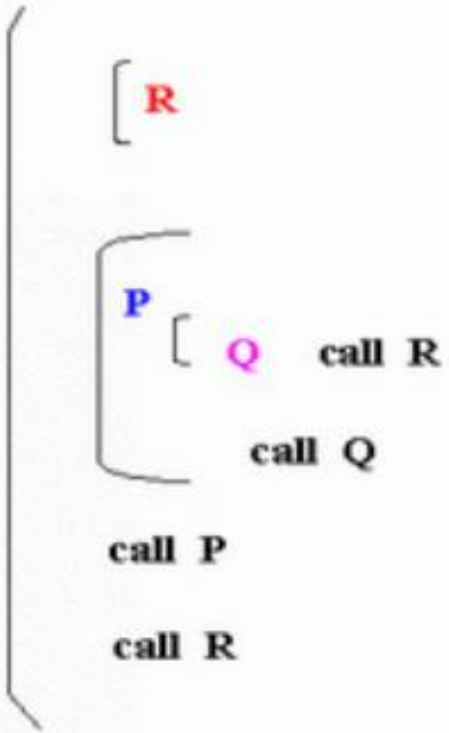
Chapter8

Running-Time Storage Management

Example

```
program main(i,0);
  ......
proc    R(c,d);
    ......
end /*R*/
proc   P (a);
     ......
     proc  Q (b);
       ......
         R(x,y);
     end   /*Q*/
       ......
    Q(z);
end  /*P*/
......
P (W) ;
 ......
R (U, V) ;
......
end /*main*/
```

Program structure



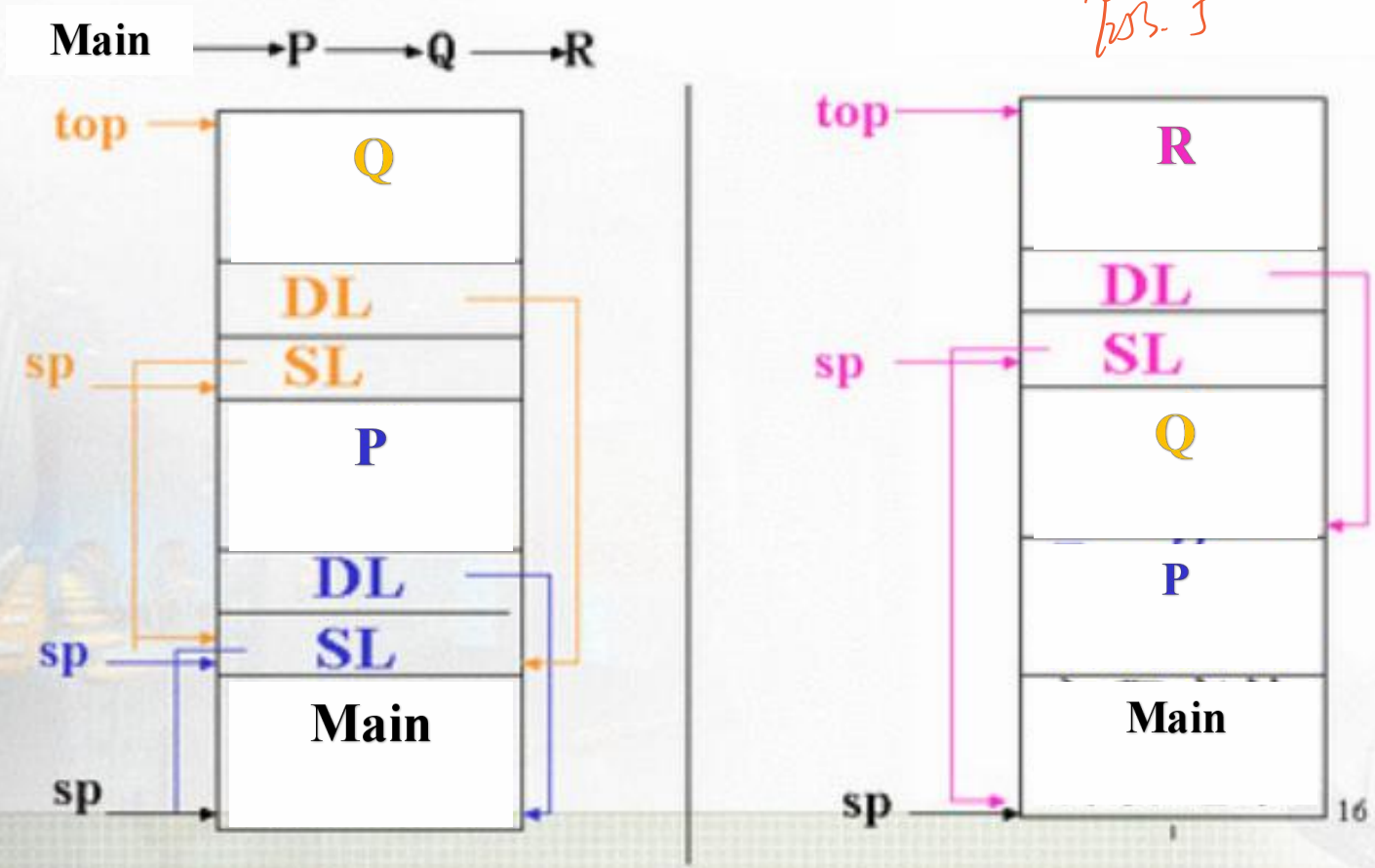
Main → P → Q → R

Running stack

动态链：可变置地址记存器

P82.2

P83.5



```
(1)program  reference(input,output);
(2)var  a,b:integer;
(3)procedure  swap(VAR  x,y:integer);
(4)        var  temp:integer;
(5)        begin
(6)              temp:=x;
(7)                   x:=y;
(8)                  y:=temp
(9)        end;
(10)procedure exchange(i,j:integer);
(11)      var x:integer;
(12)      begin;
(13)         x:=i; i:=j;  j:=x
(14)      end;
(15)begin
(16)       a:=1;  b:=2;
(17)       swap(a,b);
(18)       writeln('a=',a);writeln('b=',b)
(19)       exchange(a,b);
(20)       writeln('a=',a);writeln('b=',b)
(21)end.
```

```
(1)swap(x,y)
(2)int   *x,*y;
(3){         int   temp;
(4)              temp=*x;  *x=*y;  *y=temp;
(5)}
(6)main(        )
(7){    int   a=1,b=2;
(8)         swap(&a,&b);
(9)         printf("a is now %d,b is now %d\n",a,b);
(10)}
```

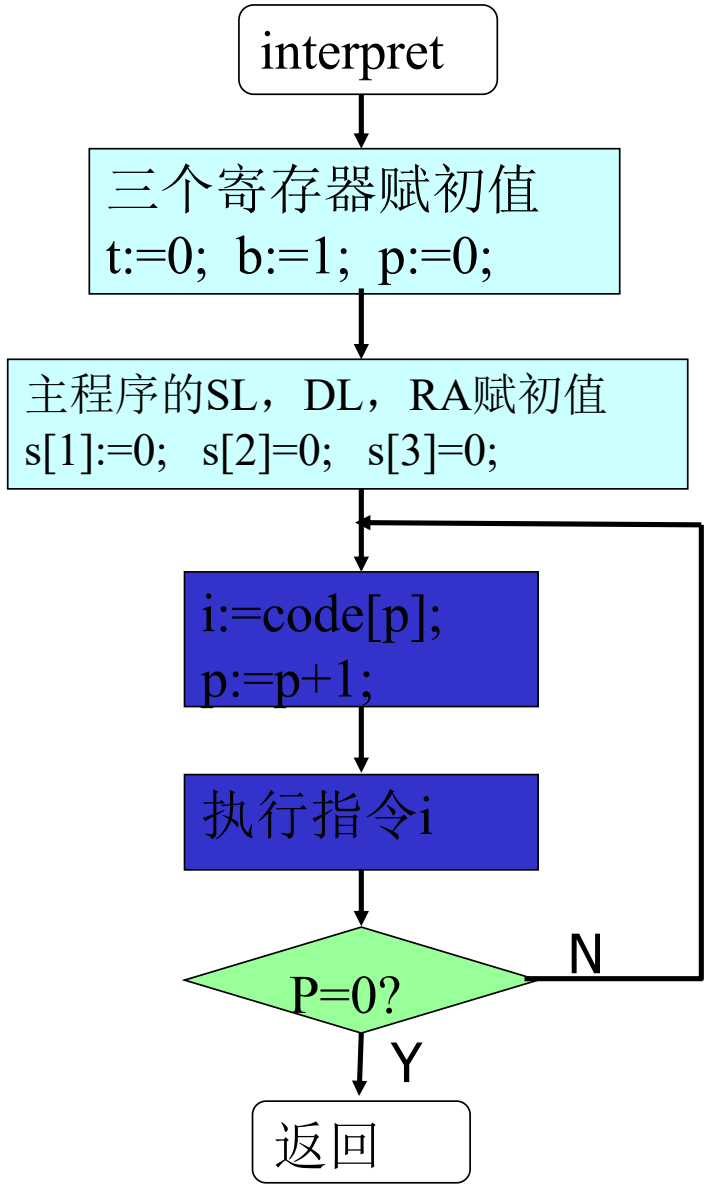
Pointer as a parameter is equivalent to transfer address in C language

PL/0编译程序的运行时存储组织

在每个过程调用时在栈顶分配3个联系单元：

- SL： 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。
- DL： 动态链，指向调用该过程前正在运行过程的数据段基地址。
- RA： 返回地址，记录调用该过程时目标程序的断点，即调用过程指令（call语句）的下一条指令的地址。

解释执行的流程图


interpret
三个寄存器赋初值
t:=0; b:=1; p:=0;
主程序的SL，DL，RA赋初值
s[1]:=0; s[2]=0; s[3]=0;
i:=code[p];
p:=p+1;
执行指令i
P=0?
N
Y
返回
主程序
的RA
s[3]=0

目标代码的解释执行

调用过程：

```
cal: begin (*generat new block mark*)
        s[t+1]:=base(l);  填写静态链
        s[t+2]:=b;        填写动态链
        s[t+3]:=p;        填写返回地址
        b:=t+1;      被调用过程的基地址
        p:=a          过程入口地址a送p
     end;
```

目标代码的解释执行

```
function base(l:integer): integer;
  var b1:integer;
   begin  b1:=b; (*find base l level down*)
      while l>0 do
         begin
            b1:=s[b1]; l:=l-1;
         end;
      base:=b1
   end (*base*);
```

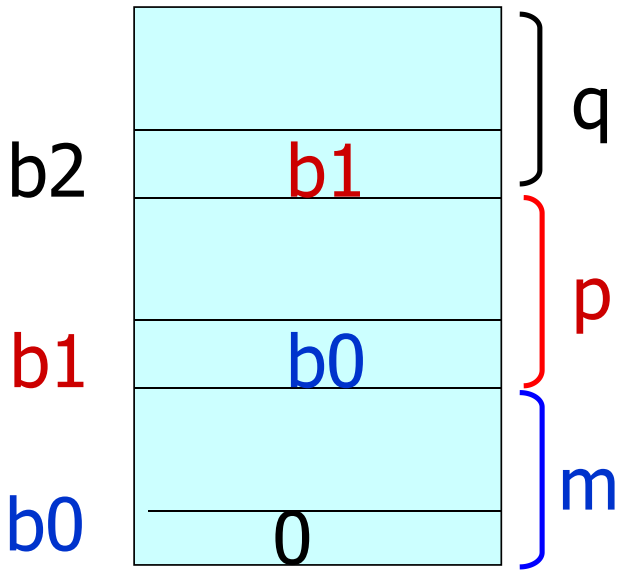
目标代码的解释执行

- **base (l:integer): integer;**

例：

主程序m中定义过程p

过程p中定义过程q



目标代码的解释执行

- 几条特殊指令的解释执行：

过程入口：开辟a个单元（见教材304页）

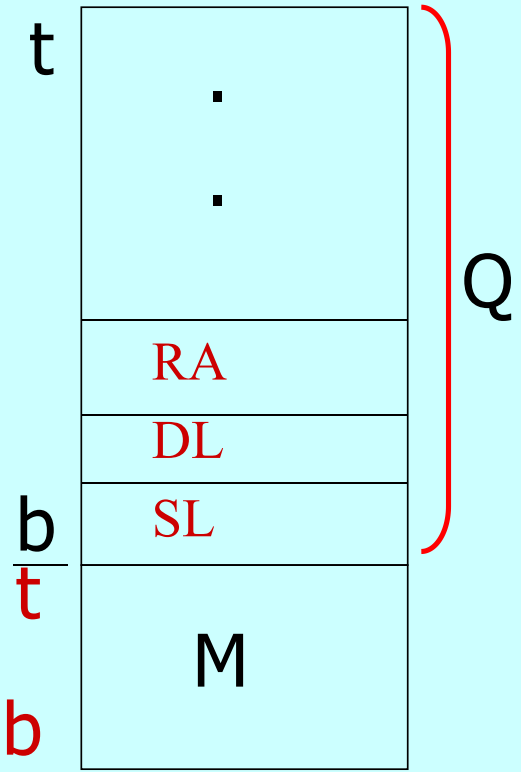
int: t:=t+a; （ t是当前栈顶值）

过程出口：释放数据段（退栈）（见教材302页）

```
opr: case a of  (*operator*)
          0: begin (*return*)
               t:=b-1;        恢复调用前栈顶
               p:=s[t+3];   送返回地址到p(RA)
               b:=s[t+2]   恢复调用前基地址(DL)
                end;
```

目标代码的解释执行

- 过程出口




```
t:=b-1;
p:=s[t+3];
b:=s[t+2]
```

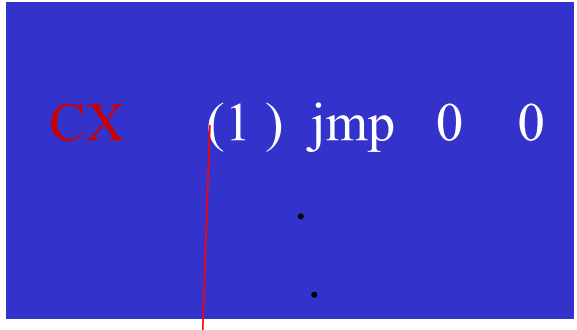
```
CONST   a=10;
VAR     b,c;
PROCEDURE p;
   BEGIN
      c:=b+a;
   END;
BEGIN
   READ(b);
   WHILE b#0 DO
   BEGIN
      CALL p;
      WRITE(2*c);
      READ(b);
   END
END.
```

```
( 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
```

Table表

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

记录 过程在code的入口到table中的adr域



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

PL/0代码生成

过程体入口时的处理

```
code[table[tx0].adr].a:=cx;
      (*过程入口地址填 写在code中*)
with table[tx0] do
  begin
    adr:=cx;   (*过程的入口填 写在table中*)
    size:=dx;  (*过程占的空间填 写在table中*)
  end;
cxo:=cx;     (*保留过程在code中的入口地址*)
gen(int,0,dx);  (*生成过程入口指令*)
```

```
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
```