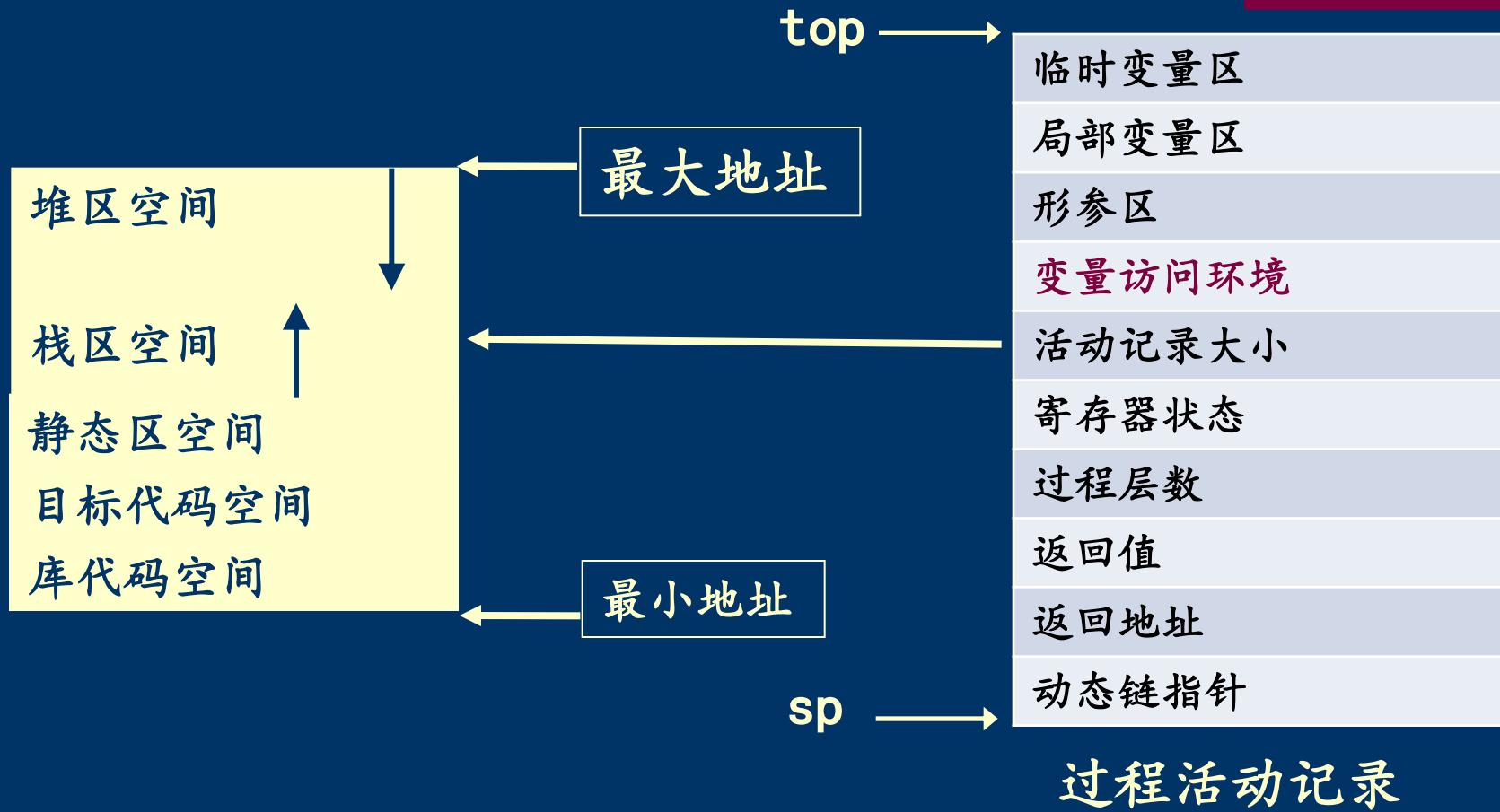


第七章：运行时存储空间管理（2）

变量访问环境

变量访问环境



调用链、动态链

❖ 调用链：过程名序列

若M是主程序名，则(M)是一个调用链；

若(M, …, R)是调用链，且在R中有S的调用，则(M, …, R, S)也是调用链。

记为： $\text{CallChain}(S) = (M, \dots, R, S)$

❖ 动态链：

如果有调用链 $\text{CallChain}(S) = (M, \dots, R, S)$ ，则它对应的动态链为：

$\text{DynamicChain} = [\text{AR}(M), \dots, \text{AR}(R), \text{AR}(S)]$

活跃活动记录 (LAR)

✿ LiveAR (LAR) :

一个过程S在动态链中可有多个AR，但其中只有最新AR(S)是可访问的，称此AR(S)为S的活跃活动记录，并记为LiveAR(S)，简写为LAR(S)。

活跃活动记录 (LAR)

例：假设有当前调用链是 $(M, P^1, P^2, Q^1, R^1, R^2, R^3)$

则当前动态AR链为

$[AR(M), AR(P^1), AR(P^2), AR(Q), AR(R^1), AR(R^2), AR(R^3)]$

活跃活动记录LAR为：

$$LAR(M) = AR(M)$$

$$LAR(P) = AR(P^2)$$

$$LAR(Q) = AR(Q^1)$$

$$LAR(R) = AR(R^3)$$

声明链和变量访问环境

❖ 过程声明链 (DeclaChain) : 过程名序列 (M) 是过程声明链, M 是主程序名; 若 (M, \dots, P) 是过程声明链, 且 P 中有过程 Q 的声明, 则 (M, \dots, P, Q) 也是过程声明链;

记为: $\text{DeclaChain}(Q) = (M, \dots, P, Q)$

❖ 当前变量访问环境VarVisitEnv:

若 $\text{DeclaChain}(Q) = [M, \dots, P, Q]$ 则

$\text{VarVisitEnv}(\text{LAR}(Q)) = [\text{LAR}(M), \dots, \text{LAR}(P), \text{LAR}(Q)]$

例子

例： (M, P, Q, R) 为 R 的声明链， 假设有当前调用链是
 $(M, P^1, P^2, Q^1, R^1, R^2, R^3)$

则当前动态链为：

$[AR(M), AR(P^1), AR(P^2), AR(Q^1), AR(R^1), AR(R^2), AR(R^3)]$

R 的当前变量访问环境：

$\text{VarVisitEnv}(\text{LAR}(R)) = [AR(M), AR(P^2), AR(Q^1), AR(R^3)]$

非局部变量访问的实现：

假设Q的变量访问环境：

VarVisitEnv (LAR (Q))

= [LAR (M) , …, LAR (P) , LAR (Q)] , 在Q中有变量
 X_Q , Y_M , Z_P , 它们分别定义在过程Q、M和P中，则它们的存
储单元地址可表示如下（其中<LAR (Q)>表示LAR (Q) 的
始地址，其它类似）：

$$\text{addr} (X_Q) = \langle \text{LAR} (Q) \rangle + \text{Offset}_X$$

$$\text{addr} (Y_M) = \langle \text{LAR} (M) \rangle + \text{Offset}_Y$$

$$\text{addr} (Z_P) = \langle \text{LAR} (P) \rangle + \text{Offset}_Z$$

结论：对于每个AR, 只要知道了它的变量访问环境
VarVisitEnv (AR) , 即可实现包括非局部变量在内的所有变
量的访问。

如何计算当前过程的变量访问环境

情况1
PROC P;
... ...
Begin
P
End

PROC Q;

Begin

...

end

... ...

PROC P;

Begin

Q

End

情况3

PROC P;

... ...

PROC Q

Begin End

... ...

Begin Q End

情况4

PROC Q;

... ...

PROC P;

Begin Q

End

Begin End

情况1: 情况2: P调用Q, P层数等于Q层数 (N).

De

De

情况4: P调用Q, P层数大于Q层数 (N).

DeclaChain(Q) = (M, P₁, P₂, ..., P_{N-1}, Q)

DeclaChain(P) = (M, P₁, P₂, ..., P_{N-1}, Q, ..., P)

如何计算当前过程的变量访问环境

❖ 定理：设

$[AR(M), \dots, AR(P), AR(Q)] \in DynamicChain(Q)$, 且Q 的层数为N, 则有：

$VarVisitEnv(AR(Q))$

$=VarVisitEnv(AR(P))_N \oplus AR(Q)$

结论：变量访问环境可由先行过程的变量访问环境求得.

变量访问环境的实现方法

Display表方法

全局表法

局部表法

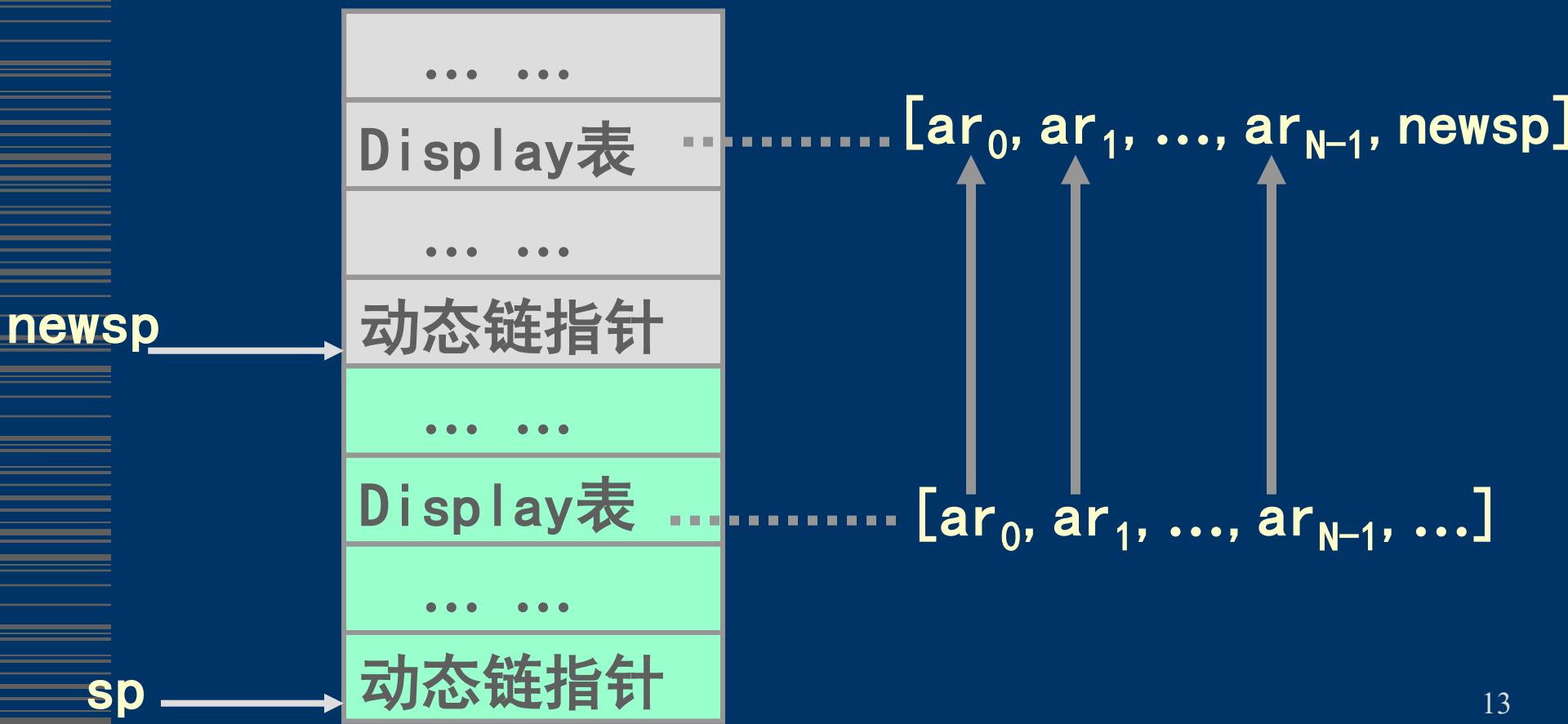
静态链方法

局部Display表方法

- 对于每个AR求出其变量访问环境，并把它以地址表的形式(Display表)保存在AR中。因为每个AR都自带Display表，称这种方法为局部化Display表方法。
- 如果层数为N的过程P的变量访问环境为：
 $\text{VarVisitEnv}(\text{AR}(P)) = [\text{AR}_0, \dots, \text{AR}_n]$,
 ar_i 表示 AR_i 的始地址，则
[$\text{ar}_0, \dots, \text{ar}_n$]是 $\text{AR}(P)$ 的Display表.

Display表的求法

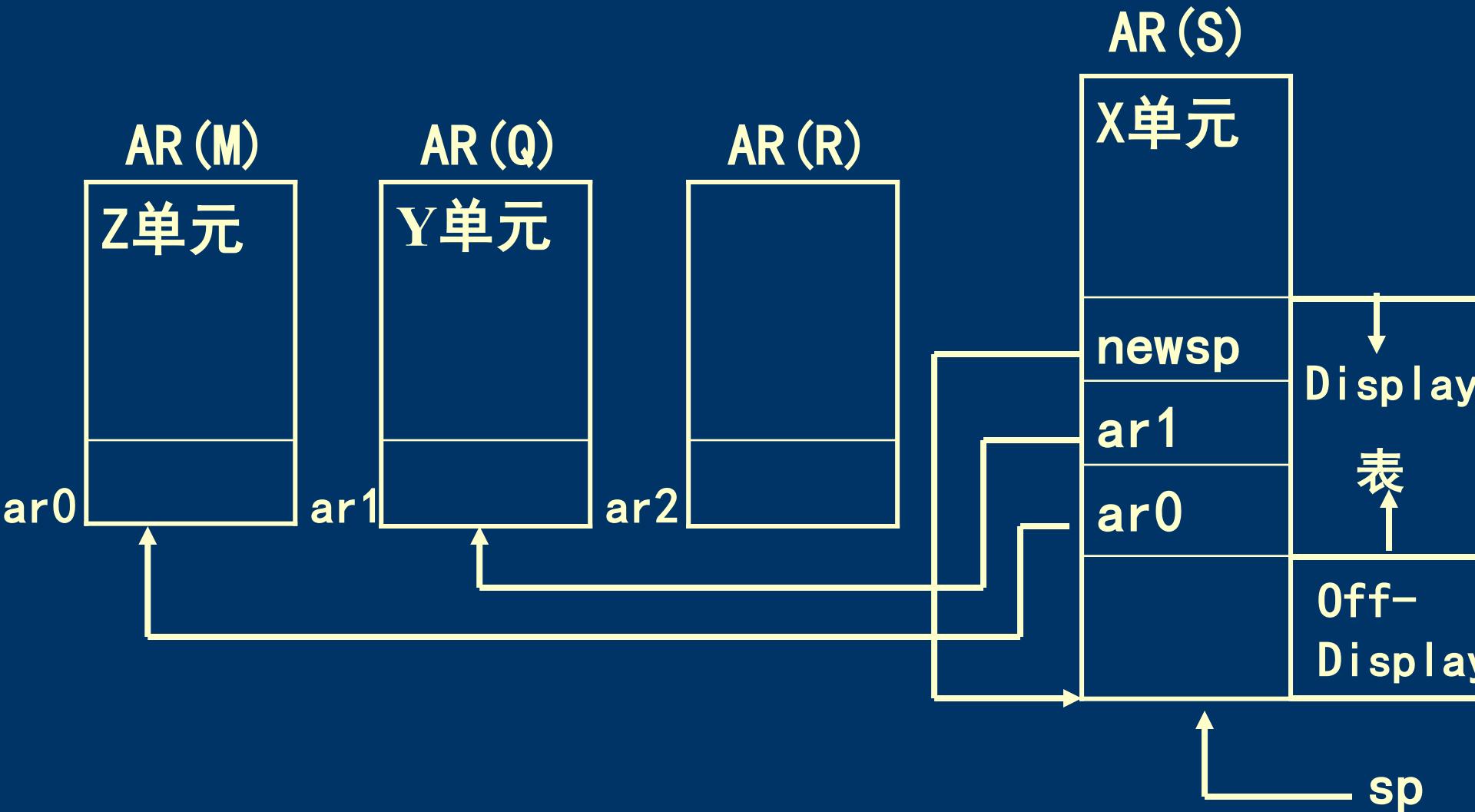
- ◆ $\text{NewAR}.\text{Display} = \text{CurrentAR}.\text{Display}$ 的前N项 $\oplus \text{newsp}$



例：有过程M, Q, R, S， 其中

level (M)=0; level (Q)=1; level (R)=1; level (S)=2,

各AR的Display表分别如下：



局部Display表时变量的访问

- ❖ 对一个变量 $X(L+1, off)$ ， 地址为：
当 $L = \text{CurrentAR.level}$ 时：
 $\text{addr}(X) = sp + off$
否则：
 $\text{addr}(X) = \text{CurrentAR.Display}[L] + off$
即 $[sp + D + L] + off$

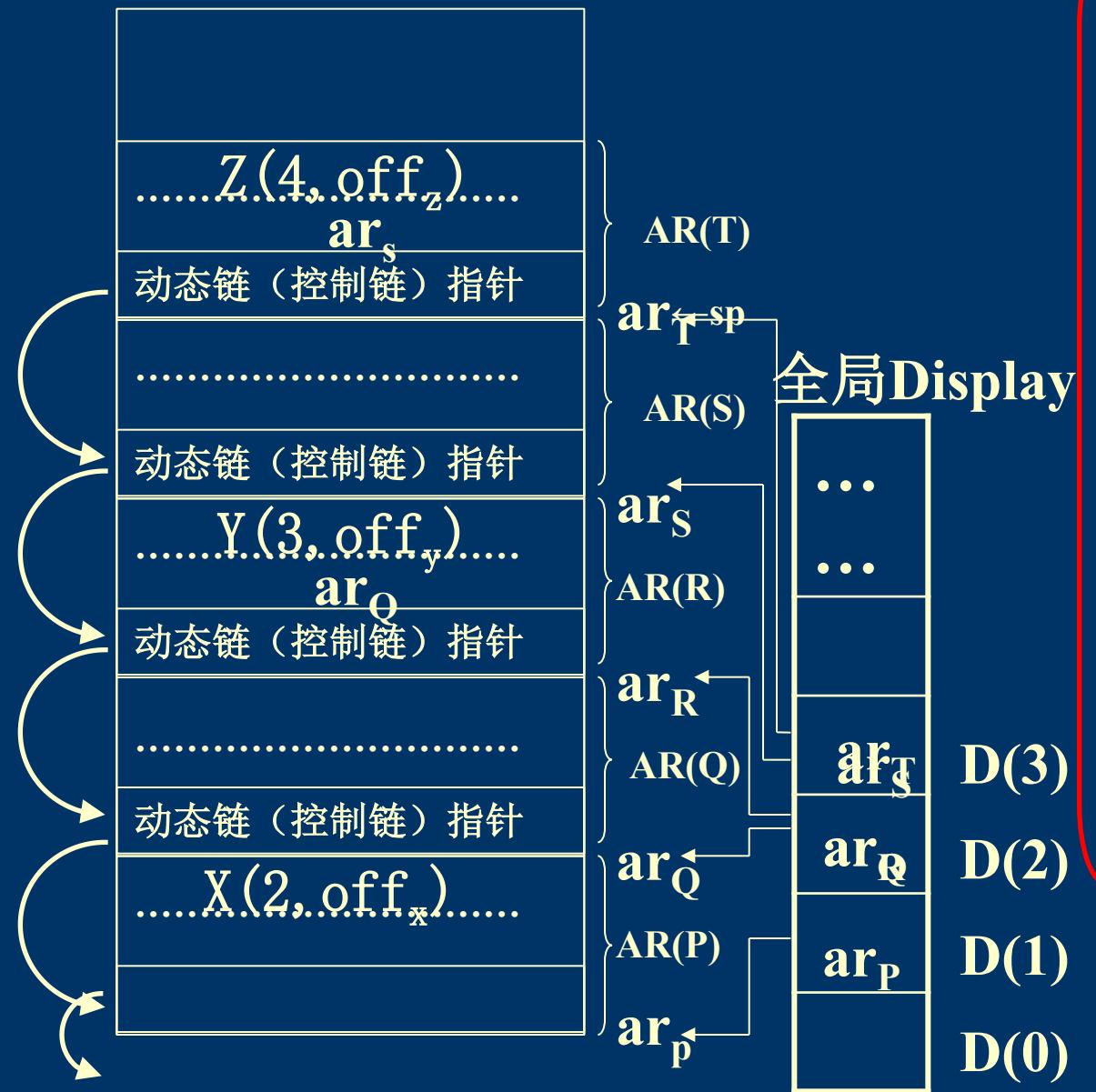
全局Display表

- ❖ 每个程序设置一个总的Display表，其长度为最大嵌套层数（最长声明链的长度），其中Display[i]存放第i层最新AR的指针，用D[i]表示。
- ❖ 该方法的理论依据：在程序的任何一点，相同层数的过程声明只能有一个有效。
- ❖ 在AR中设置一个Resume单元，用来临时保存某D[i]

全局Display表

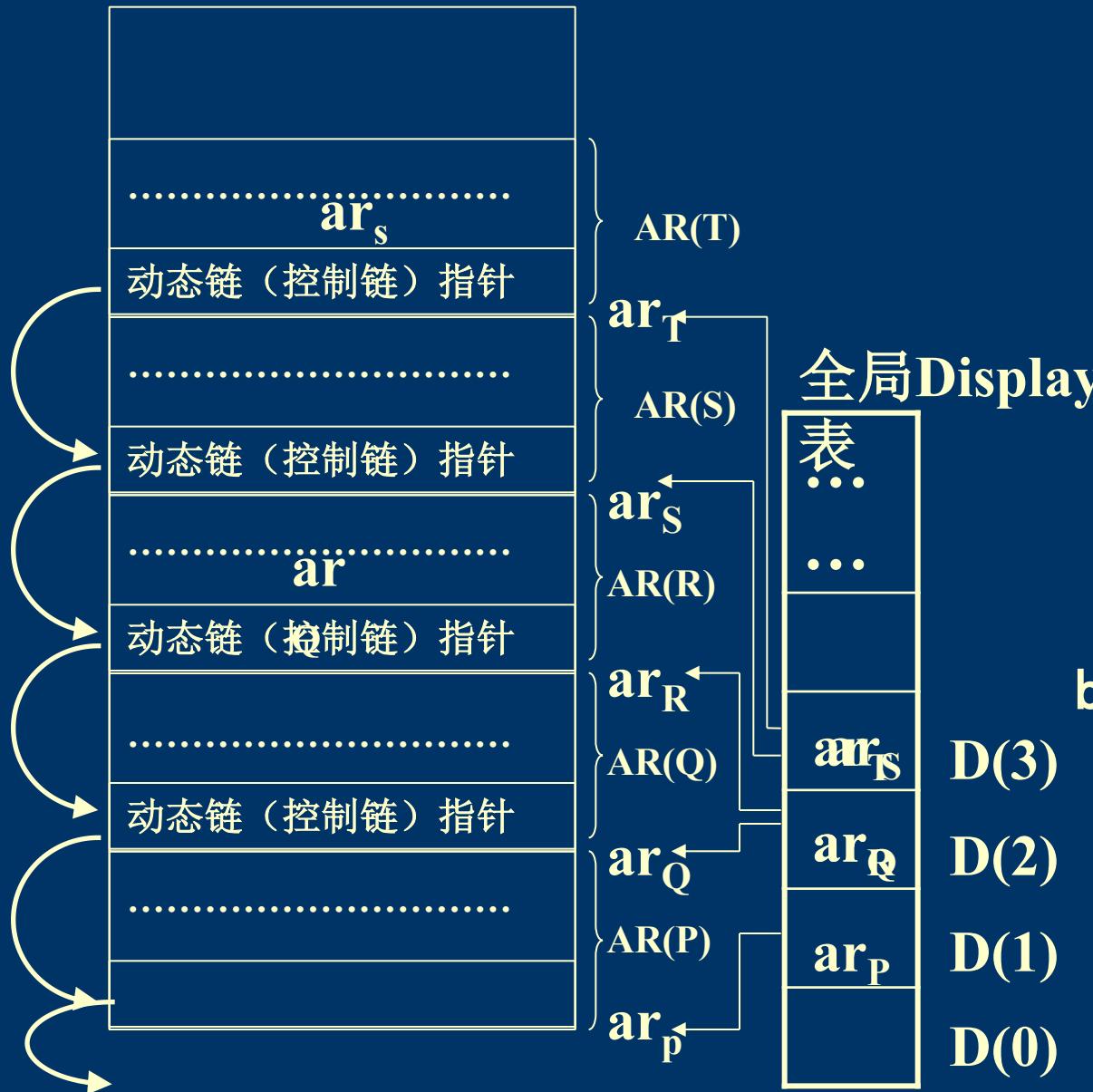
- ❖ 当层数为 j 的过程 Q 被调用时：
 1. 将旧的 D[j] 的内容保存到 NewAR(Q) 中：
 $\text{NewAR}(Q).\text{ResumeAddr} = D[j];$
 2. 改写 D[j] 的内容： $D[j] = \text{NewAR}(Q)$ 的地址；
- ❖ 当退出 Q 时：恢复原来 D[j] 的内容：
 $D[j] = \text{CurrentAR}.\text{ResumeAddr}$
- ❖ 局部变量 $X_{(L+1, off, \text{indir}/\text{dir})}$ 的访问
X 的地址：
 $\text{addr}(X) = D[L] + off$

过程 P \Rightarrow 过程 Q \Rightarrow 过程 R \Rightarrow 过程 S \Rightarrow 过程 T



```
proc P; (设P层数为1)
  proc Q; (Q层数为2)
    begin R end
  proc R; (R层数为2)
    proc S; (S层数为3)
      begin T end
    proc T; (T层数为3)
      begin X(2, offx) ...
        Y(3, offy) ...
        Z(4, offz) ...
      end
      begin S end
    begin Q end
  begin R end
end P
```

主程序 P \Rightarrow 过程 Q \Rightarrow 过程 R \Rightarrow 过程 S \Rightarrow 过程 T
proc P; (设P层数为1)
proc Q; Q层数为2
begin R end
proc R; R层数为2
proc S; S层数为3
begin T end
proc T; T层数为3
begin ... end
begin S end
begin Q end



静态链的方法

问题的提出

例：假设有调用链(M, G , H, R, S)，并且有

Level (M)=0, Level (G)=1, Level (H)=2, Level (R)=3, Level (S)=3

则相应动态链的基本结构应如下：

[AR₀(M), AR₁(G), AR₂(H), AR₃(R), AR₄(S)]

如果用局部Display表方法，则Display表情况如下：

AR₀(M). Display = [ar₀]

AR₁(G). Display = [ar₀, ar₁]

AR₂(H). Display = [ar₀, ar₁, ar₂]

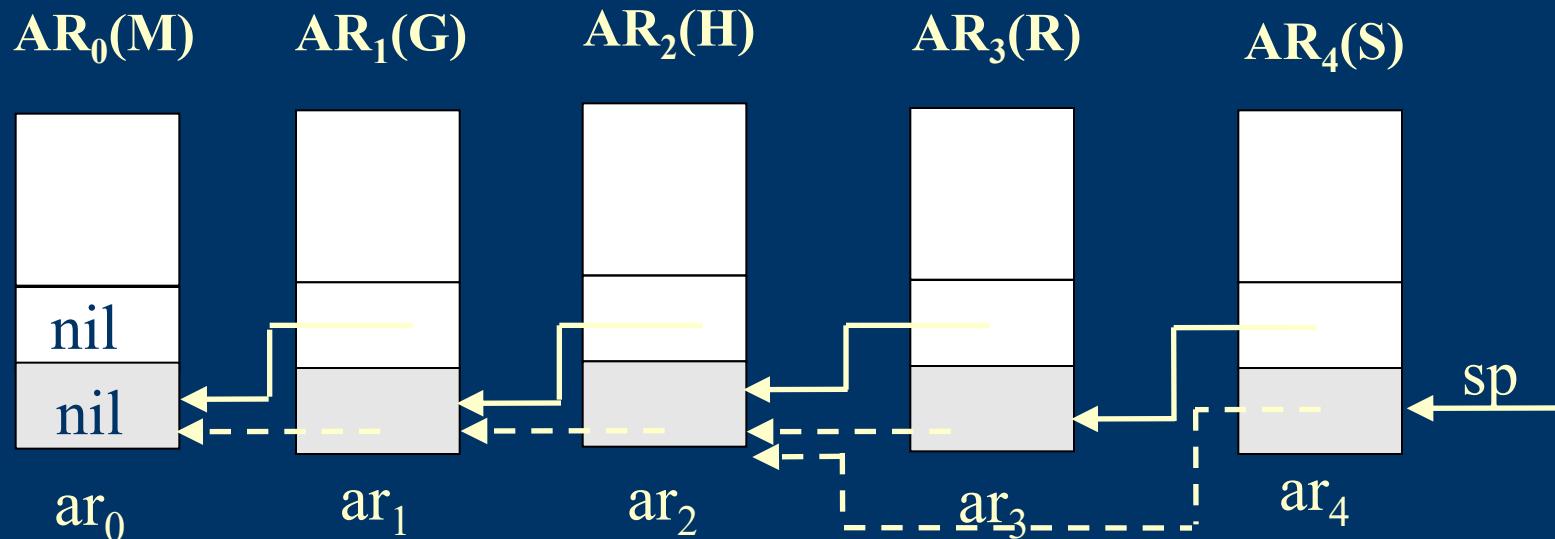
AR₃(R). Display = [ar₀, ar₁, ar₂, ar₃]

AR₄(S). Display = [ar₀, ar₁, ar₂, ar₄]

静态链的方法

- ❖ 原Display表部分变成一个单元，称为静态链单元，存放静态链指针。
- ❖ 静态链指针的确定：
若 $k = \text{CurrentAR.level} - (\text{NewAR.level} - 1)$ ，
则 $\text{NewAR.StaticChainPointer} = \text{Indir}(\text{sp}, k)$
其中 $\text{Indir}(\text{sp}, k)$ 表示 sp 的 k 次间接内容。

Level(M)=0, Level(G)=1, Level(H)=2,Level(R)=3,Level(S)=3



AR₀(M).Display
AR₁(G).Display
AR₂(H).Display
AR₃(R).Display
AR₄(S).Display

= [ar₀]
= [ar₀, ar₁]
= [ar₀, ar₁, ar₂]
= [ar₀, ar₁, ar₂, ar₃]
= [ar₀, ar₁, ar₂, ar₄]

虚线为静态链指针
实线为动态链指针

使用静态链时变量的访问

❖ 变量 $X(L+1, off)$ 的地址:

若 $L = \text{CurrentAR.Level}$, 则 $\text{addr}(X) = \text{sp} + off$

否则, $k = \text{CurrentAR.Level} - L$,

$\text{addr}(X) = \text{Indir}(\text{sp}, k) + off$

总结

Display表方法是用表结构表示变量访问环境。

- ❖ 局部Display表的产生需要花空间，但返回时不需要为恢复变量访问环境做任何事情。
- ❖ 对于全局Display表方法而言，Display表的产生需要花时间，而且返回时也需要为恢复变量访问环境而花时间，其主要优点是能节省存储单元。

总结

- 静态链方法是用链表表示变量访问环境
静态链方法实际上是一种共享化的局部Display表方法。其主要优点同全局Display表方法是能节省存储单元。产生需要花时间，但返回时不需要为恢复变量访问环境做任何事情。
- 具体采用哪种方法，取决于机器条件：如果寄存器较少，则使用Display表方法可能合适些；如果机器能提供较好的间接操作，则可选用静态链方法。