满足转)
(15) cal 0 2      调用过程p
(16) lit 0 2      常数值2进栈
(17) lod 0 4      将变量c的值取至栈顶
(18) opr 0 4      次栈顶与栈顶相乘(2*c)
(19) opr 0 14     栈顶值输出至屏幕
(20) opr 0 15     换行
(21) opr 0 16     从命令行读取值到栈顶
(22) sto 0 3      栈顶值送变量b中
(23) jmp 0 11     无条件转到循环入口(11)
(24) opr 0 0      结束退栈
```

p →

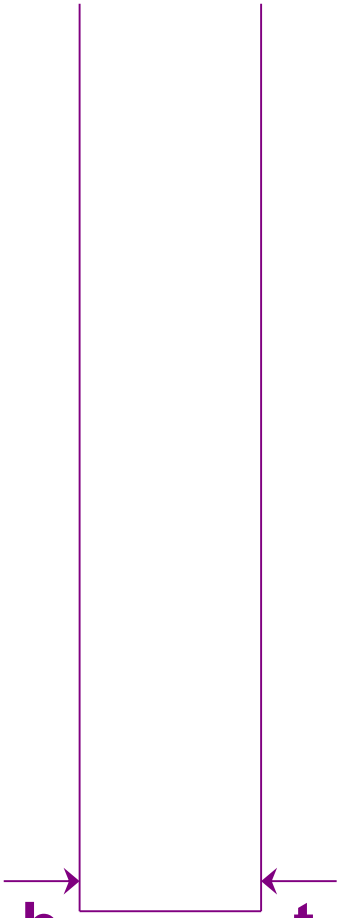


✧ 例

```
const  a=10;
var    b,c;
procedure p;
  begin
    c:=b+a;
  end;
begin
  read(b);
  while b#0 do
  begin
    call p;
    write(2*c);
    read(b);
  end
end.
```

类P-code虚拟机

```
( 0) jmp 0 8      转向主程序入口
( 1) jmp 0 2      转向过程p入口
( 2) int 0 3      过程p入口,为过程p开辟空间
( 3) lod 1 3      取变量b的值到栈顶
( 4) lit 0 10     取常数10到栈顶
( 5) opr 0 2      次栈顶与栈顶相加
( 6) sto 1 4      栈顶值送变量c中
( 7) opr 0 0      退栈并返回调用点(16)
( 8) int 0 5      主程序入口开辟5个栈空间
( 9) opr 0 16     从命令行读入值置于栈顶
(10) sto 0 3      将栈顶值存入变量b中
(11) lod 0 3      将变量b的值取至栈顶
(12) lit 0 0      将常数值0进栈
(13) opr 0 9      次栈顶与栈顶是否不等
(14) jpc 0 24     相等时转(24)（条件不满足转)
(15) cal 0 2      调用过程p
(16) lit 0 2      常数值2进栈
(17) lod 0 4      将变量c的值取至栈顶
(18) opr 0 4      次栈顶与栈顶相乘(2*c)
(19) opr 0 14     栈顶值输出至屏幕
(20) opr 0 15     换行
(21) opr 0 16     从命令行读取值到栈顶
(22) sto 0 3      栈顶值送变量b中
(23) jmp 0 11     无条件转到循环入口(11)
(24) opr 0 0      结束退栈
```



✧ 例

```
const  a=10;
var    b,c;
procedure p;
  begin
    c:=b+a;
  end;
begin
  read(b);
  while b#0 do
  begin
    call p;
    write(2*c);
    read(b);
  end
end.
```