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

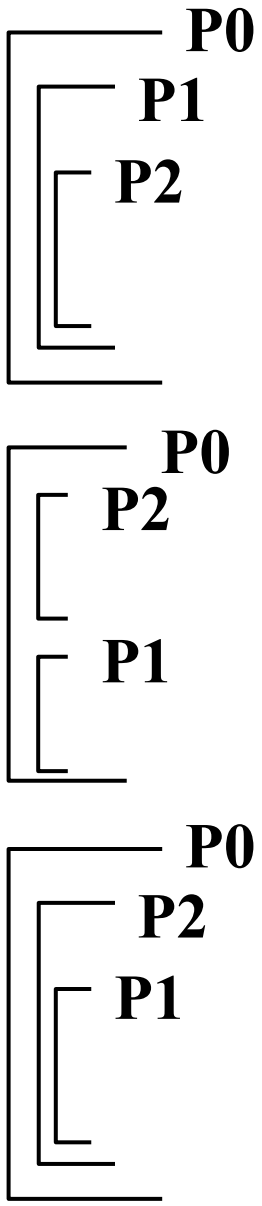
- 目标程序运行时的活动
- 运行时存储器的划分
- 静态存储管理
- 一个简单栈式存储分配
- 嵌套过程语言的栈式实现

嵌套过程语言的栈式实现

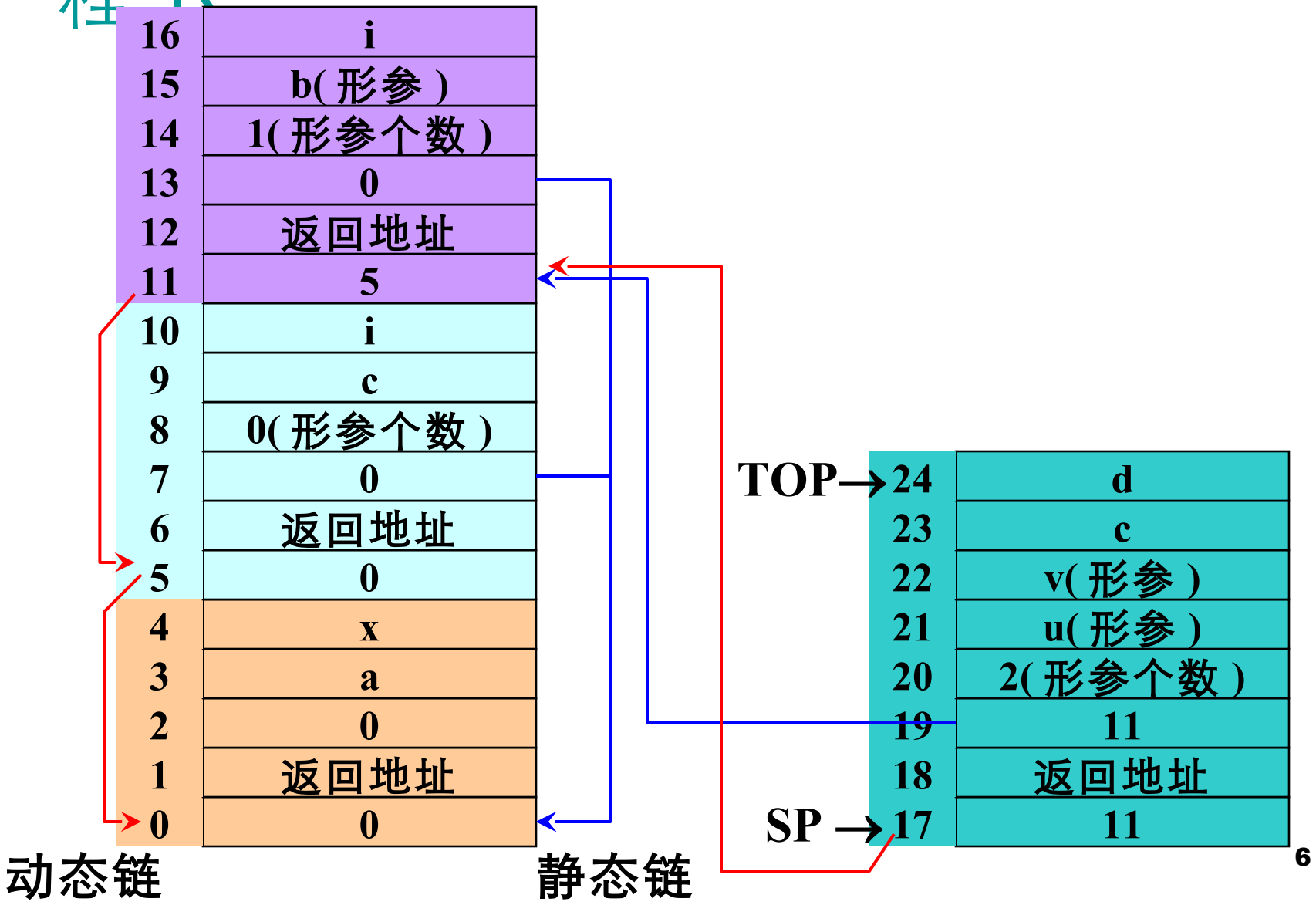
- PASCAL
- 非局部名字的访问的实现
  - 静态链和活动记录
  - 关键问题：如何根据调用过程的活动记录信息建立被调用过程的静态链?

嵌套过程语言的栈式实现

- 如何根据调用过程 P1 的活动记录信息建立被调用过程 P2 的静态链？
  - 第 N 层过程调用第 N+1 层过程： P2 的静态链为调用过程 P1(第 N 层过程) 的最新活动记录的起始地址
  - 第 N 层过程调用第 N 层过程： P2 的静态链为调用过程 P1(第 N 层过程) 的静态链的值
  - 第 N 层过程调用第 N-x 层过程： P2 的静态链为沿着调用过程 P1(第 N 层过程) 的静态链的向前走 x 步到达的活动记录的静态链的值



- 主程序 P → 过程 S → 过程 Q → 过程 R

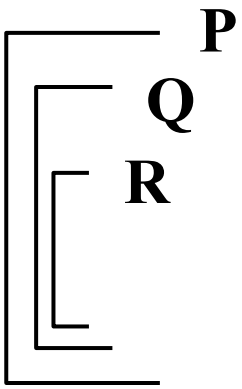


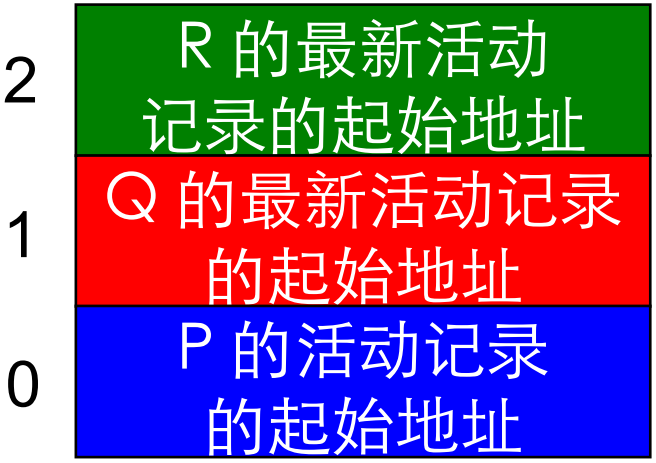
嵌套过程语言的栈式实现

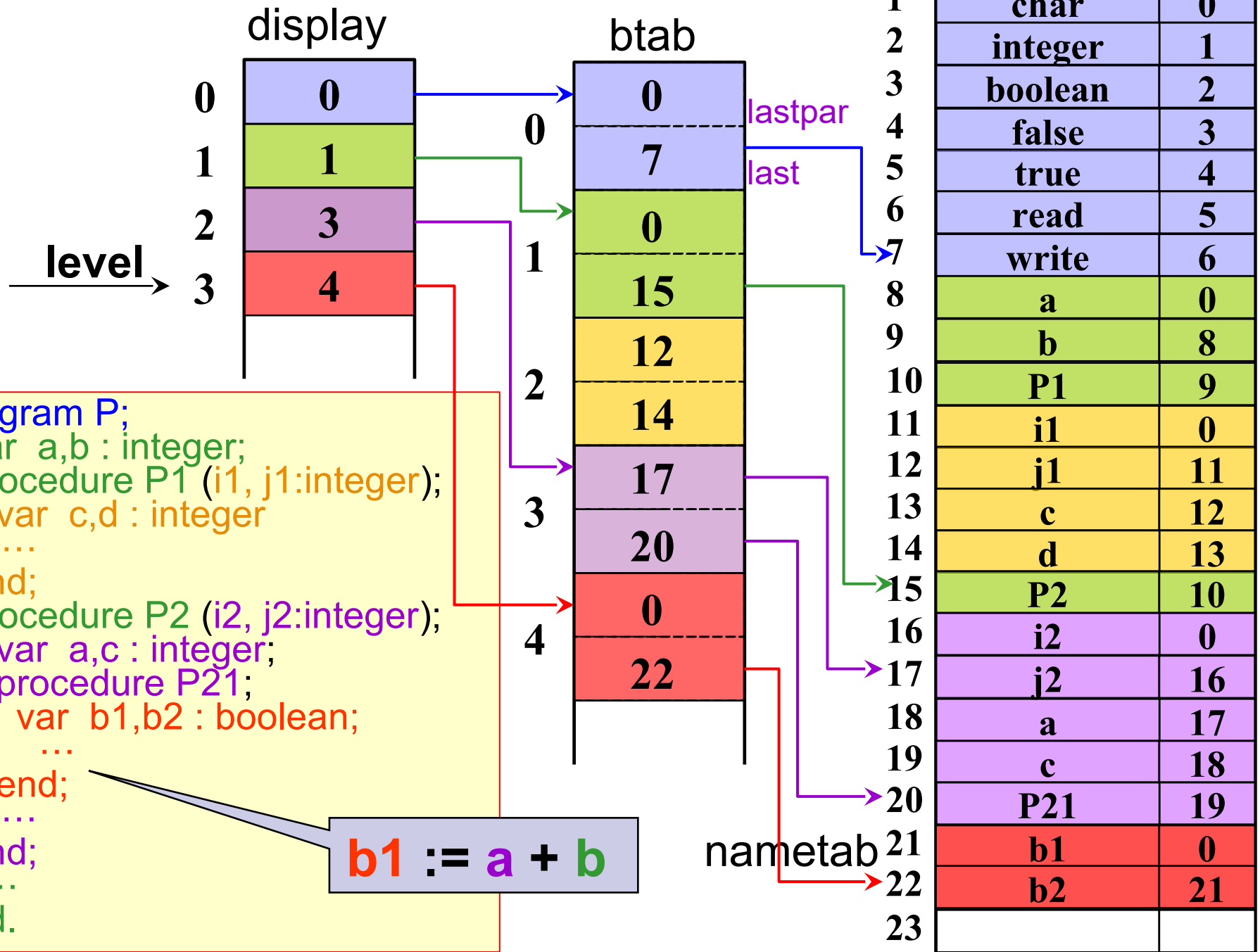
- PASCAL
- 非局部名字的访问的实现
  - 静态链和活动记录
  - 嵌套层次显示表 display
- 过程调用、过程进入、过程返回

嵌套层次显示表 display

- 当进入一个过程后，在建立其活动记录区的同时建立一张嵌套层次显示表 display，把 display 表作为活动记录的一部分
- 令过程 R 的外层为 Q， Q 的外层为主程序为 P，则过程 R 运行时的 DISPLAY 表内容为



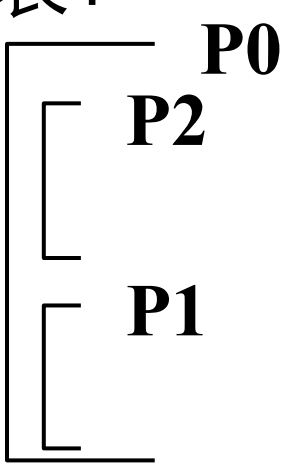


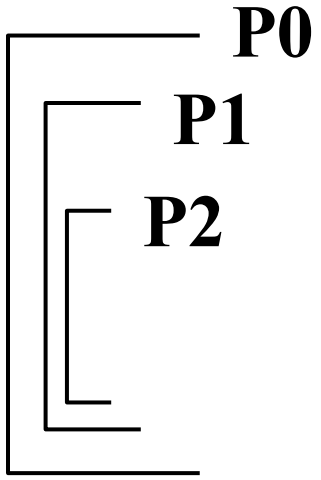

display
btab
level
0
1
2
3
0
1
2
3
4
lastpar
last
nametab
name
link
char 0
integer 1
boolean 2
false 3
true 4
read 5
write 6
a 0
b 8
P1 9
i1 0
j1 11
c 12
d 13
P2 10
i2 0
j2 16
a 17
c 18
P21 19
b1 0
b2 21
program P;
var a,b : integer;
procedure P1 (i1, j1:integer);
var c,d : integer
…
end;
procedure P2 (i2, j2:integer);
var a,c : integer;
procedure P21;
var b1,b2 : boolean;
…
end;
…
end;
…
end.
b1 := a + b

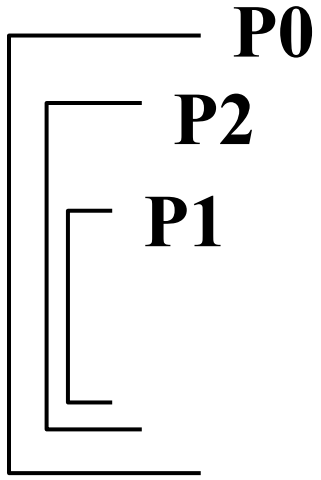
运行时 Display 表 vs. 编译时 Display 表

- 相联系
  - 处理非局部名字的访问
- 有区别
  - 编译时 Display 表：帮助编译确定变量的层数
  - 运行时 Display 表：帮助运行时在外层过程的活动记录中找到非局部名的存储单元

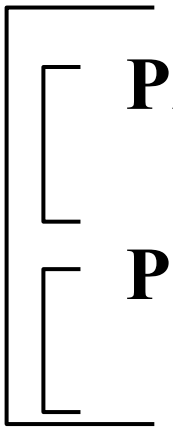
- 问题：当过程 P1 调用过程 P2(*l2* 层) 而进入 P2 后， P2 应如何建立起自己的 display 表?



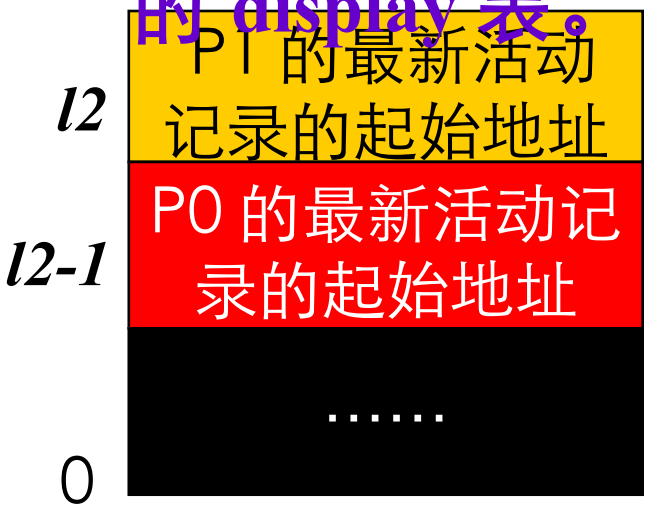




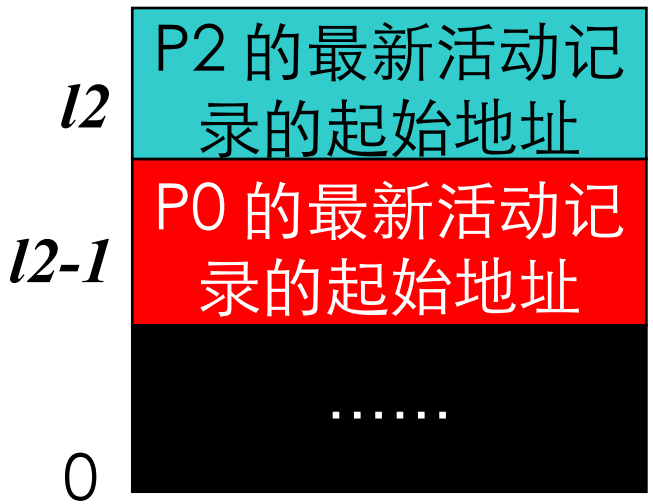
- 问题：当过程 P1 调用过程 P2 (*l2* 层) 而进入 P2 后， P2 应如何建立起自己的 display 表?



从 P1 的 display 表中自底而上地取过 *l2* 个单元（ *l2* 为 P2 的层数）再添上进入 P2 后新建立的 SP 值就构成了 P2 的 display 表。



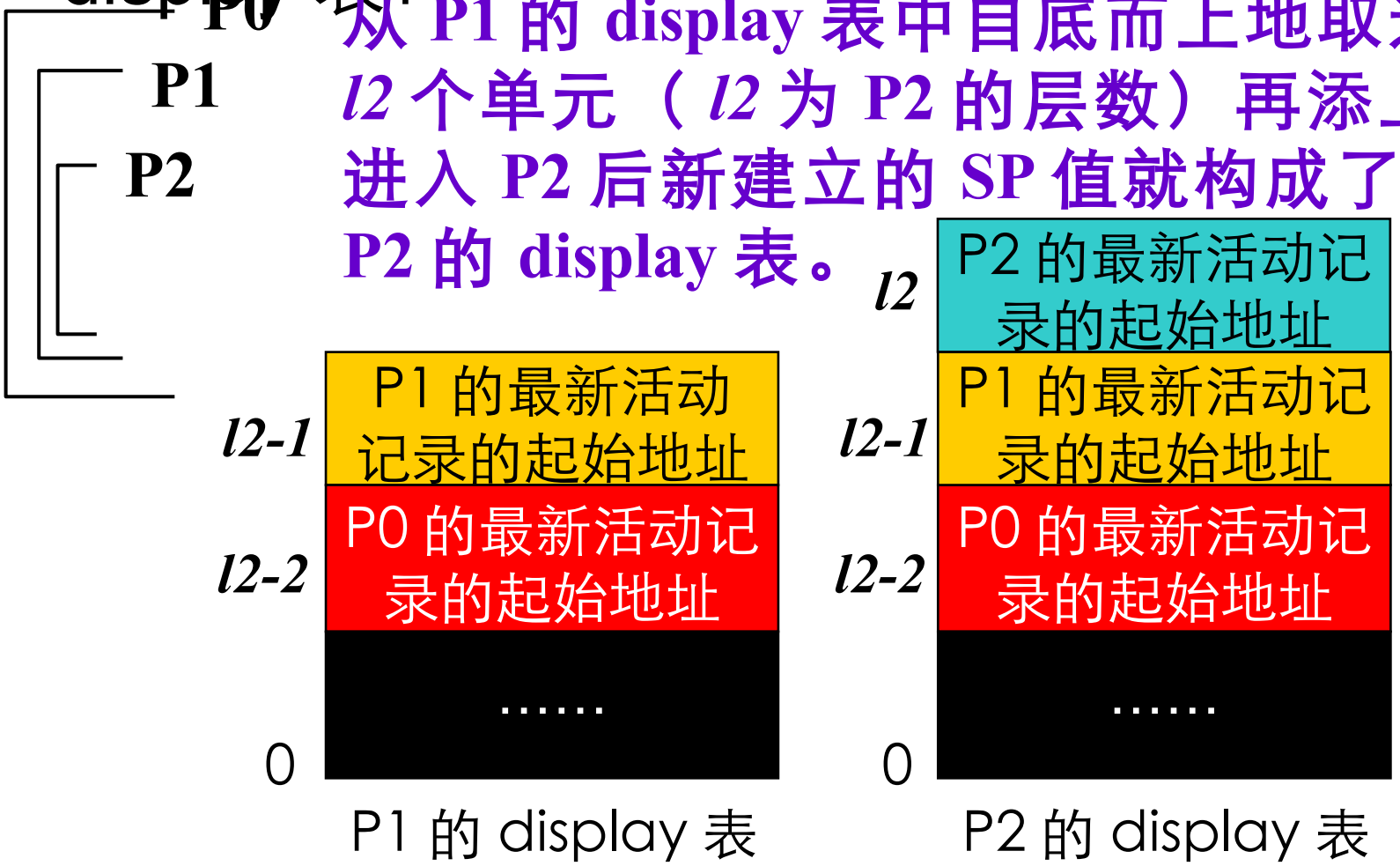
P1 的 display 表



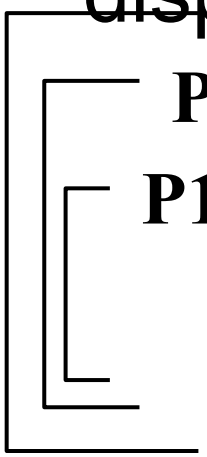
P2 的 display 表

- 问题：当过程 P1 调用过程 P2 (*l2* 层) 而进入 P2 后， P2 应如何建立起自己的 display 表?

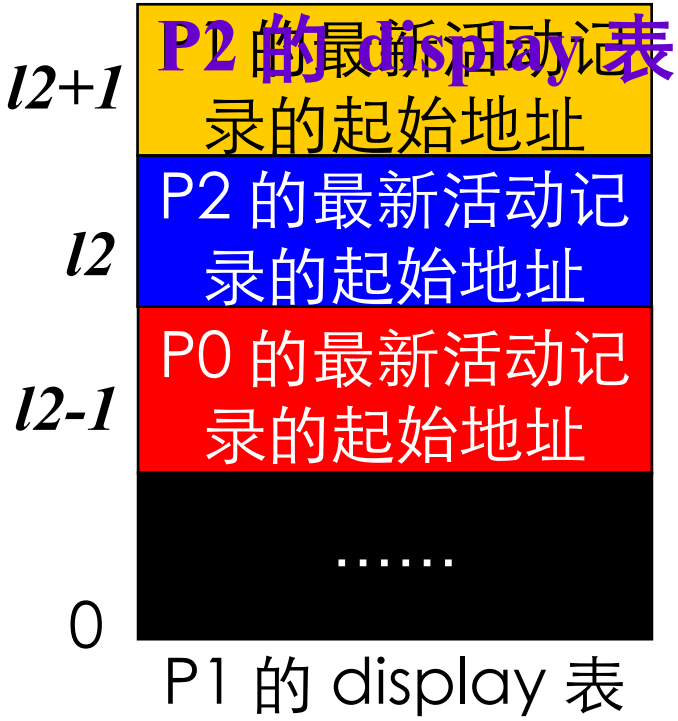
从 P1 的 display 表中自底而上地取过 *l2* 个单元（ *l2* 为 P2 的层数）再添上进入 P2 后新建立的 SP 值就构成了 P2 的 display 表。

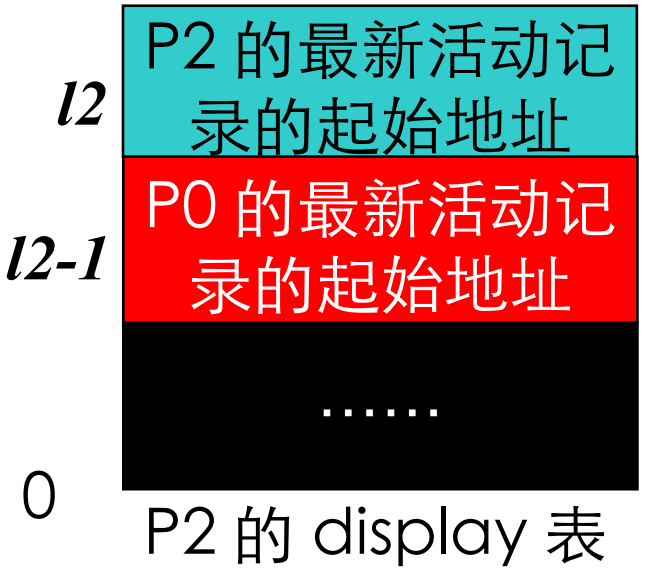


- 问题：当过程 P1 调用过程 P2 (*l2* 层) 而进入 P2 后， P2 应如何建立起自己的 display 表?

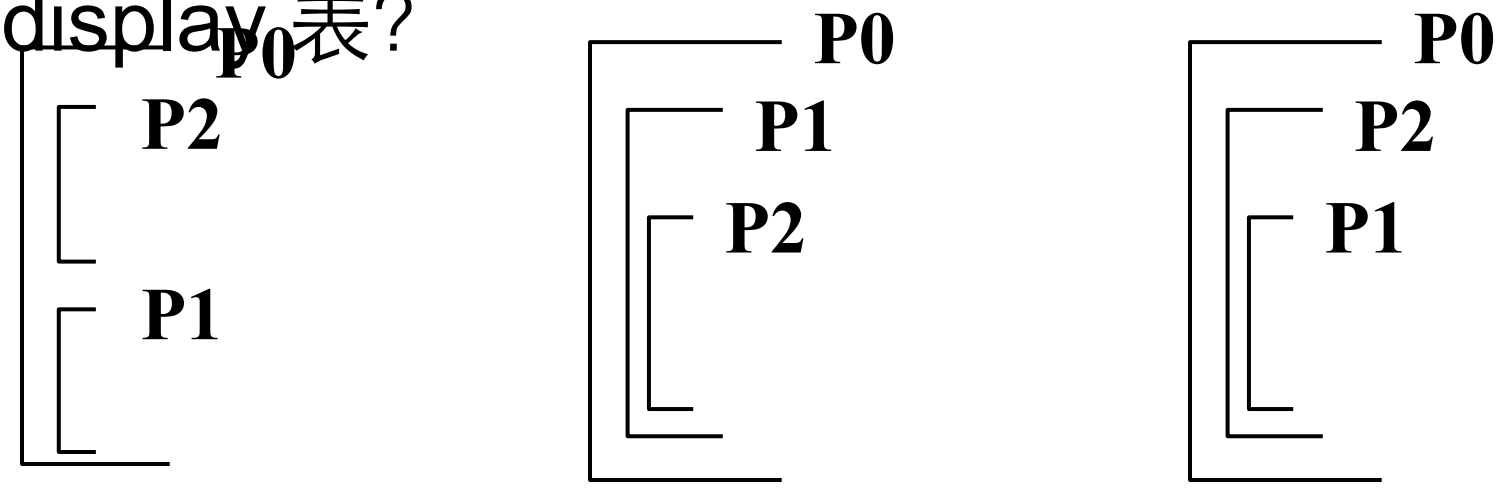


从 P1 的 display 表中自底而上地取过 *l2* 个单元（ *l2* 为 P2 的层数）再添上进入 P2 后新建立的 SP 值就构成了 P2 的 display 表。





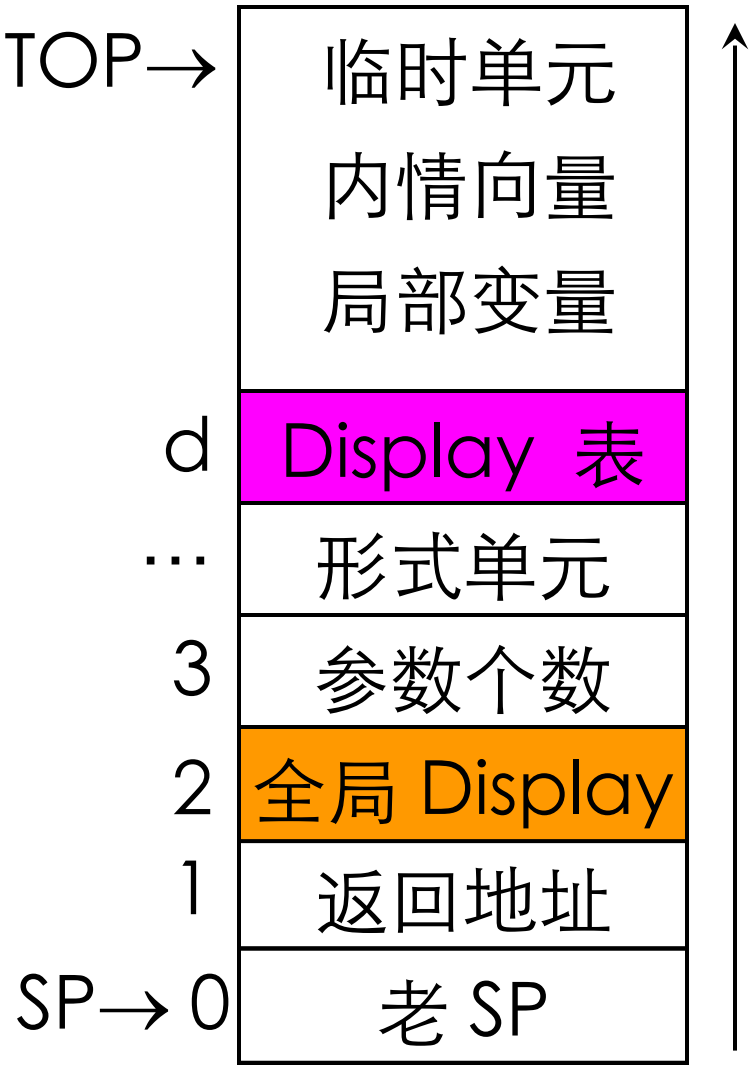
- 问题：当过程 P1 调用过程 P2 (*l2* 层) 而进入 P2 后， P2 应如何建立起自己的 display 表?



答案：从 P1 的 display 表中自底而上地取过 *l2* 个单元（ *l2* 为 P2 的层数）再添上进入 P2 后新建立的 SP 值就构成了 P2 的 display 表。

- 把 P1 的 display 表地址 (全局 Display) 作为连接数据之一传送给 P2 ， 然后建立 P2 的 display 表

嵌套过程语言活动记录



- 全局 Display：调用过程的 Display 表地址
- Display 表在活动记录中的相对地址 d 在编译时能完全确定
- 假定在现行过程中引用了某层过程 (令其层次为 k) 的 X 变量，那么，可用下面两条指令获得 X 的地址：

  LD R_1 (d+k)[SP]

  LD R_2 dx[R_1]

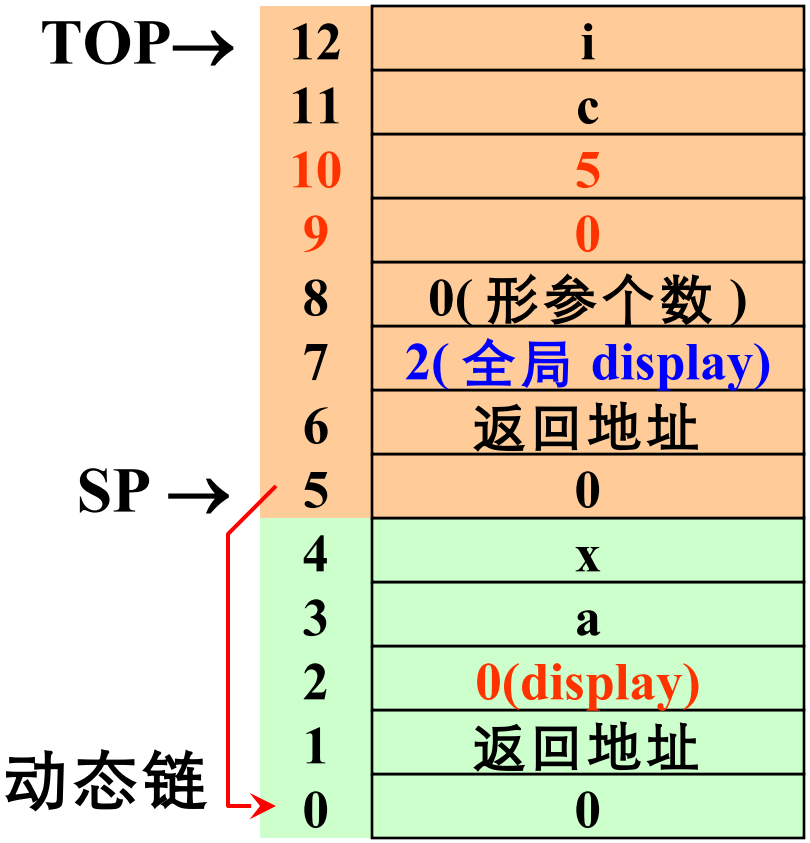
图 9.15 程序

```
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→
过程 S

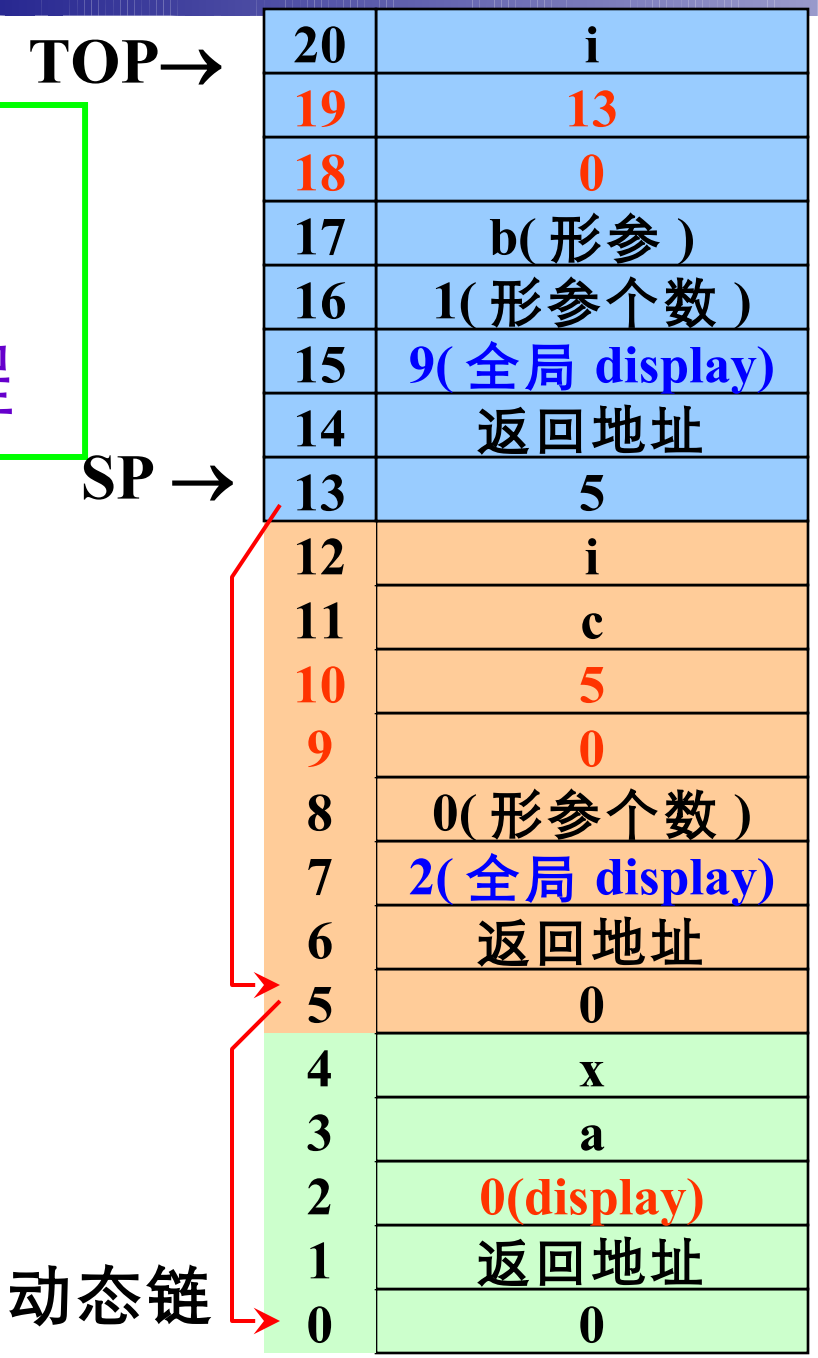

TOP→
SP →
动态链
12
11
10
9
8
7
6
5
4
3
2
1
0
i
c
5
0
0(形参个数)
2(全局 display)
返回地址
0
x
a
0(display)
返回地址
0

主程序

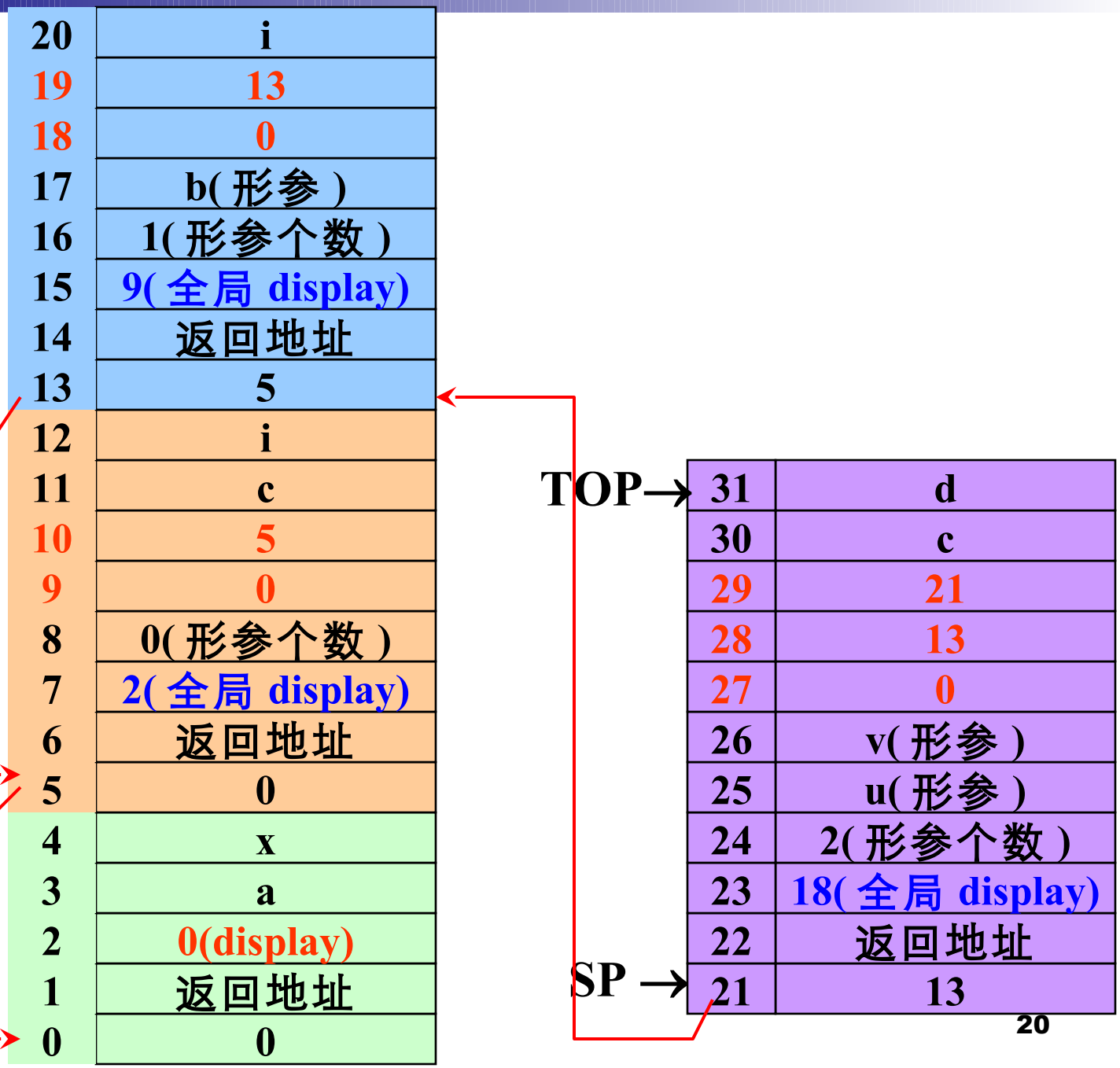
P→ 过程

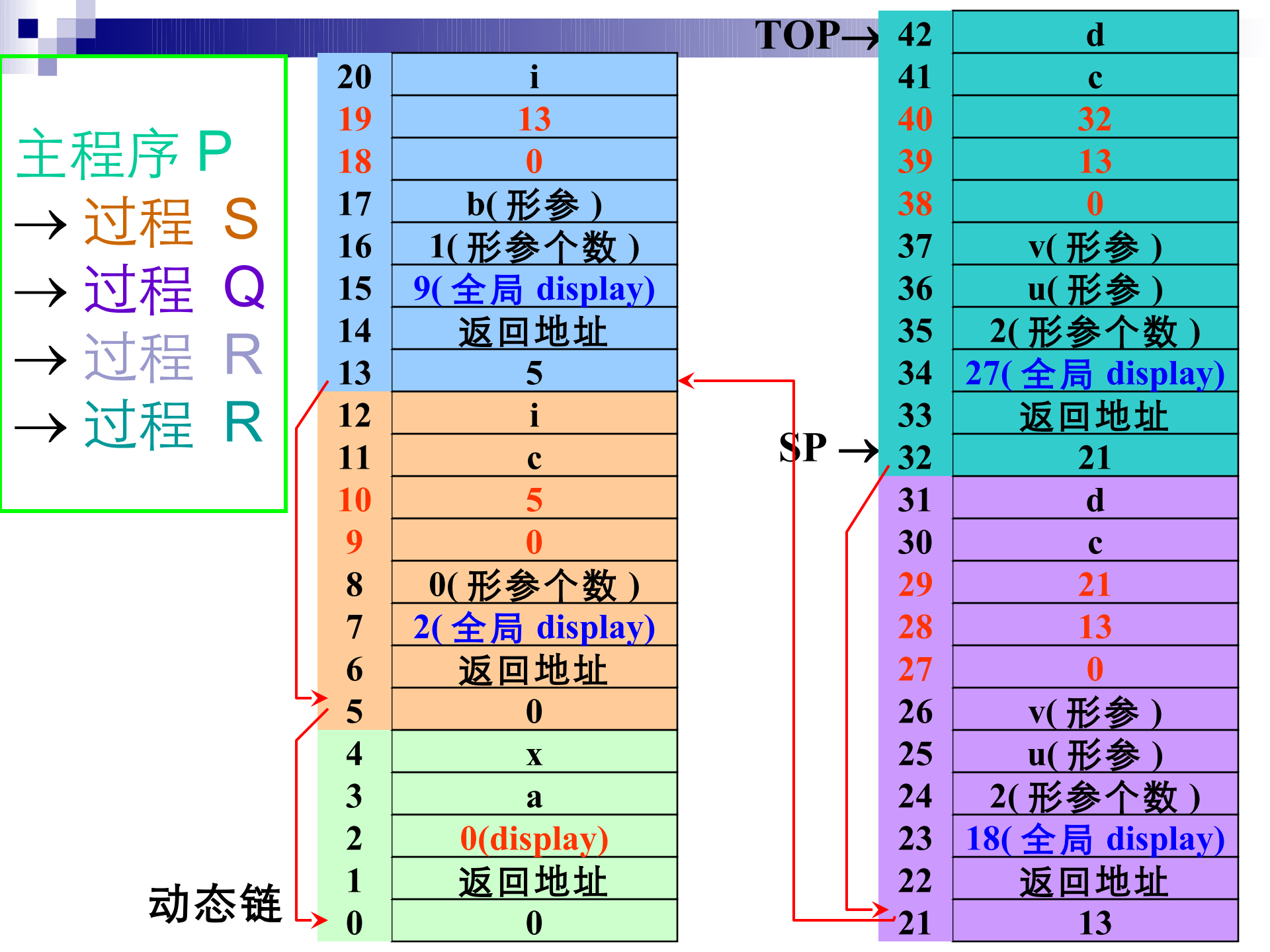
S → 过程

Q


TOP→
SP →
20 i
19 13
18 0
17 b(形参)
16 1(形参个数)
15 9(全局 display)
14 返回地址
13 5
12 i
11 c
10 5
9 0
8 0(形参个数)
7 2(全局 display)
6 返回地址
5 0
4 x
3 a
2 0(display)
1 返回地址
0 0
动态链

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
TOP→
SP →
动态链
20 i
19 13
18 0
17 b(形参)
16 1(形参个数)
15 9(全局 display)
14 返回地址
13 5
12 i
11 c
10 5
9 0
8 0(形参个数)
7 2(全局 display)
6 返回地址
5 0
4 x
3 a
2 0(display)
1 返回地址
0 0
42 d
41 c
40 32
39 13
38 0
37 v(形参)
36 u(形参)
35 2(形参个数)
34 27(全局 display)
33 返回地址
32 21
31 d
30 c
29 21
28 13
27 0
26 v(形参)
25 u(形参)
24 2(形参个数)
23 18(全局 display)
22 返回地址
21 13

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

过程调用、过程进入、过程返回

1. 每个 par T_i(i=1,2,…n) 可直接翻译成如下指令：

   (i+4)[TOP]:= T_i　　（传值）

   (i+4)[TOP]:=addr(T_i)　　（传地址）

过程调用、过程进入、过程返回

2. call P ， n 被翻译成：

1[TOP]:=SP （保护现行 SP)

3[TOP]:=SP+d （传送现行 display 地址）

4[TOP]:=n （传递参数个数）

JSR （转子指令）

3. 转进过程 P 后，首先定义新的 SP 和 TOP ，保存返回地址。

4. 根据 " 全局 display" 建立现行过程的 display ： 从全局 display 表中自底向上地取 *i* 个单元，在添上进入 P 后新建立的 SP 值就构成了 P 的 display 。

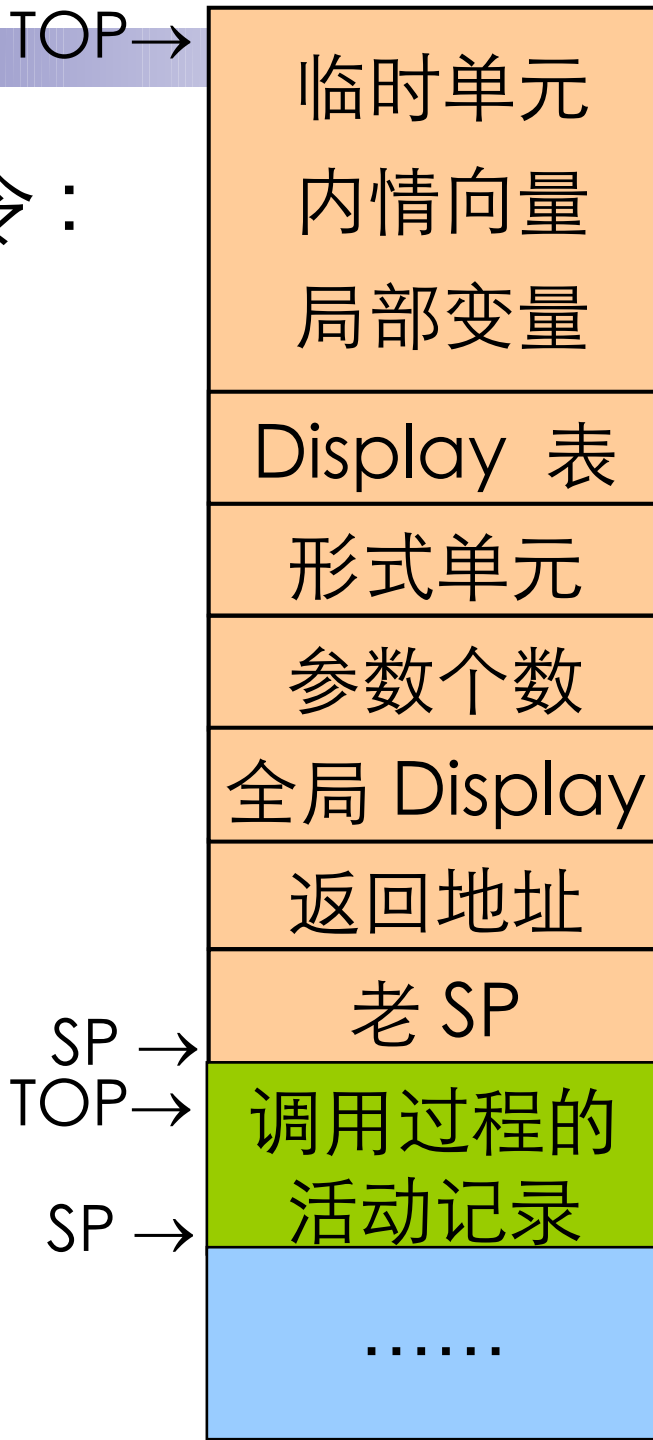
5. 过程返回时，执行下述指令：

TOP:=SP-1

SP:=0[SP]

X:=2[TOP]

UJ 0[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

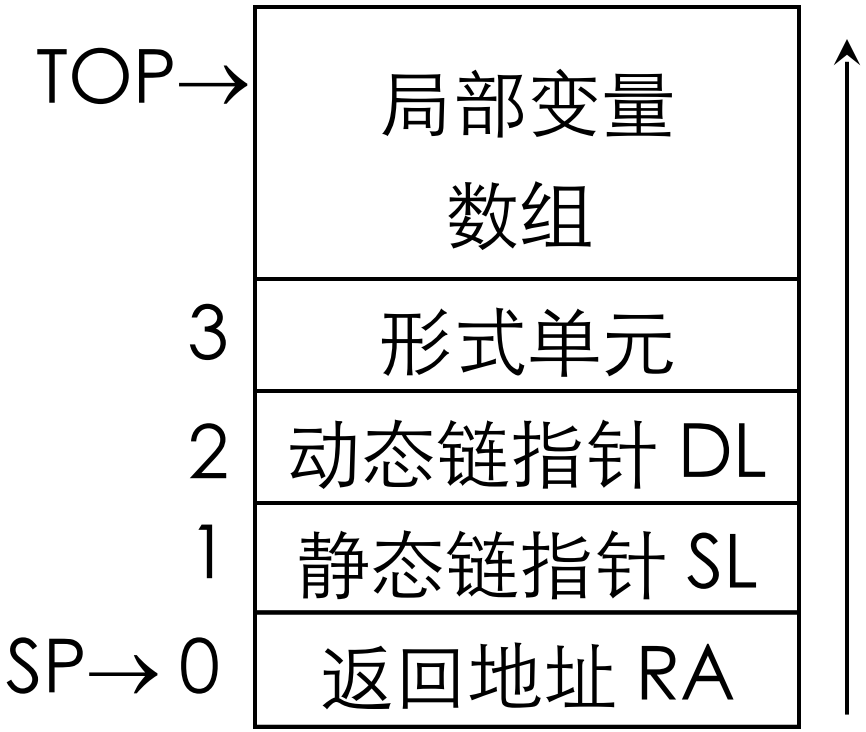
上面两种方法的 " 折中 "——PL 编译程序

上面两种方法的 " 折中 "——PL 编译程序

- 建立一个总的运行时的 display 表 (而不是每个过程的活动记录中都有一个)，它记录正被调用执行的过程及其所有外层过程的活动记录的起始地址。通过这个 display 访问数据。
- 在各过程活动记录之间保留一条 " 静态链 SL"，它主要用于由层次大的过程调用层次小的过程后返回时，恢复调用前的 display 内容。
  - 调用返回后，执行一条 UDIS 指令

- PL 中每个过程的活动记录结构：

1) 简单局部变量
2) 连接数据
   - 返回地址 RA
   - 动态链 DL，指向过程调用者的活动记录起始地址
   - 静态链 SL，指向过程的直接外层过程活动记录起始地址



```
program P;
   var x,y: integer;
   ...
   procedure P1;
      var i,j:integer;
      ...
      procedure P11(a,b:integer);
      ...
      begin
      ...
      end;
    begin
      ...
      call P11(i,j);
      ...
    end;
```

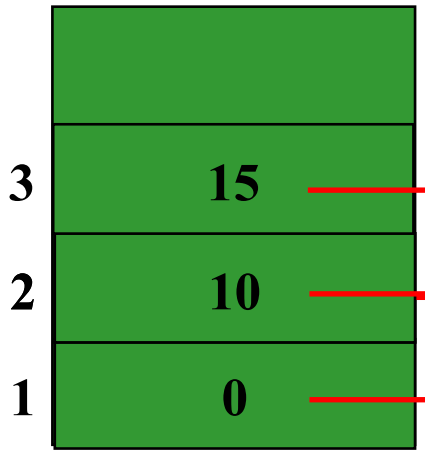
```
procedure P2;
  var s,t:integer;
    ...
    procedure P21;
      begin
      ...
      end;
    begin
      ...
      call P1
      ...
    end;
begin
    ...
    call P2;
    ...
end.
```

主程序 P

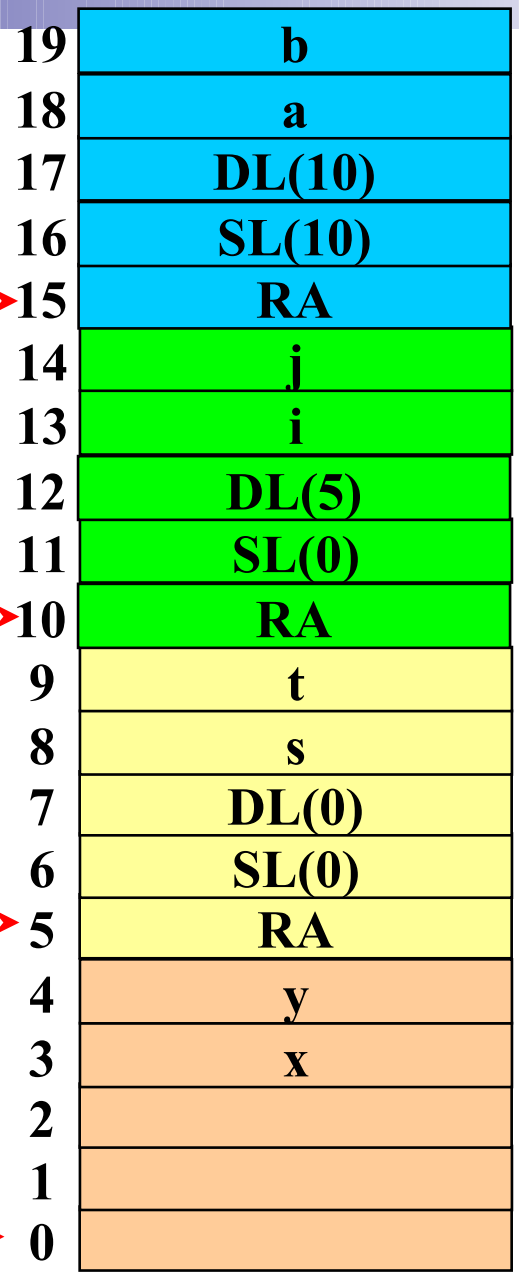
→ 过程 P2

→ 过程 P1

→ 过程 P11



Display

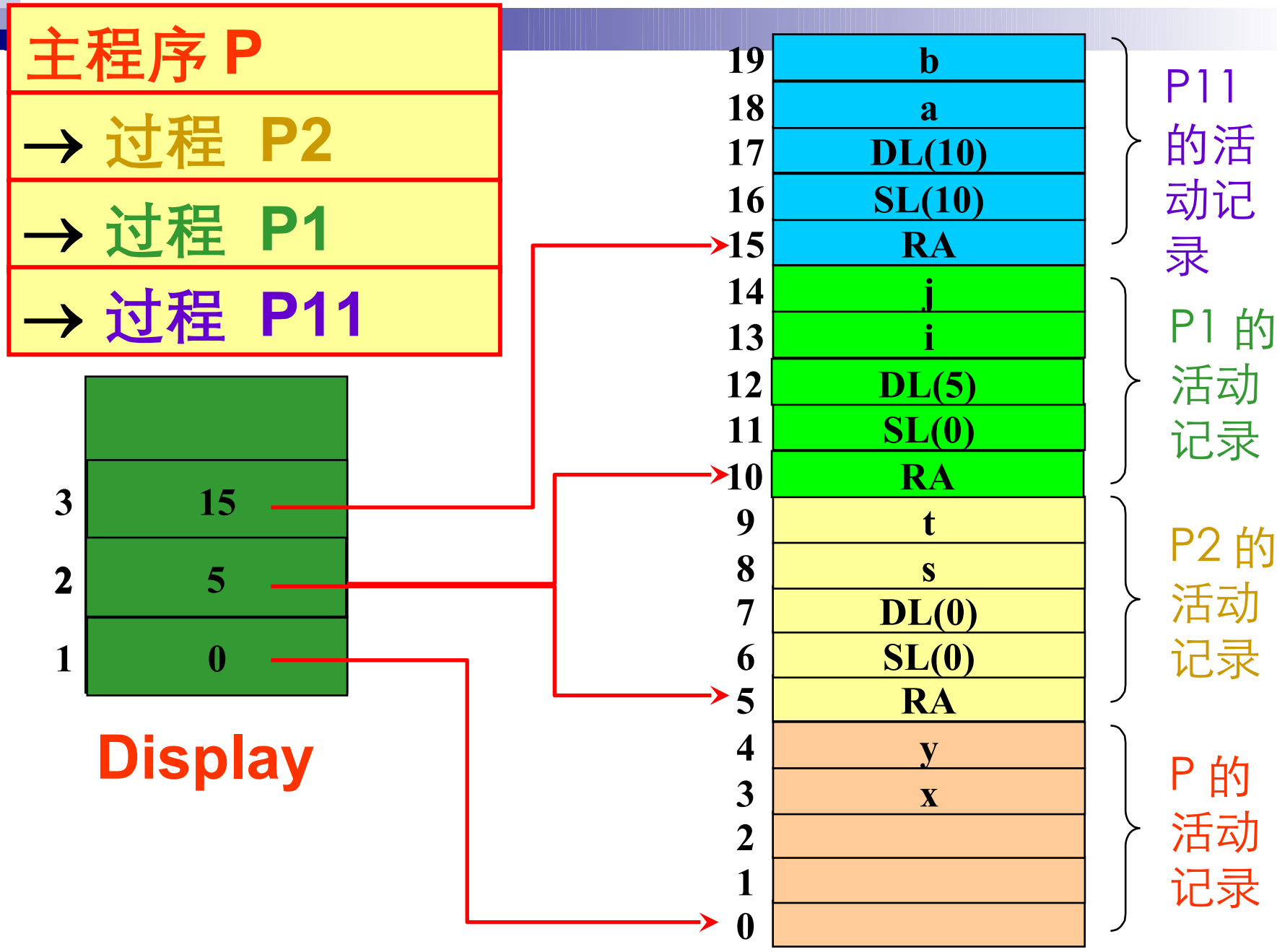


P11 的活动记录

P1 的活动记录

P2 的活动记录

P 的活动记录


主程序 P
→ 过程 P2
→ 过程 P1
→ 过程 P11
3
15
2
5
1
0
Display
19
b
18
a
17
DL(10)
16
SL(10)
15
RA
14
j
13
i
12
DL(5)
11
SL(0)
10
RA
9
t
8
s
7
DL(0)
6
SL(0)
5
RA
4
y
3
x
2
1
0
P11 的活动记录
P1 的活动记录
P2 的活动记录
P 的活动记录

```
program P;
   var x,y: integer;
     ...
   procedure P1;
      var i,j:integer;
        ...
      procedure P11(a,b:integer);
          ...
      begin
          ...
      end;
  begin
        ...
       call P11(i,j);
        ...
  end;
procedure P2;
     var s,t:integer;
       ...
```

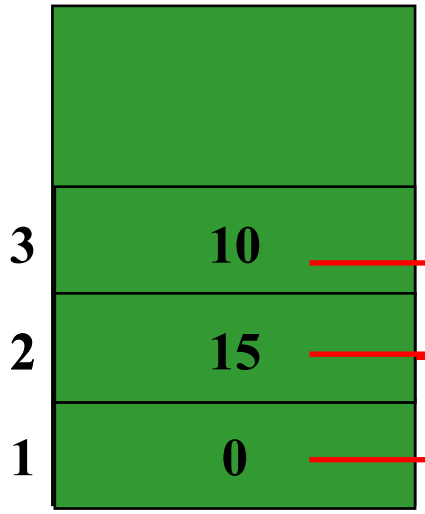
```
procedure P21;
     var u,v:integer;
       ...
      begin
           ...
                call P1
           ...
      end;
   begin
       ...
      call P21
        ...
   end;
begin
      ...
     call P2;
      ...
end.
```

主程序 P

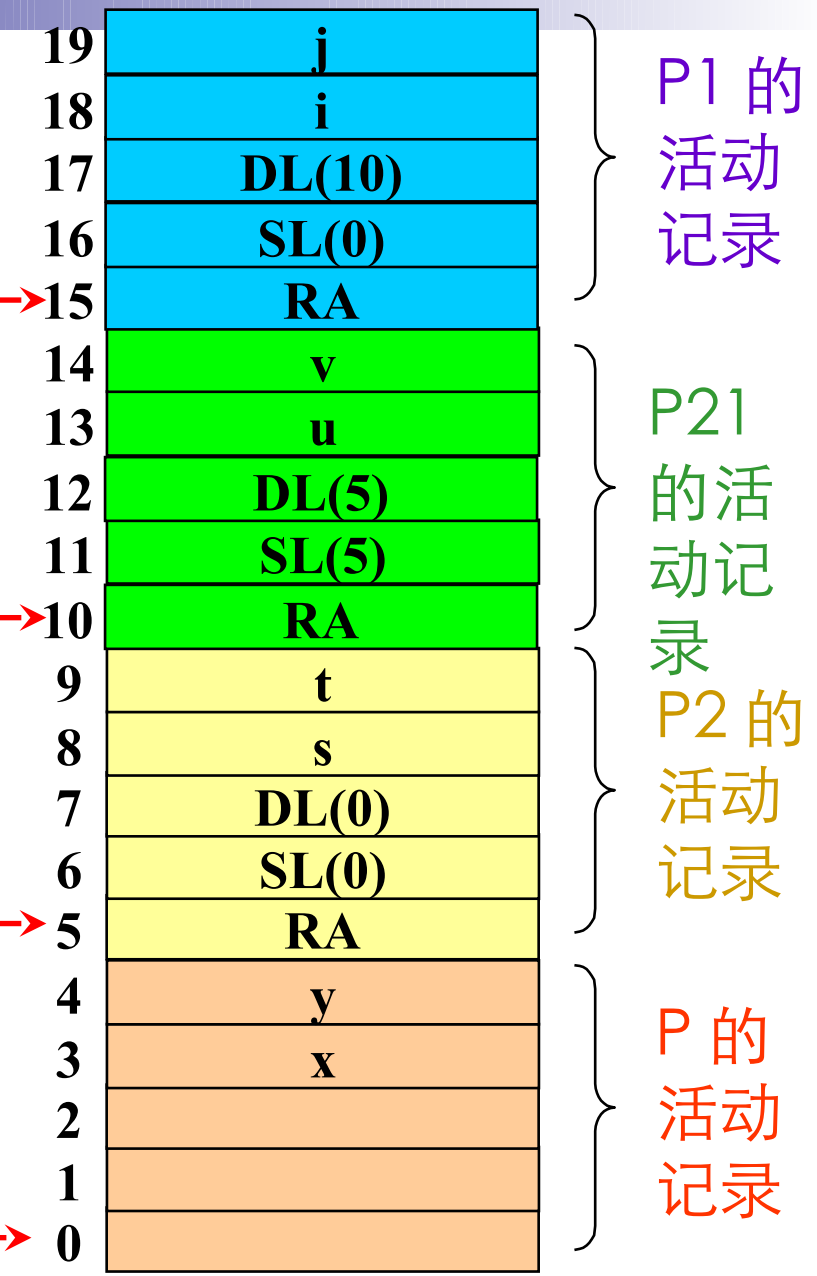
→ 过程 P2

→ 过程 P21

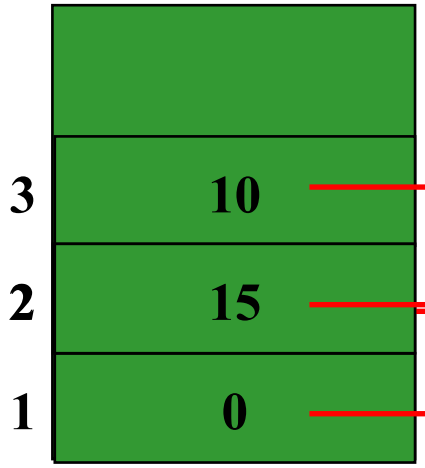
→ 过程 P1



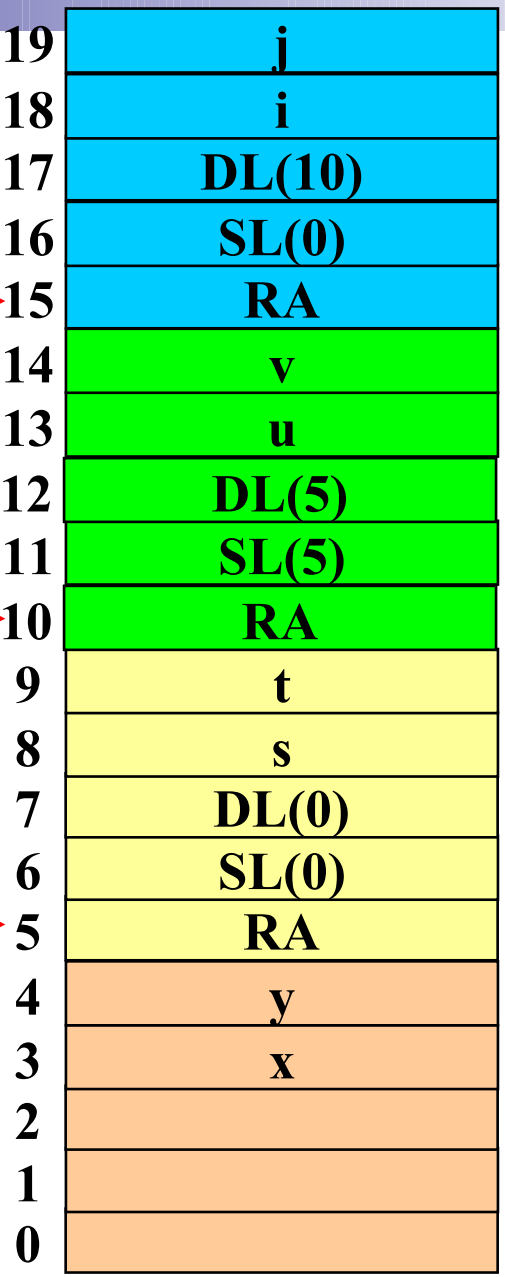
Display



```
udis:begin
        h1:=a;h2:=l;h3:=base;
        repeat
            display[h1]:=h3;
            h1:=h1-1;
            h3:=s[h3+1]
        until h1<h2
    end;
```



Display



P1 的活动记录

P21的活动记录

P2 的活动记录

P 的活动记录

小结

- 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

本章小结

- 目标程序运行时的活动
- 参数传递
- 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