编译原理

第九章 运行时存储空间组织

第九章 运行时存储空间组织

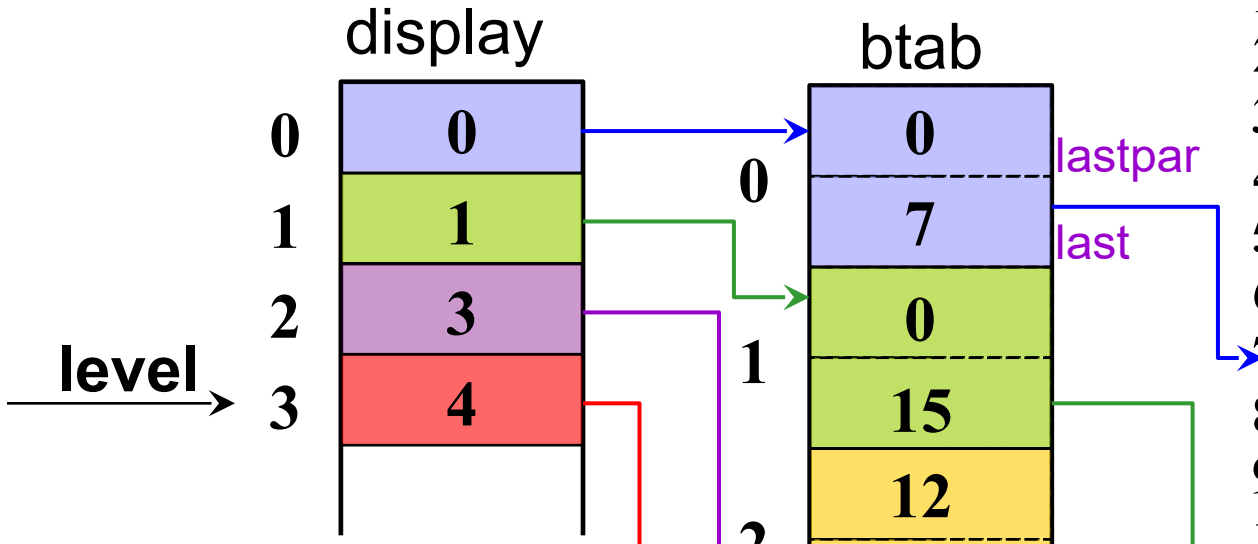
- 目标程序运行时的活动
- 运行时存储器的划分
- 静态存储管理
- 一个简单栈式存储分配
- 嵌套过程语言的栈式实现

第九章 运行时存储空间组织

- 目标程序运行时的活动
- 运行时存储器的划分
- 静态存储管理
- 一个简单栈式存储分配
- 嵌套过程语言的栈式实现

PL 语言的中间代码

- 指令格式： opcod l a
  - opcod： 操作码
  - l： 第一操作数，程序体层数
  - a： 第二操作数，相对地址
- 编译时如何确定变量的层数 (地址)？




```
program P;
  var  a,b : integer;
  procedure P1 (i1, j1:integer);
     var  c,d : integer
     …
  end;
  procedure P2 (i2, j2:integer);
     var  a,c : integer;
     procedure P21;
       var  b1,b2 : boolean;
          …
     end;
     …
  end;
    …
end.
```

b1 := a + b

nametab

	name	link
0		
1	char	0
2	integer	1
3	boolean	2
4	false	3
5	true	4
6	read	5
7	write	6
8	a	0
9	b	8
10	P1	9
11	i1	0
12	j1	11
13	c	12
14	d	13
15	P2	10
16	i2	0
17	j2	16
18	a	17
19	c	18
20	P21	19
21	b1	0
22	b2	21
23		

PL 语言的中间代码

- 指令格式： opcod l a
  - opcod： 操作码
  - l： 第一操作数，程序体层数
  - a： 第二操作数，相对地址
- 编译时如何确定变量的层数 (地址)？
- 运行时如何根据指令中变量的层数和相对地址确定变量的存储单元?

9.5 嵌套过程语言的栈式实现

- PASCAL
- 非局部名字的访问的实现
  - 静态链和活动记录
  - 嵌套层次显示表 display
- 过程调用、过程进入、过程返回

嵌套过程语言的栈式实现

- PASCAL
- 非局部名字的访问的实现
  - 静态链和活动记录
  - 嵌套层次显示表 display
- 过程调用、过程进入、过程返回

嵌套过程语言的栈式实现

- 假定语言不仅允许过程的递归调用（和可变数组），而且允许过程定义的嵌套，如PASCAL， PL 语言。

嵌套过程语言的栈式实现

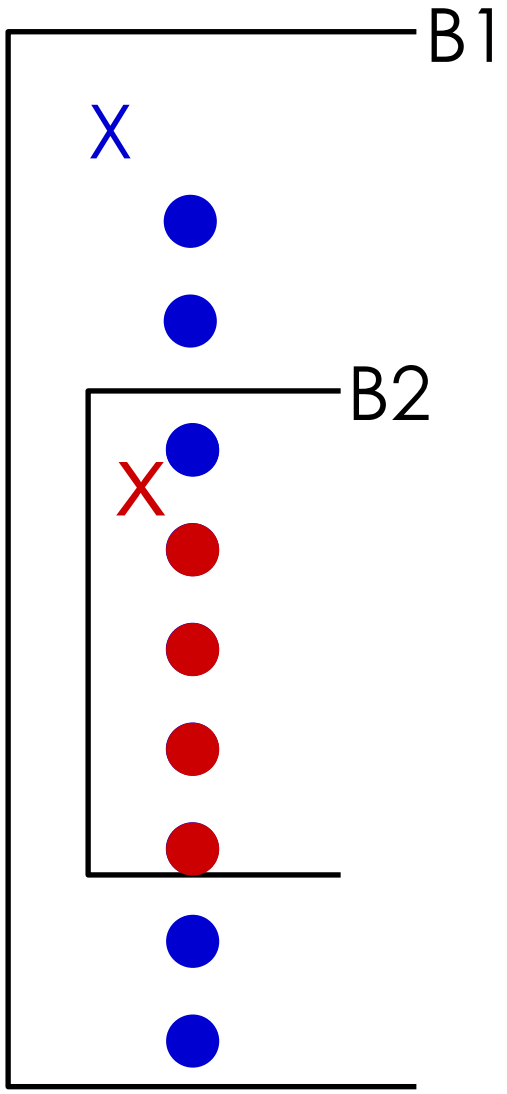
- PASCAL
  - PASCAL 程序本身可以看成是一个操作系统所调用的过程，过程可以嵌套和递归。
  - 一个 PASCAL 过程:

```
过程头;
    说明段（由一系列的说明语句组成）;
begin
    执行体（由一系列的执行语句组成）;
end
```

- 作用域: 一个名字能被使用的区域范围称作这个名字的作用域

最近嵌套原则

- 一个在子程序 B1 中说明的名字 X 只在 B1 中有效（局部于 B1）
- 如果 B2 是 B1 的一个内层子程序且 B2 中对标识符 X 没有新的说明，则原来的名字 X 在 B2 中仍然有效
- 如果 B2 对 X 重新作了说明，那么， B2 对 X 的任何引用都是指重新说明过的这个 X



嵌套过程语言的栈式实现

- PASCAL
- 非局部名字的访问的实现
  - 静态链和活动记录
  - 嵌套层次显示表 display
- 过程调用、过程进入、过程返回

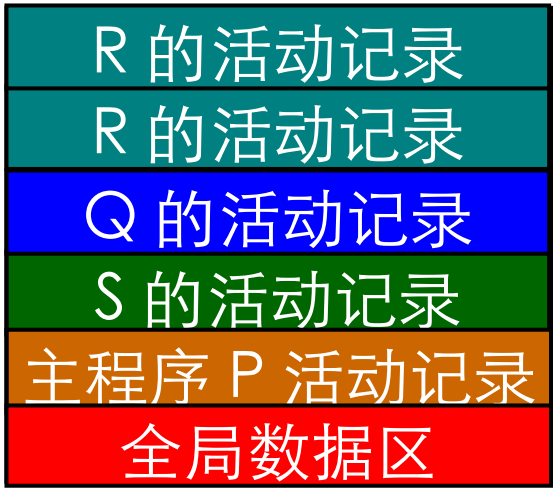
9.5.1 非局部名字的访问的实现

- 主程序的层次为 0；在 i 层中定义的过程，其层次为 i+1;

（层数， 偏移量）

图 9.15 程序

```
program P;
var a, x : integer;
procedure Q(b: integer);
    var i: integer;
    procedure R(u: integer; var v:
    integer);
    var c, d: integer;
    begin
        if u=1 then R(u+1, v)
        ......
        v:=(a+c)*(b-d);
        ......
    end {R}
begin
    ......
    R(1,x);
    ......
end {Q}
```



主程序 → 过程 S → 过程 Q → 过程 R → 过程 R

```
procedure S;
    var c, i:integer;
    begin
        a:=1;
        Q(c);
        ......
    end {S}
begin
    a:=0;
    S;
    ......
end. {P}
```

9.5.1 非局部名字的访问的实现

- 主程序的层次为 0；在 i 层中定义的过程，其层次为 i+1;

（层数， 偏移量）

- 过程运行时，必须知道其所有外层过程的当前活动记录的起始地址

嵌套过程语言的栈式实现

- PASCAL
- 非局部名字的访问的实现
  - 静态链和活动记录
  - 嵌套层次显示表 display
- 过程调用、过程进入、过程返回

一、静态链和活动记录

- 静态链：指向本过程的直接外层过程的活动记录的起始地址，也称存取链。
- 动态链：指向本过程的调用过程的活动记录的起始地址，也称控制链。

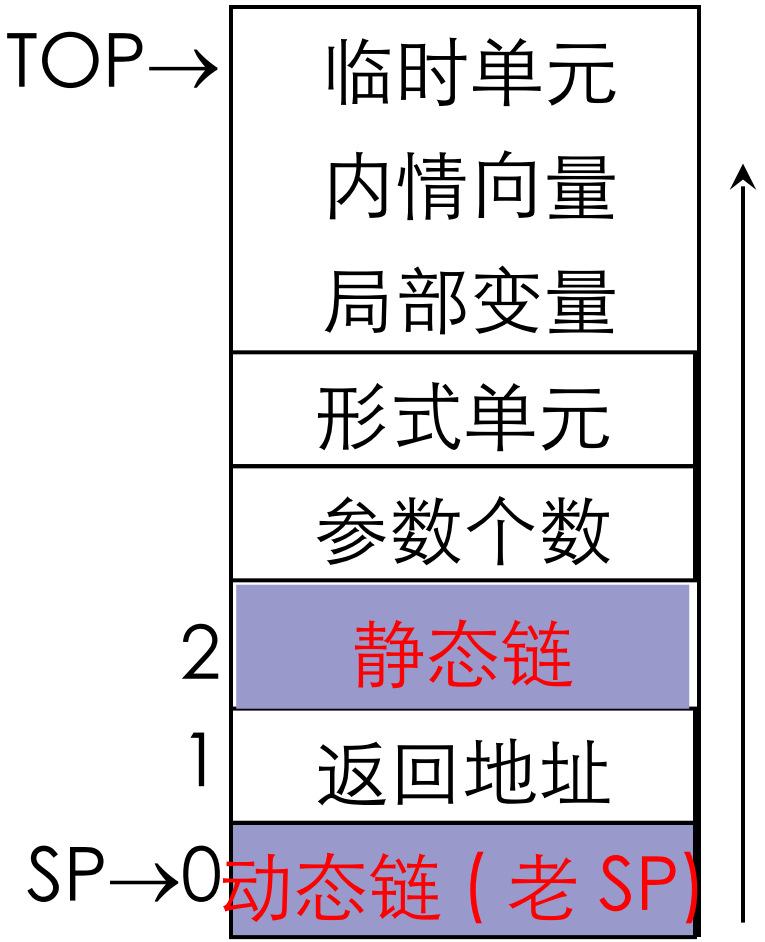


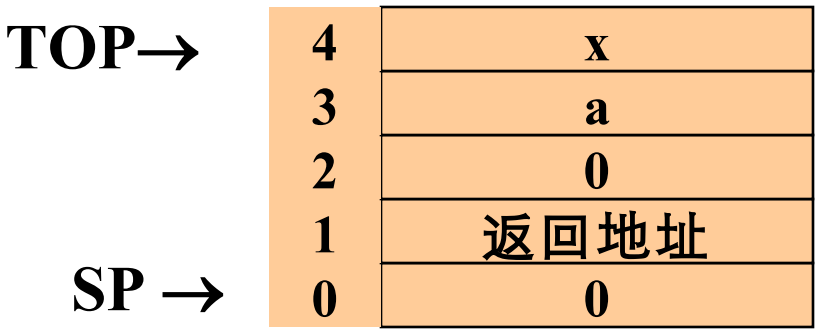
图 9.15 程序

```
program P;
var a, x : integer;
procedure Q(b: integer);
    var i: integer;
    procedure R(u: integer; var v:
    integer);
    var c, d: integer;
    begin
        if u=1 then R(u+1, v)
        ......
        v:=(a+c)*(b-d);
        ......
    end {R}
begin
    ......
    R(1,x);
    ......
end {Q}
```

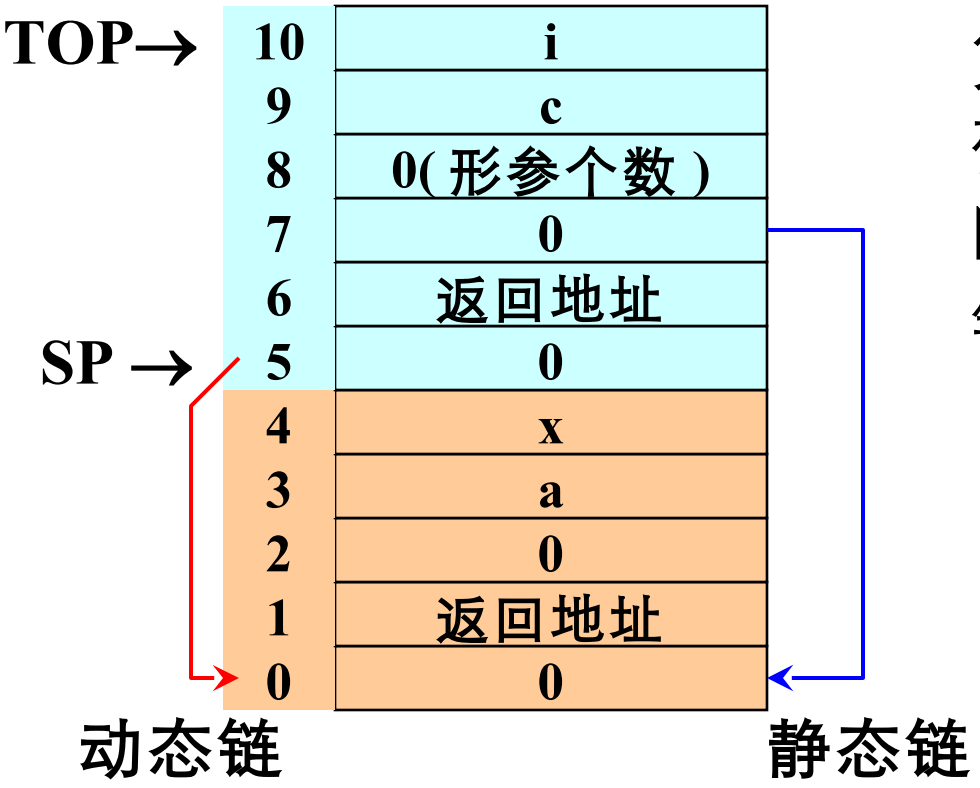
主程序 → 过程 S → 过程 Q → 过程 R → 过程 R

```
procedure S;
    var c, i:integer;
    begin
        a:=1;
        Q(c);
        ......
    end {S}
begin
    a:=0;
    S;
    ......
end. {P}
```

主程序 P



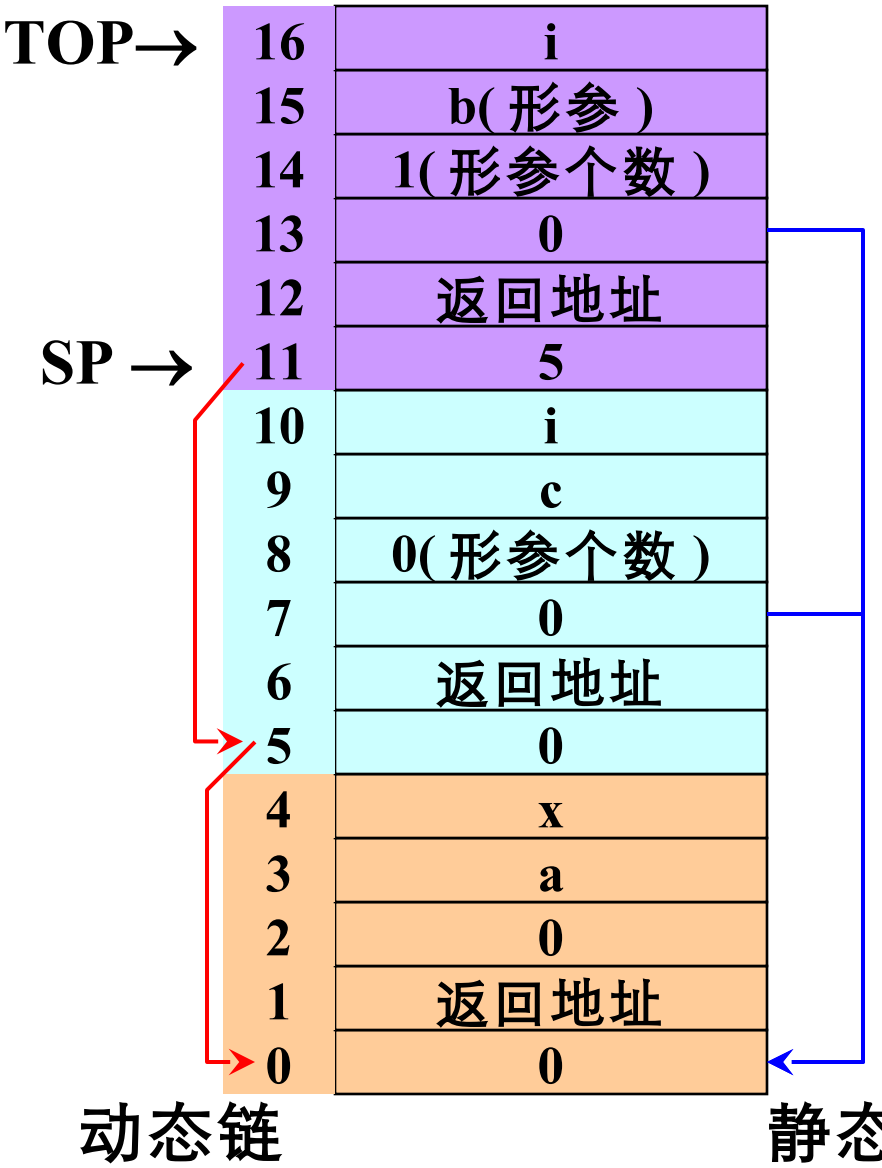
主程序 P→ 过程 S



? 第 N 层过程调用第 N+1 层过程，如何确定被调用过程（第 N+1 层）过程的静态链?

A: 调用过程（第 N 层过程）的最新活动记录的起始地址 .

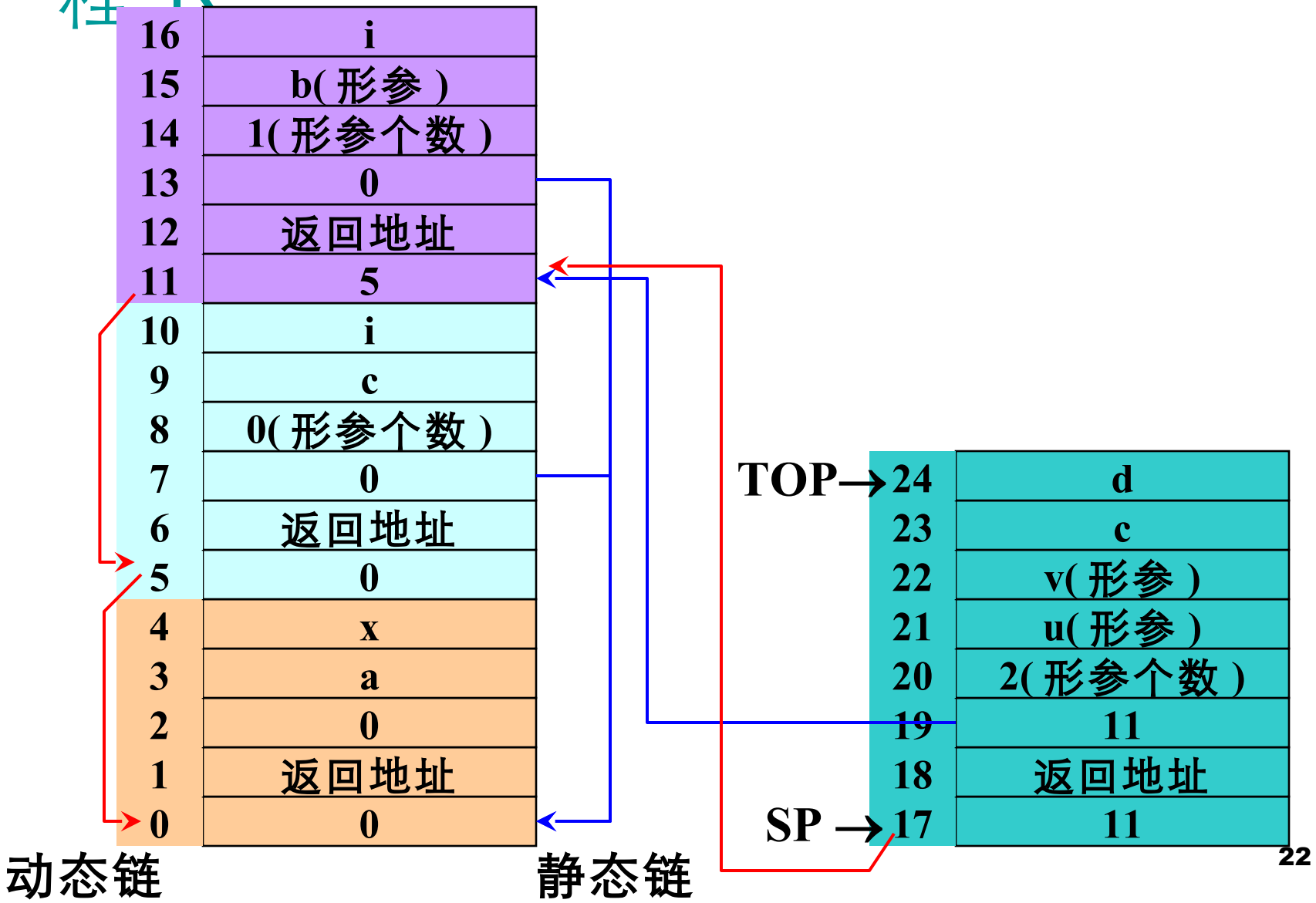
主程序 P → 过程 S → 过程 Q



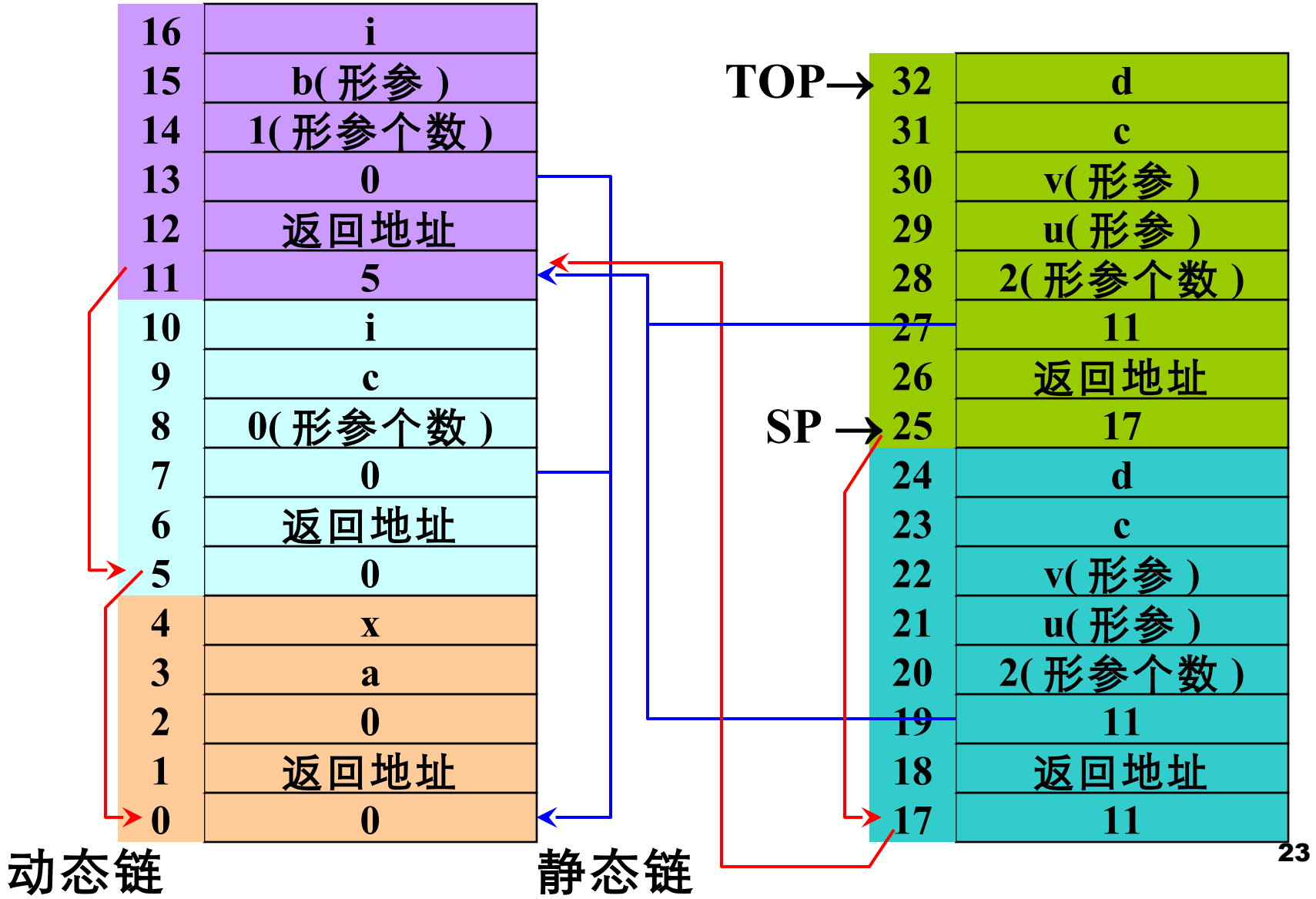
? 第 N 层过程调用第 N 层过程，如何确定被调用过程（第 N 层）过程的静态链?

A: 调用过程（第 N 层过程）的静态链的值。

- 主程序 P → 过程 S → 过程 Q → 过程 R



主程序 P → 过程 S → 过程 Q → 过程 R → 过程 R



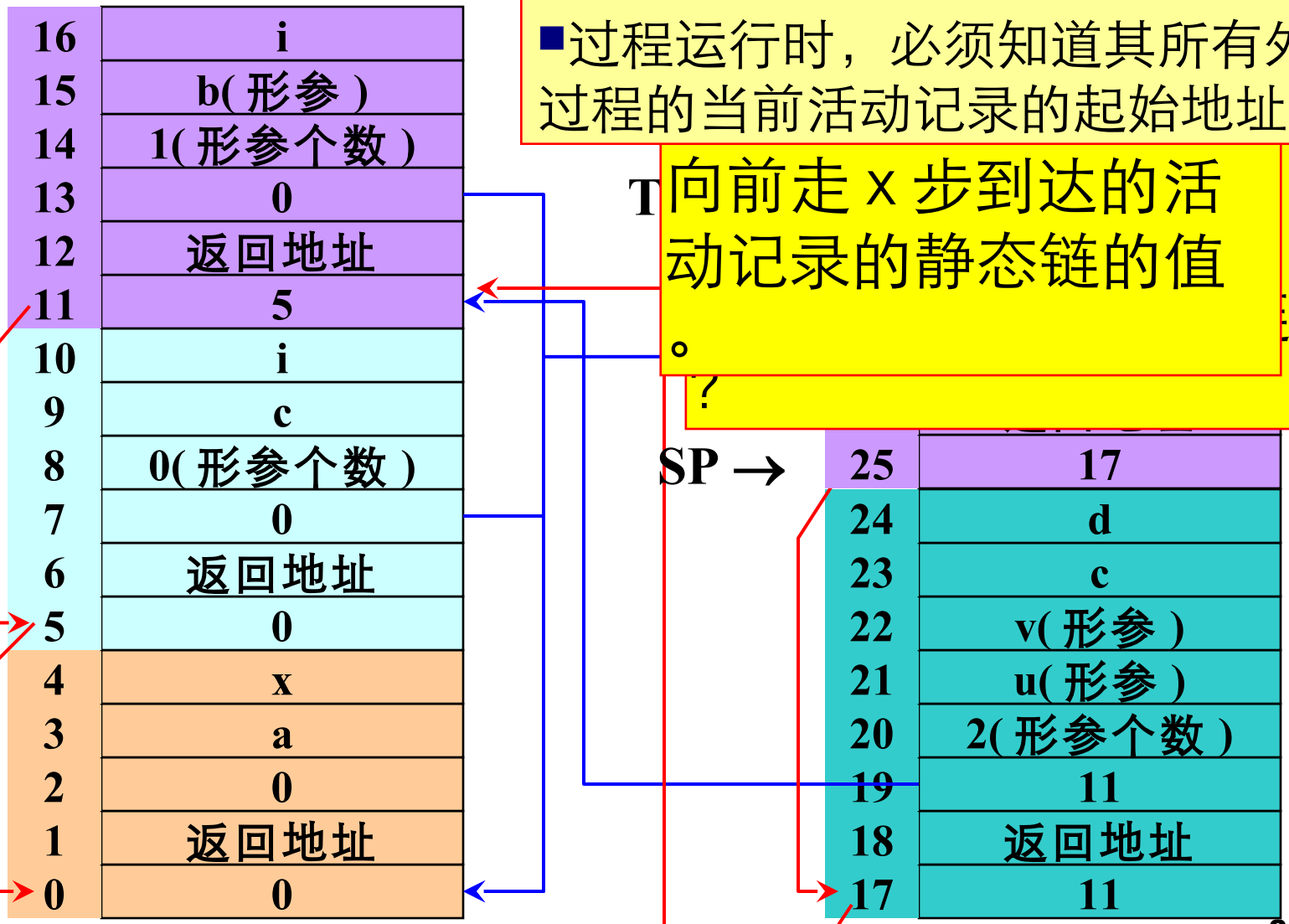
主程序 P → 过程 S → 过程 Q

- 主程序的层次为 0；在 i 层中定义的过程，其层次为 i+1;

（层数， 偏移量）

- 过程运行时，必须知道其所有外层过程的当前活动记录的起始地址

向前走 x 步到达的活动记录的静态链的值。



小结

- 嵌套过程语言的栈式实现
- 非局部名字的访问的实现
  - 静态链和活动记录

作业

- 《编译原理大作业实习》
  - 阅读
    - 第四节 PL 语义分析及中间代码产生