编译原理

动态存储管理

存储分配策略

- 静态分配策略
  - 在编译时能确定数据空间的大小，并为每个数据项目确定出在运行时刻的存储空间中的位置
  - FORTRAN等
- 动态分配策略
  - 在编译时不能确定运行时数据空间的大小，允许递归过程和动态申请释放内存
  - 栈式动态分配、堆式动态分配
  - PASCAL，C/C++等

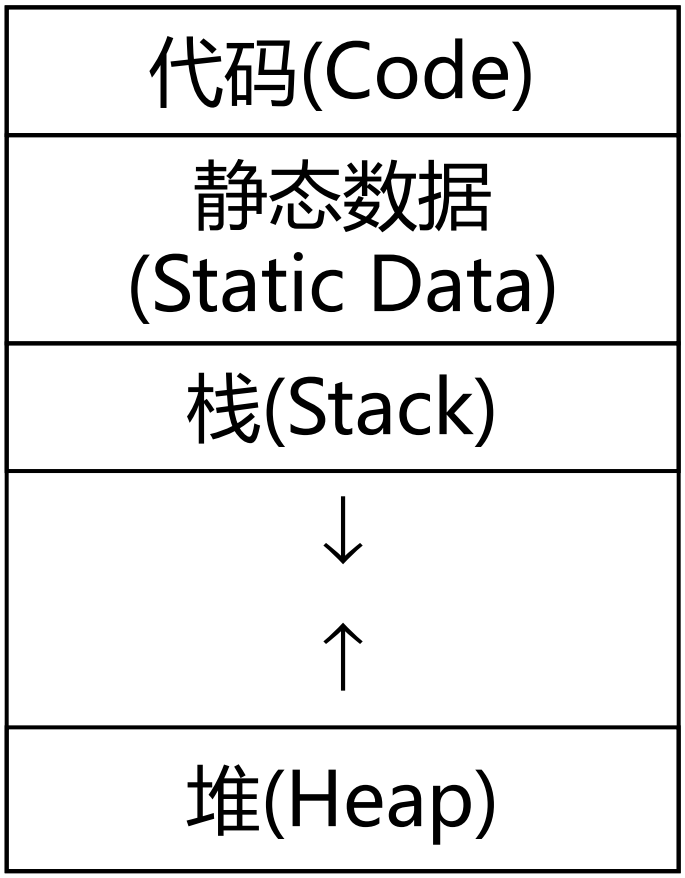
编译原理

运行时存储器的划分

运行时存储器的划分

- 一个目标程序运行所需的存储空间包括
  - 存放目标代码的空间
  - 存放数据项目的空间
  - 存放程序运行的控制或连接数据所需单元

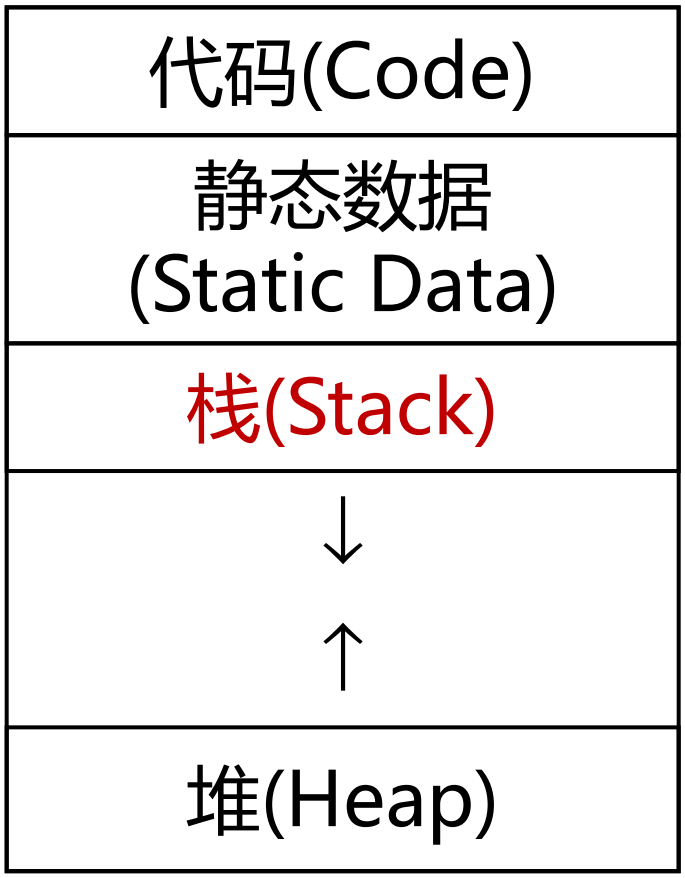
程序在运行时刻的内存划分



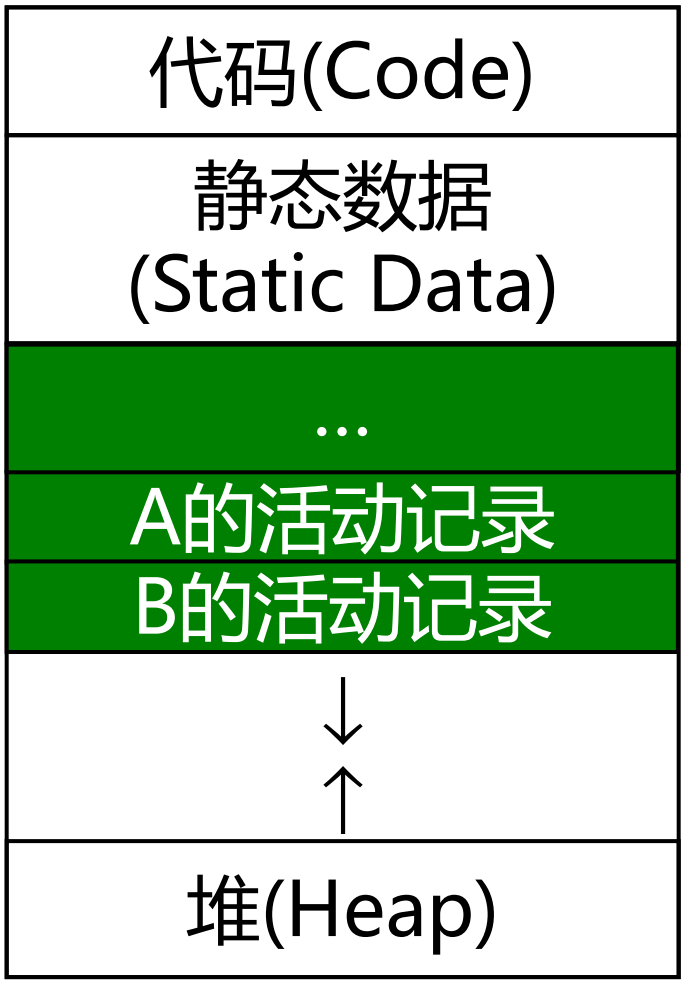
过程的活动与活动记录

- 一个过程的活动指的是该过程的一次执行
- 活动记录
  - 运行时，每当进入一个过程就有一个相应的活动记录累筑于栈顶
  - 此记录含有连接数据、形式单元、局部变量、局部数组的内情向量和临时工作单元等

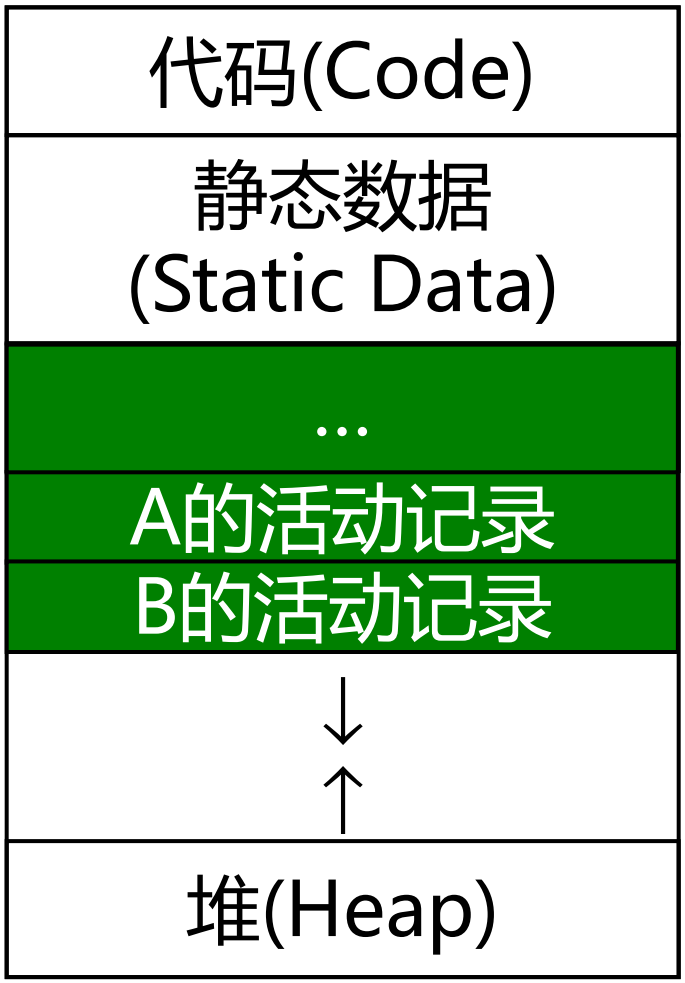
程序在运行时刻的内存划分



程序在运行时刻的内存划分



程序在运行时刻的内存划分



编译原理

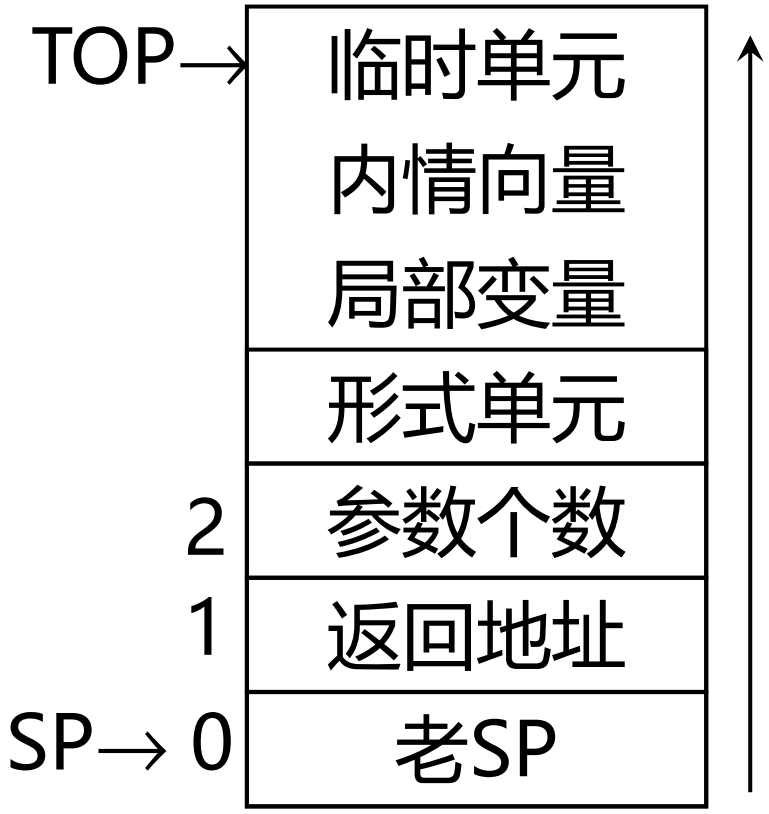
非嵌套过程语言的动态存储管理

非嵌套过程语言

- 特点
  - 允许过程递归调用、也可以允许过程含有可变数组
  - 过程定义不允许嵌套
  - 如C语言
- 采用栈式存储分配机制
- 活动记录
  - 运行时，每当进入一个过程就有一个相应的活动记录累筑于栈顶
  - 此记录含有连接数据、形式单元、局部变量、局部数组的内情向量和临时工作单元等

非嵌套过程语言的活动记录内容

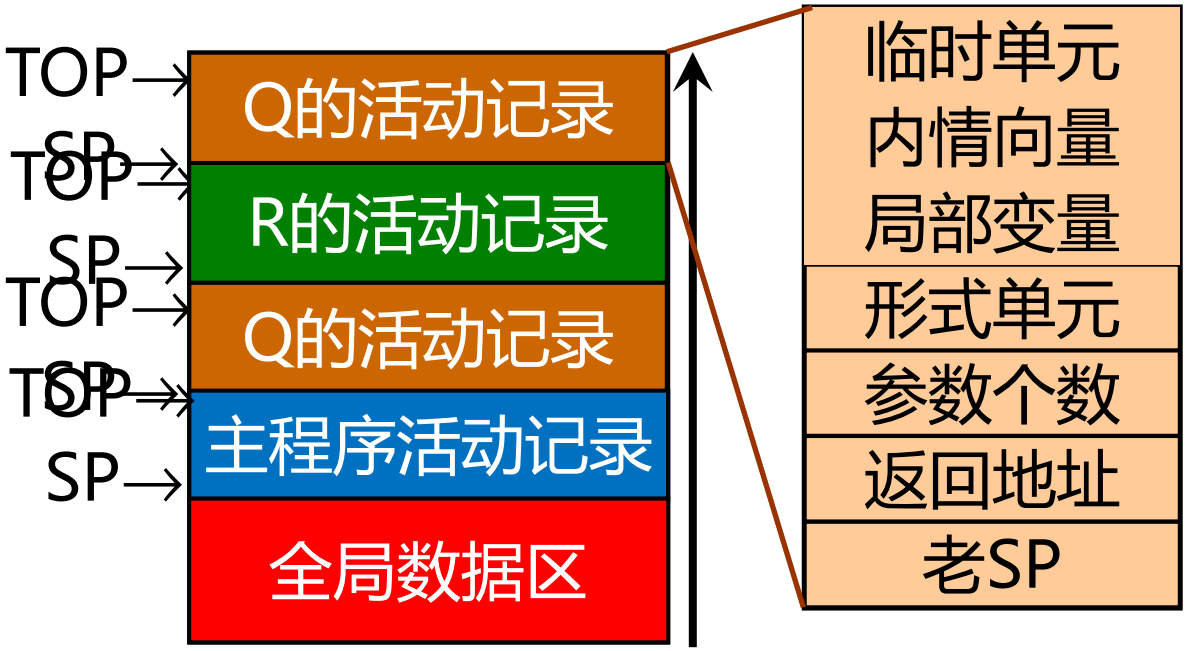
- 对任何局部变量X的引用可表示为变址访问: dx[SP]
- dx：编译时确定的变量X相对于活动记录起点的相对地址



非嵌套过程语言过程调用示例

```
全局数据说明
Main( ) {
    Main中的数据说明
    Main中的数据处理
}
void R( ) {
    R中的数据说明
    R中的数据处理
}
…
void Q( ) {
    Q中的数据说明
    Q中的数据说明
}
```

主程序 →过程Q
→过程R
→过程Q



过程调用和过程返回

- 过程调用的语句

$$P(T_1, T_2, \ldots, T_n)$$

- 翻译成四元式序列

par T_1
par T_2
...
par T_n
call P, n

par和call产生的目标代码

- 每个par T_i(i=1,2,...n)可直接翻译成如下指令:
  (i+3)[TOP]:= T_i (传值)
  (i+3)[TOP]:=addr(T_i) (传地址)

- call P，n 被翻译成:

1[TOP]:=SP (保护现行SP)

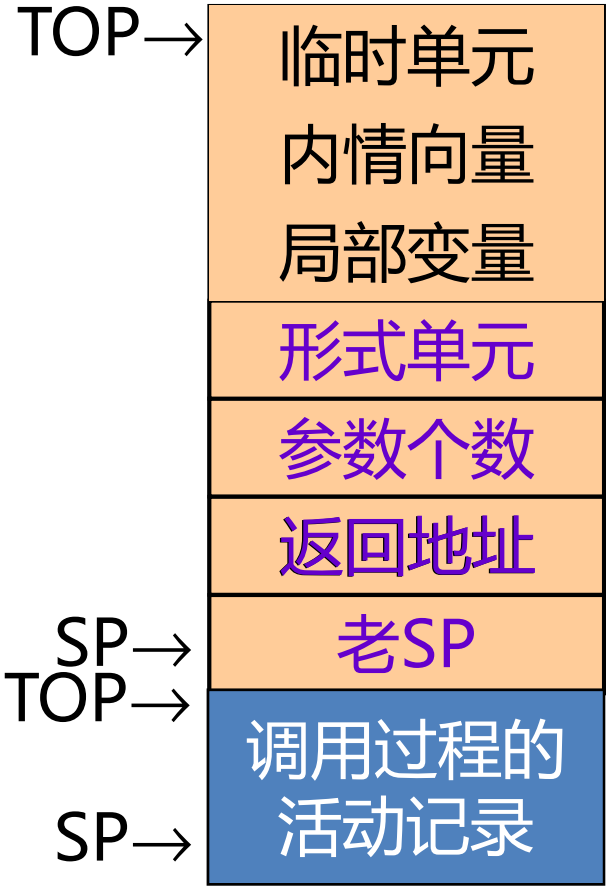
3[TOP]:=n (传递参数个数)

JSR P (转子指令)

进入过程体后执行的指令

- 转进过程P后，首先执行下述指令
  SP:=TOP+1　　(定义新的SP)
  1[SP]:=返回地址　(保护返回地址)
  TOP:=TOP+L　　(设置新TOP)

  L：过程P的活动记录所需单元数，在编译时可确定。



过程返回执行的指令

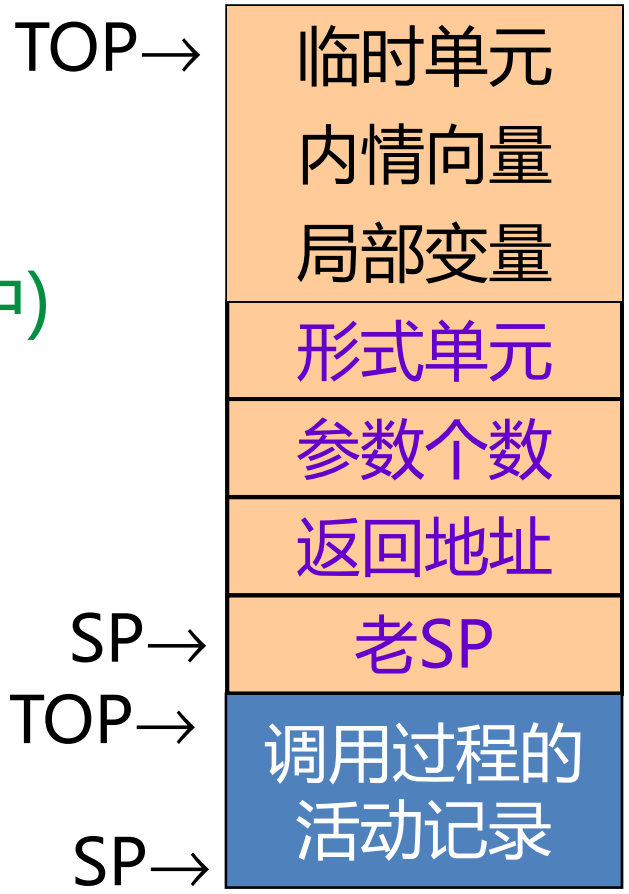
- 过程返回时，应执行下列指令:

  TOP:=SP-1　(恢复调用前TOP)

  SP:=0[SP]　(恢复调用前SP)

  X:=2[TOP]　(把返回地址取到X中)

  UJ 0[X]　　(按X返回)



小结

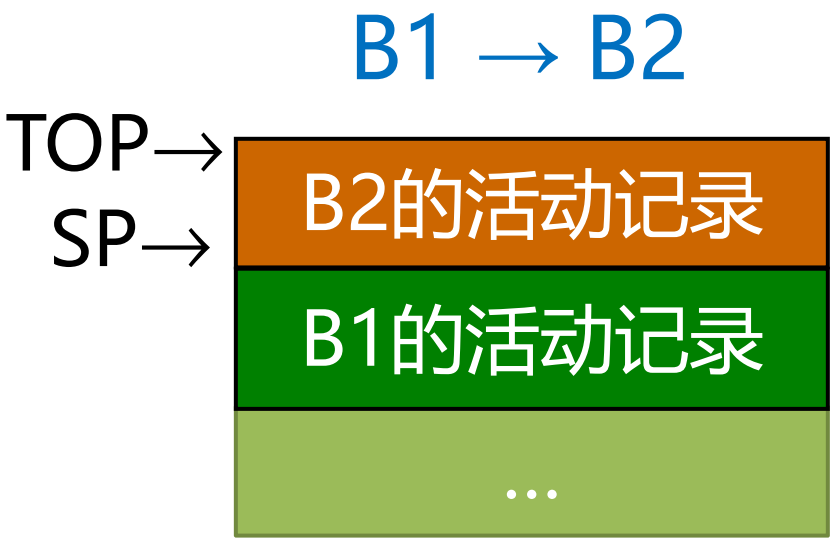
- 非嵌套过程语言的动态存储管理
  - 非嵌套过程语言的特点
  - 栈式管理与活动记录
  - 过程调用和过程返回

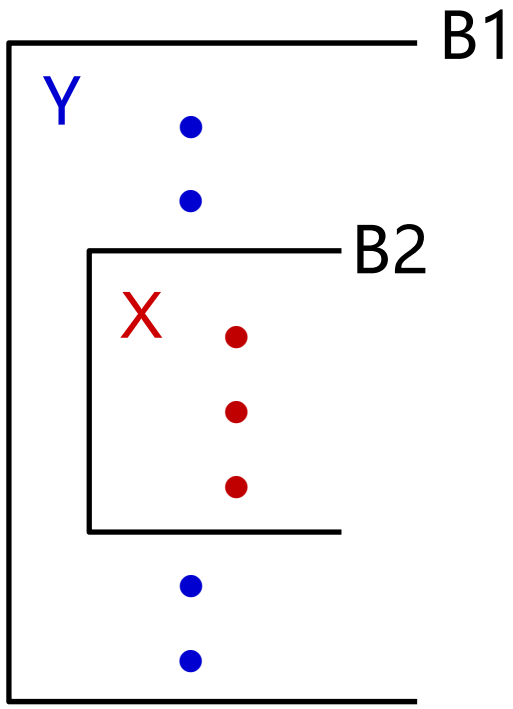
编译原理

嵌套过程语言的动态存储管理

嵌套过程语言

- 特点
  - 允许过程递归调用、也可以允许可变数组
  - 过程定义允许嵌套
  - 如PASCAL语言





嵌套过程语言

- 特点
  - 允许过程递归调用、也可以允许可变数组
  - 过程定义允许嵌套
  - 如PASCAL语言
- 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

编译原理

嵌套过程语言的动态存储管理
——静态链方法

嵌套层次与地址

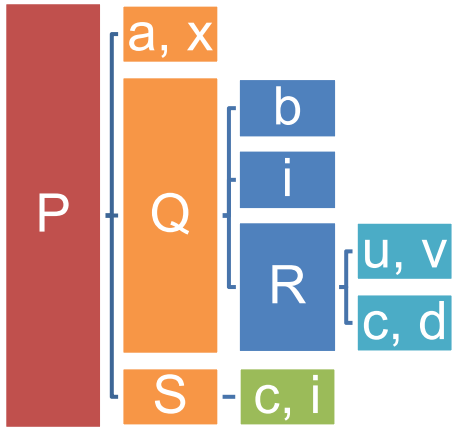
- 嵌套层次
  - 主程序的层次为0
  - 在i层中定义的过程，其层次为i+1
- 指令中的地址描述为 (层数， 偏移量)

示例程序

主程序P → 过程 S → 过程 Q
→ 过程 R →过程 R

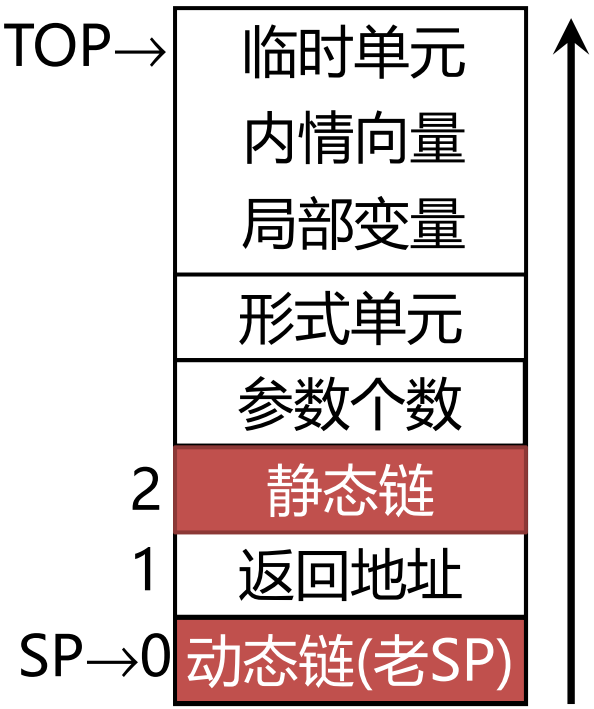
过程运行时,必须能知道它的所有外层过程的当前活动记录的起始地址

```
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);        [a(0, 3)、b(1, 4)]
          ……
        end {R}
    begin
        ……
        R(1, x);                 [x(0, 4)]
        ……
    end {Q}
    procedure S;
        var c, i:integer;
    begin                        [a(0, 3) ]
        a:=1;
        Q(c);
        ……
    end {S}
begin                            [a(0, 3) ]
    a:=0;
    S;
    ……
end. {P}
```



静态链和活动记录

- 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。
- 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。

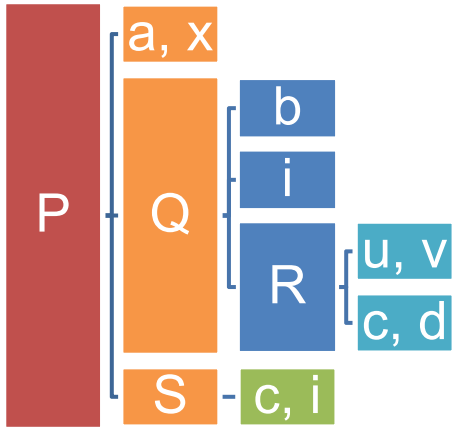


示例程序

主程序P → 过程 S → 过程 Q
→ 过程 R →过程 R

过程运行时,必须能知道它的所有外层过程的当前活动记录的起始地址

```
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);        [a(0, 3)、b(1, 4)]
          ……
        end {R}
    begin
        ……
        R(1, x);                 [x(0, 4)]
        ……
    end {Q}
    procedure S;
        var c, i:integer;
    begin
        a:=1;                    [a(0, 3) ]
        Q(c);
        ……
    end {S}
begin
    a:=0;                        [a(0, 3) ]
    S;
    ……
end. {P}
```



主程序P

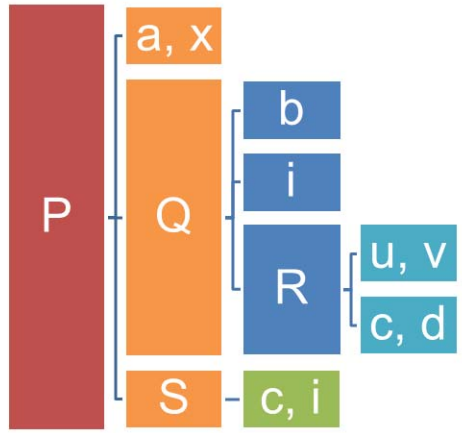
```
program P;
    var a, x : integer;
    procedure Q(b: integer);
      var i: integer;
      procedure R(u: integer; var v: integer);
        var c, d: integer;
      begin
        if u=1 then R(u+1, v)
        ……
        v:=(a+c)*(b-d);
        ……
      end {R}
    begin
      ……
      R(1, x);
      ……
    end {Q}
    procedure S;
      var c, i:integer;
    begin
      a:=1;
      Q(c);
      ……
    end {S}
begin
    a:=0;
    S;
    ……
end. {P}
```

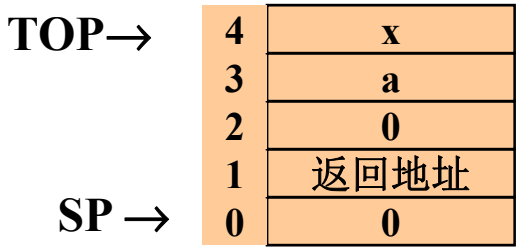
[a(0, 3)、b(1, 4)]

[x(0, 4)]

[a(0, 3)]

[a(0, 3)]

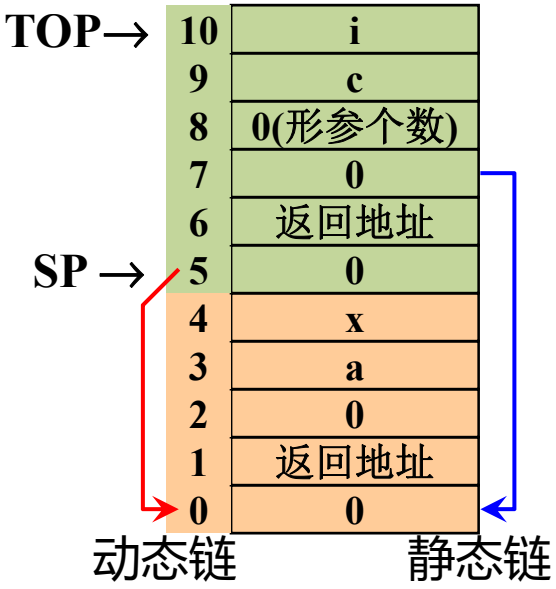




主程序P→过程 S

? 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)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.



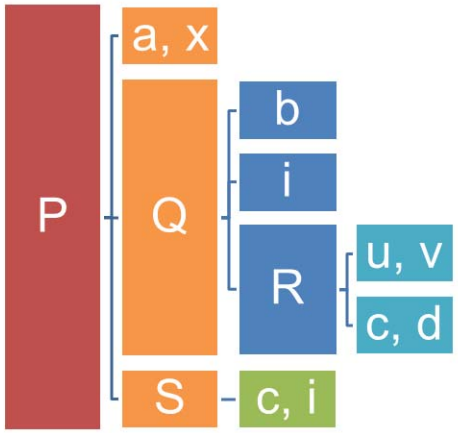

```
program P;
    var a, x : integer;
    procedure Q(b: integer);
        var i: integer;
        procedure R(u: integer; var v: integer);
            integer;
        hen R(u+1, v)
        )*(b-d);
    begin
        R(1, x);
    end {Q}
    procedure S;
        var c, i:integer;
    begin
        a:=1;
        Q(c);
    end {S}
begin
    a:=0;
    S;
end. {P}
```

[a(0, 3)、b(1, 4)]

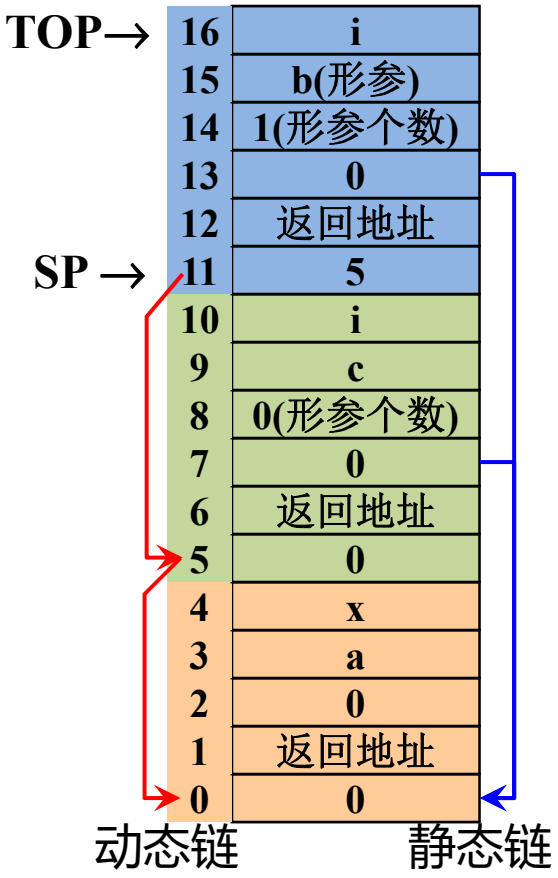
[x(0, 4)]

[a(0, 3)]

[a(0, 3)]



主程序P→过程 S→过程 Q



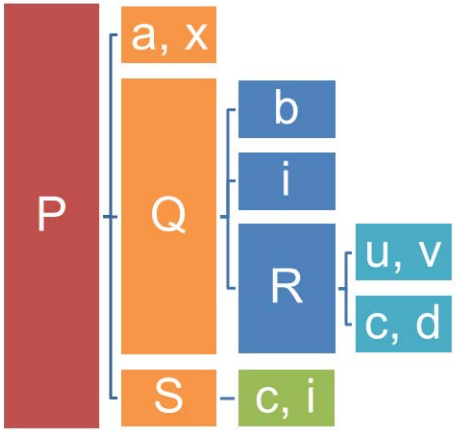
? 第N层过程调用第N层过程，如何确定被调用过程(第 N层)的静态链?

A:调用过程(第N层过程)的静态链的值。

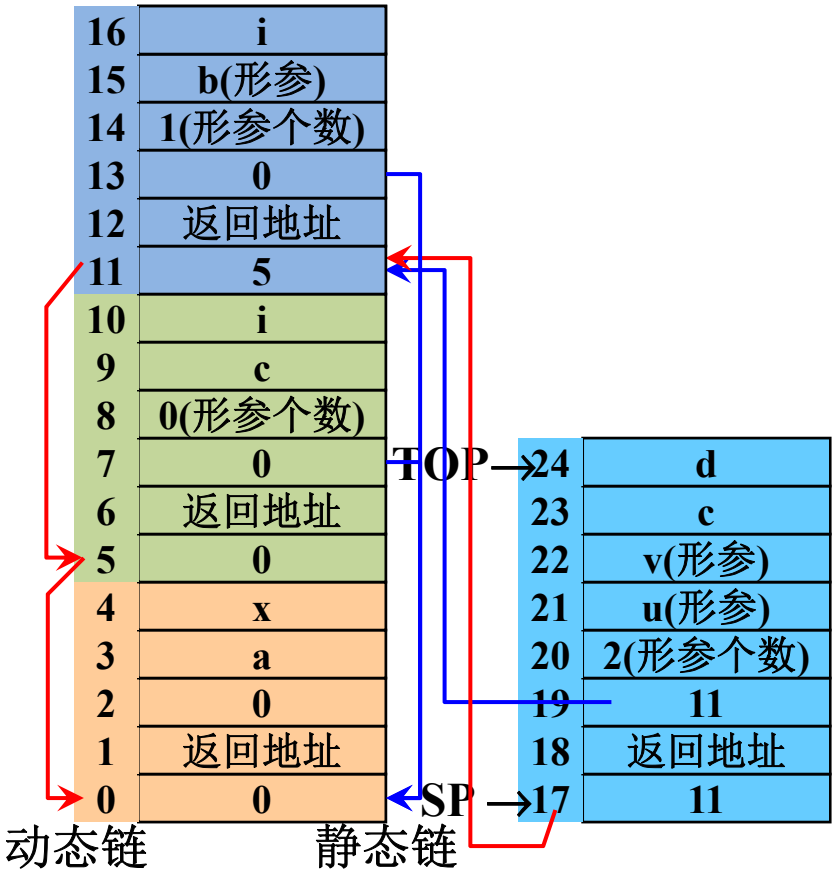
```
program P;
    var a, x : integer;
    procedure Q(b: integer);
        var i: integer;
        procedure R(u: integer; var v: integer);
        ...
        R(u+1, v)
        ...
        -d);
        ...
        [a(0, 3)、b(1, 4)]
        ...
        [x(0, 4)]
        ...
        er;
        begin
            a:=1;
            Q(c);
            ...
        end {S}
begin
    a:=0;
    S;
    ...
end. {P}
```

[a(0, 3)]

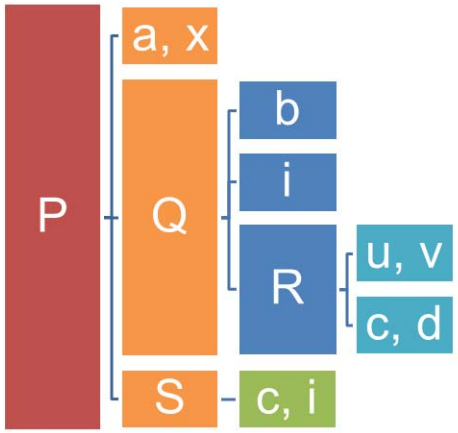
[a(0, 3)]



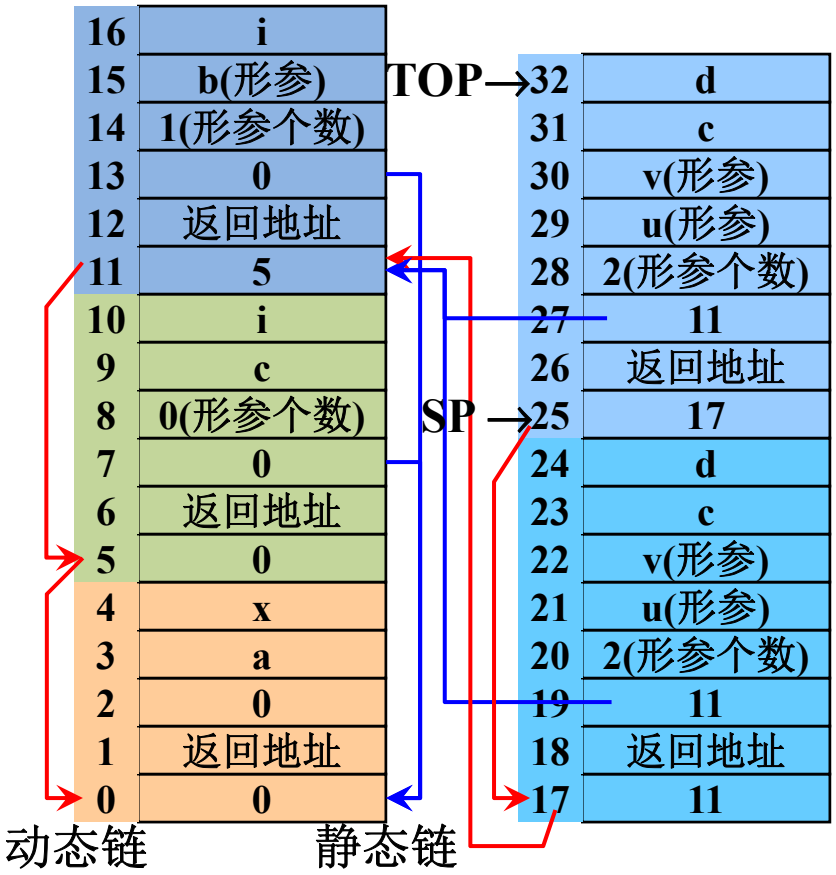
主程序P→过程 S→过程 Q
→过程 R



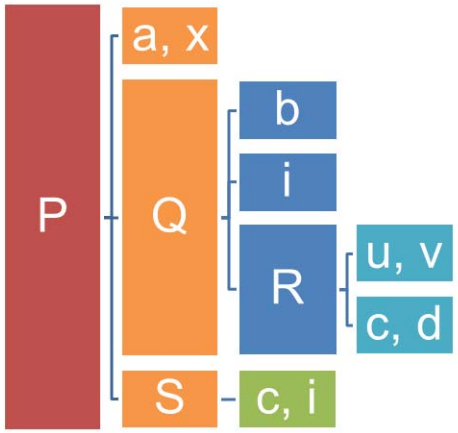

```
program P;
    var a, x : integer;
    procedure Q(b: integer);
      var i: integer;
      procedure R(u: integer; var v: integer);
        var c, d: integer;
      begin
        if u=1 then R(u+1, v)
        ……
        v:=(a+c)*(b-d);          [a(0, 3)、b(1, 4)]
        ……
      end {R}
    begin
      ……
      R(1, x);                   [x(0, 4)]
      ……
    end {Q}
    procedure S;
      var c, i:integer;
    begin                        [a(0, 3) ]
      a:=1;
      Q(c);
      ……
    end {S}
begin                            [a(0, 3) ]
    a:=0;
    S;
    ……
end. {P}
```



主程序P→过程 S→过程 Q
→过程 R→过程 R




```
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);          [a(0, 3)、b(1, 4)]
            ……
        end {R}
    begin
        ……
        R(1, x);                     [x(0, 4)]
        ……
    end {Q}
    procedure S;
        var c, i:integer;
    begin                            [a(0, 3) ]
        a:=1;
        Q(c);
        ……
    end {S}
begin                                [a(0, 3) ]
    a:=0;
    S;
    ……
end. {P}
```



测试

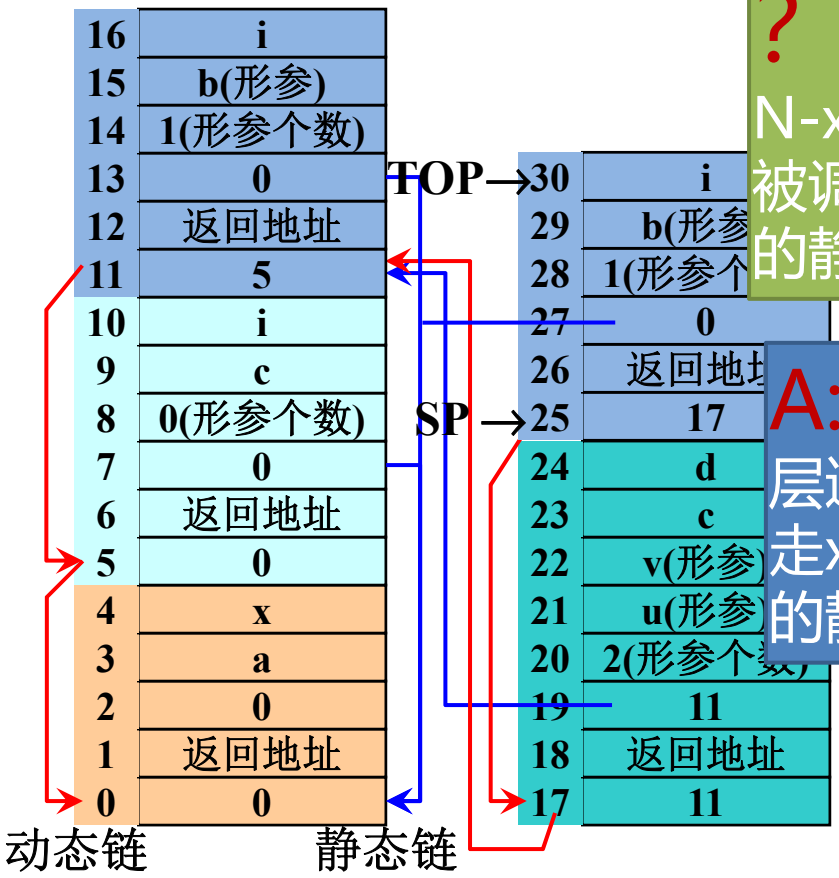
- 对于图示的程序，过程R能否调用过程Q?

A. 不可以

B. 可以

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

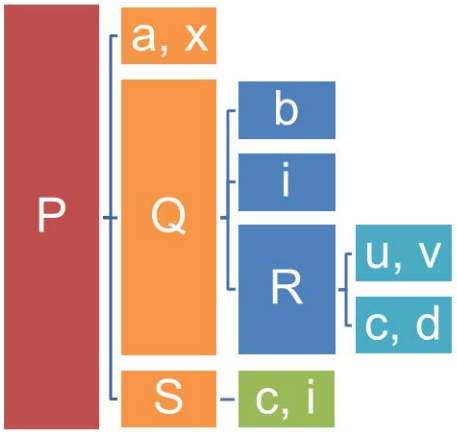
主程序P→过程 S→过程 Q
→过程 R→过程 Q



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

A:沿着调用过程(第N层过程)的静态链向前走x步到达的活动记录的静态链的值。

```
program P;
    var a, x : integer;
    procedure Q(b: integer);
      var i: integer;
                                  nteger; var v: integer);
                                  +1, v)
                                                [a(0, 3)、b(1, 4)]
                                                [x(0, 4)]
                                                ) ]
    end {S}
begin
    a:=0;                                       [a(0, 3) ]
    S;
    ......
end. {P}
```

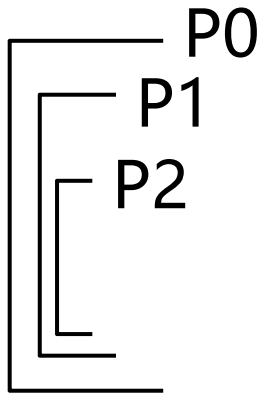


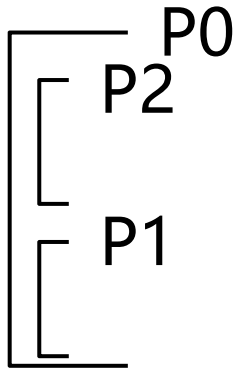
嵌套过程语言的动态存储管理

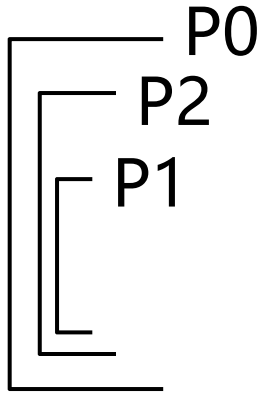
- 非局部名字的访问的实现
  - 静态链和活动记录
  - 根据调用过程的信息建立被调用过程的静态链

静态链的建立和维护

- 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)的静态链向前走x步到达的活动记录的静态链的值





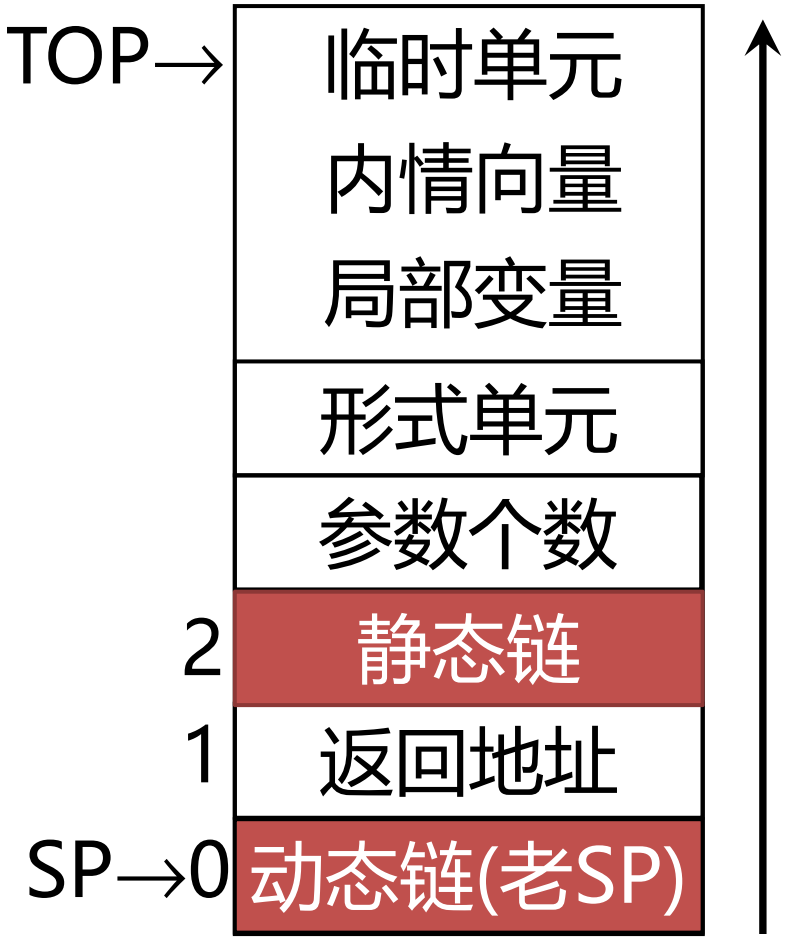


编译原理

嵌套过程语言的动态存储管理
——Display表方法

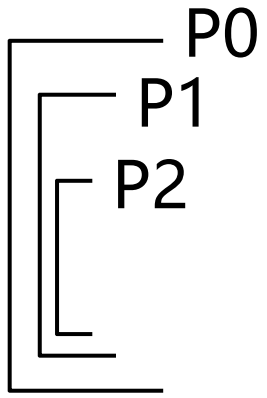
静态链方法访问非局部名字

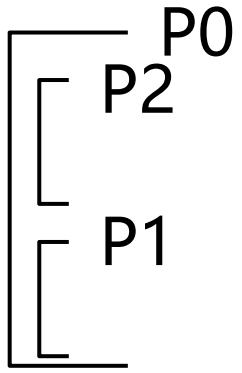
- 非局部名字的访问
  - 静态链和活动记录
  - 根据调用过程的活动记录信息建立被调用过程的静态链

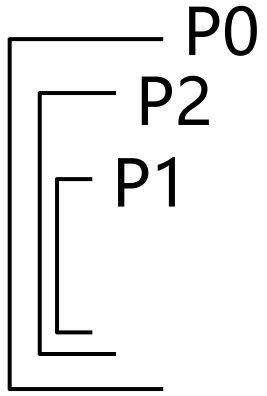


静态链的建立和维护

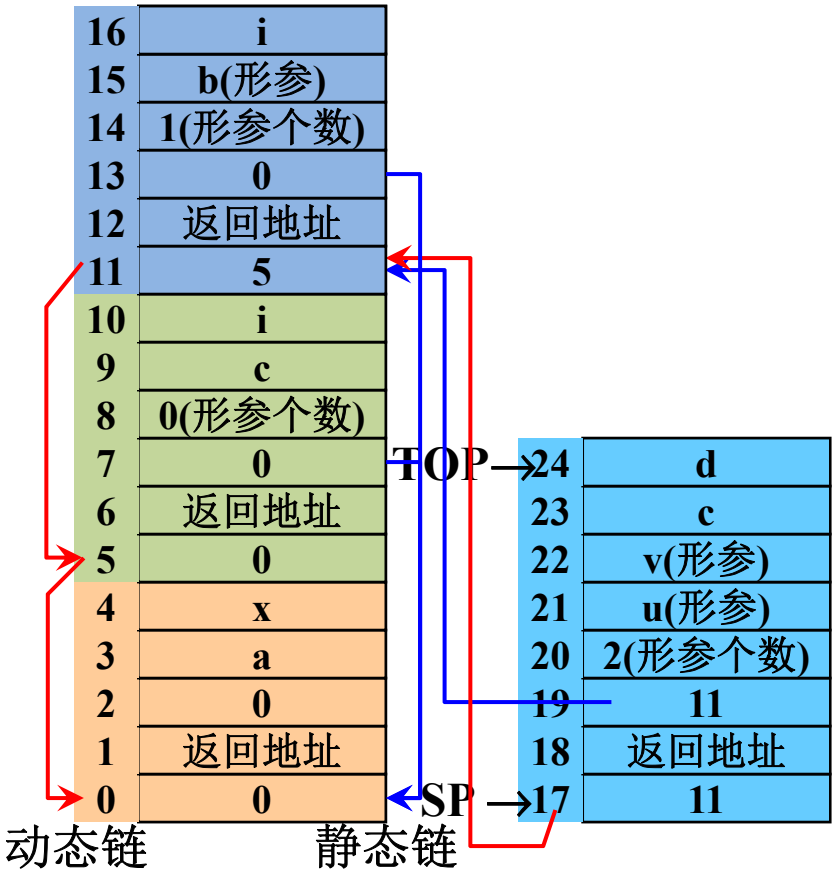
- 如何根据调用过程P1的活动记录信息建立被调用过程P2的静态链?
  - 第N层过程调用第 N+1层过程： P2的静态链为调用过程P1(第N层过程)的最新活动记录的起始地址
  - 第N层过程调用第 N层过程： P2的静态链为调用过程P1(第N层过程)的静态链的值
  - 第N层过程调用第 N-x层过程： P2的静态链为沿着调用过程P1(第N层过程)的静态链向前走x步到达的活动记录的静态链的值



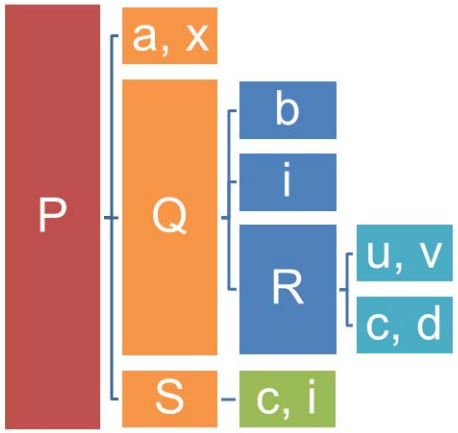




主程序P→过程 S→过程 Q
→过程 R

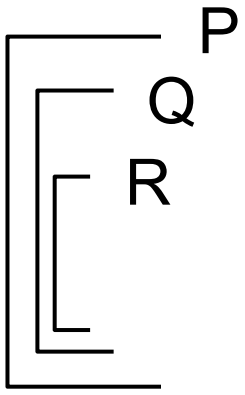


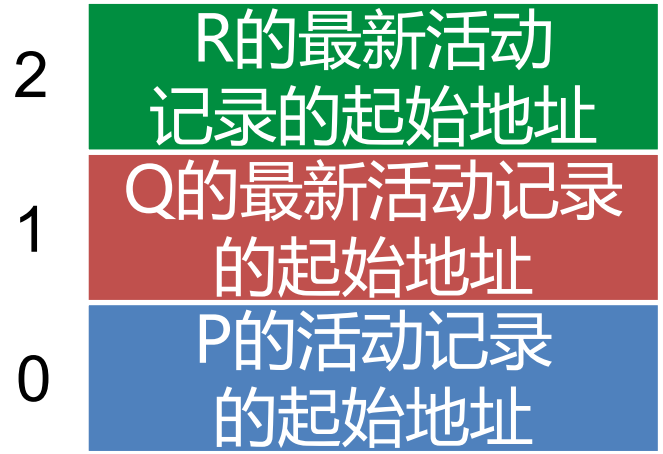

```
program P;
    var a, x : integer;
    procedure Q(b: integer);
      var i: integer;
      procedure R(u: integer; var v: integer);
        var c, d: integer;
      begin
        if u=1 then R(u+1, v)
        ……
        v:=(a+c)*(b-d);          [a(0, 3)、b(1, 4)]
        ……
      end {R}
    begin
      ……
      R(1, x);                   [x(0, 4)]
      ……
    end {Q}
    procedure S;
      var c, i:integer;
    begin                        [a(0, 3) ]
      a:=1;
      Q(c);
      ……
    end {S}
begin                            [a(0, 3) ]
    a:=0;
    S;
    ……
end. {P}
```



嵌套层次显示表Display

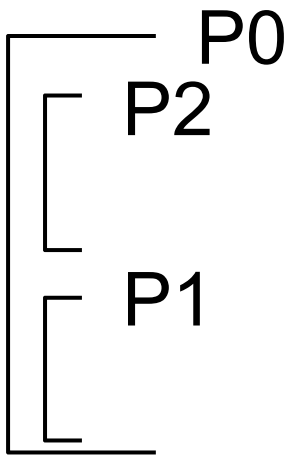
- 在活动记录中增加Display表
  - 进入一个过程时，在建立其活动记录的同时建立一张嵌套层次显示表Display，自顶向下依次存放着当前过程、直接外层、…、直至最外层(主程序,0层)过程的最新活动记录的起始地址
  - 若过程R的外层为Q，Q的外层为主程序为P，则过程R运行时的Display表内容为

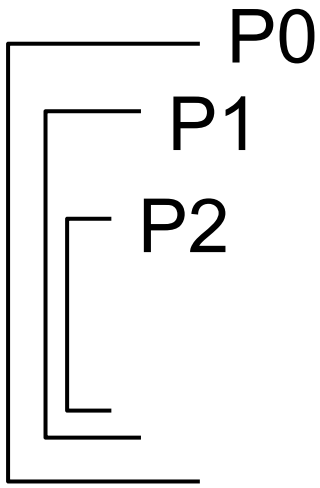


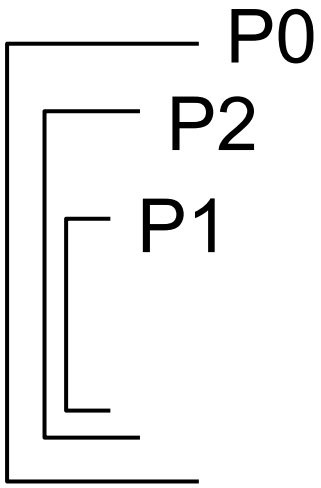


如何维护嵌套层次显示表Display

- 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?

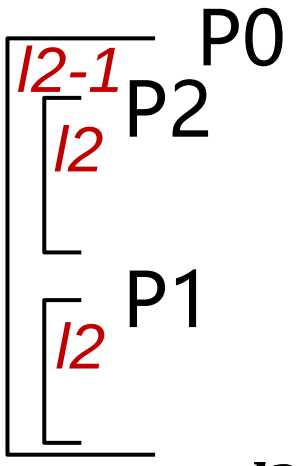




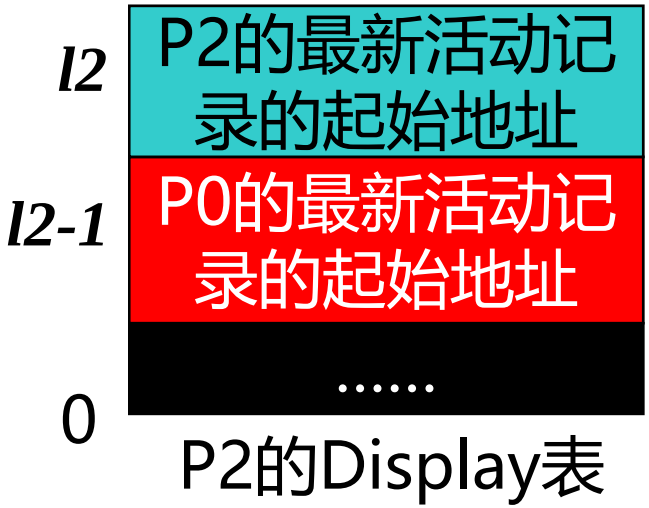


如何维护嵌套层次显示表Display

- 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?

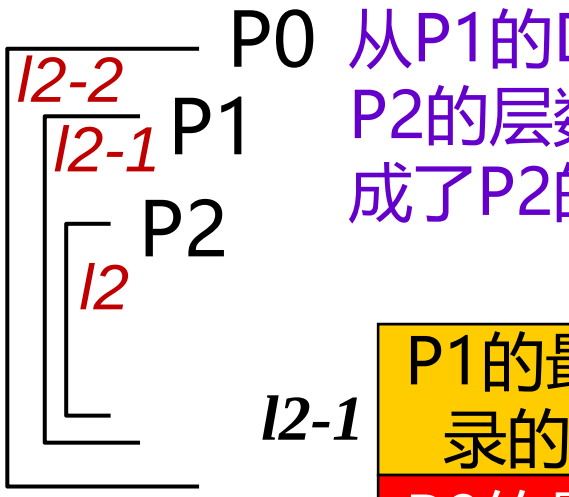


从P1的Display表中自底而上取*l2*个单元(*l2*为P2的层数)再添上进入P2后新建立的SP值就构成了P2的Display表。

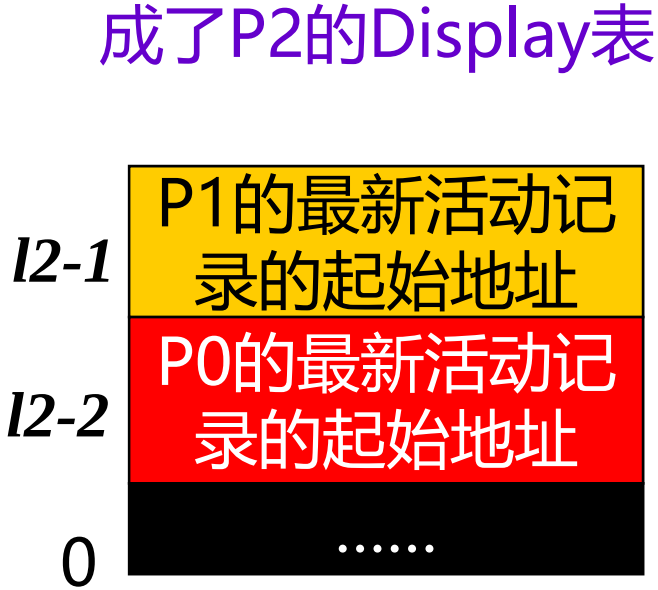


如何维护嵌套层次显示表Display

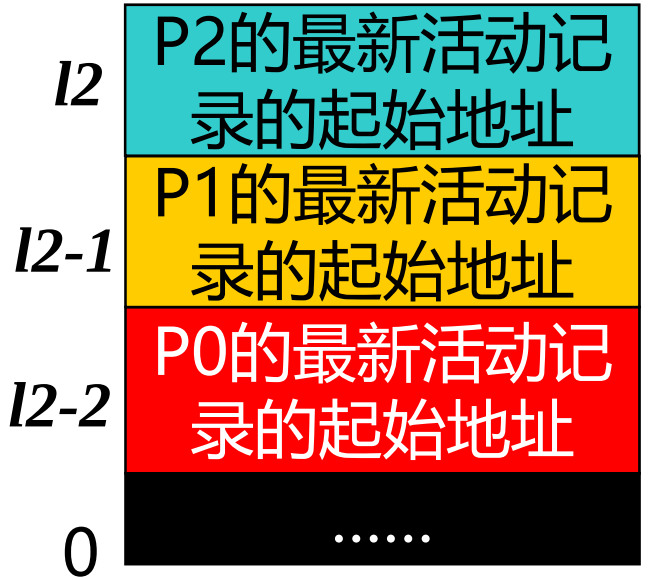
- 问题：当过程P1调用过程P2(*l2*层)而进入P2后，P2应如何建立起自己的Display表?



从P1的Display表中自底而上取*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。



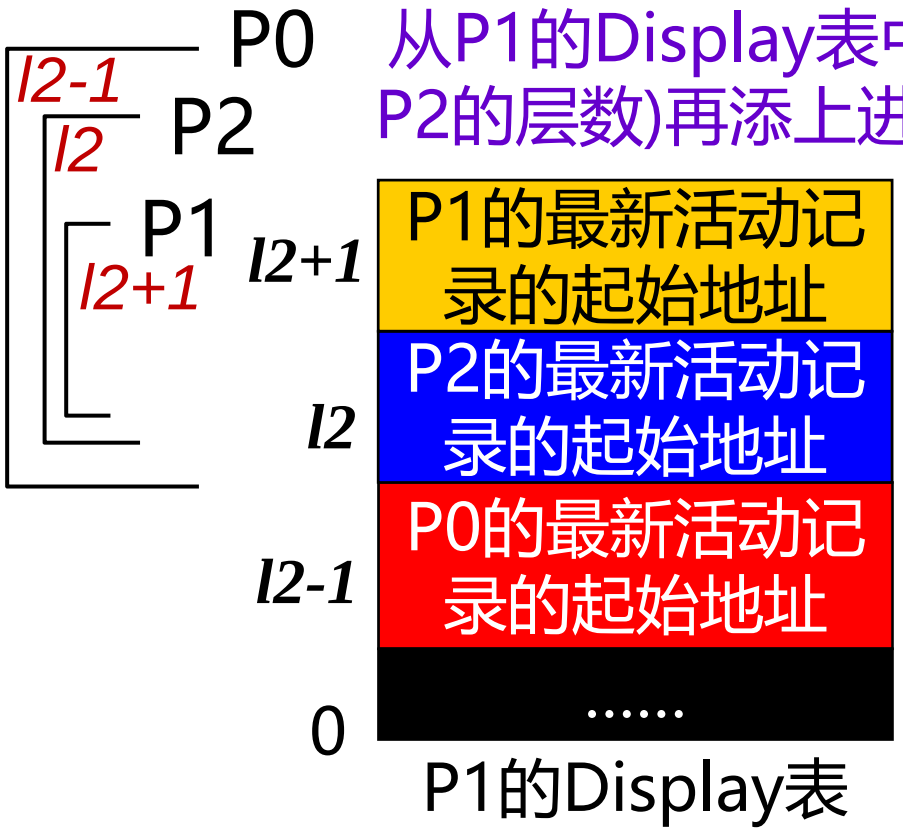
P1的Display表



P2的Display表

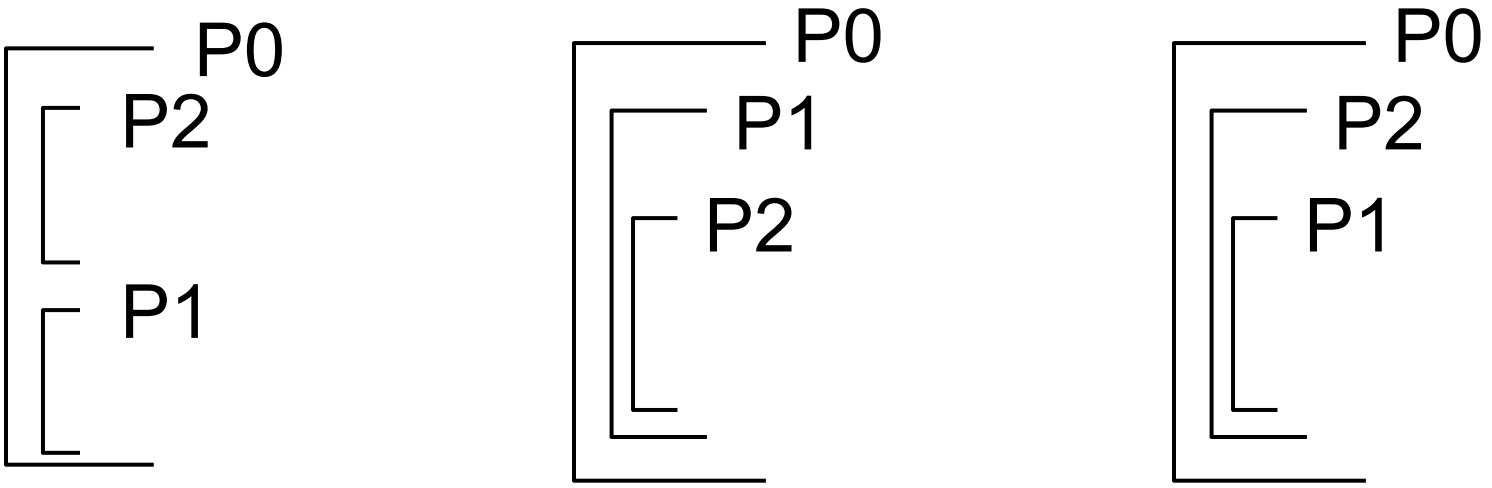
如何维护嵌套层次显示表Display

- 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?



如何维护嵌套层次显示表Display

- 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?



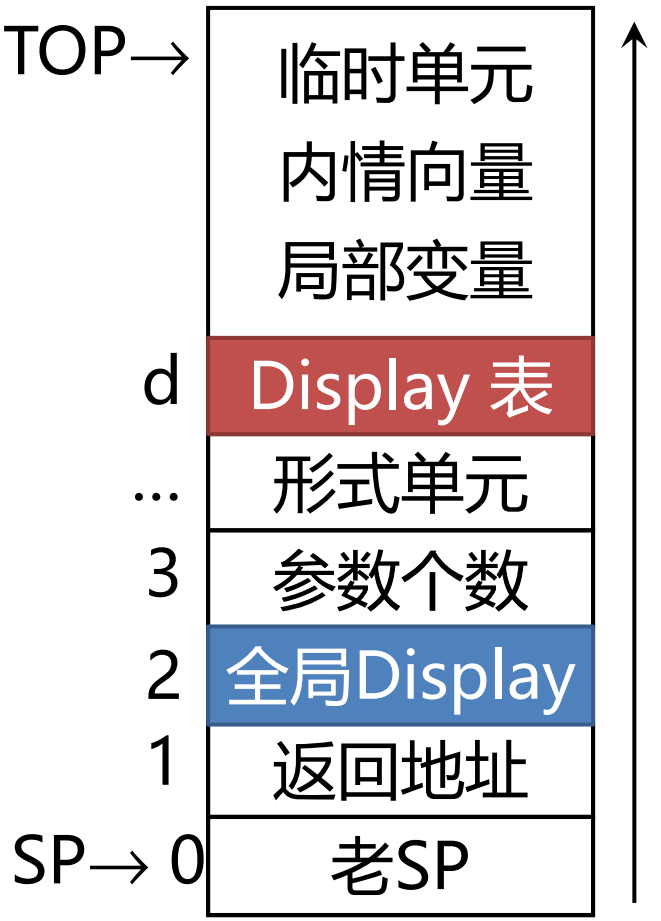
答案：从P1的Display表中自底而上取*l2*个单元(*l2*为P2的层数)再添上进入P2后新建立的SP值就构成了P2的Display表。把P1的display表地址(全局Display)作为连接数据之一传送给P2，然后建立P2的Display表。

Display表和活动记录

- 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(设层次为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]

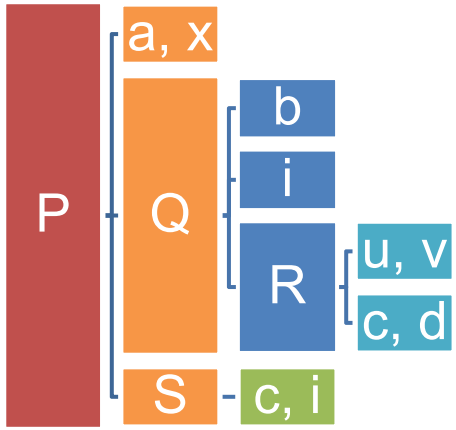


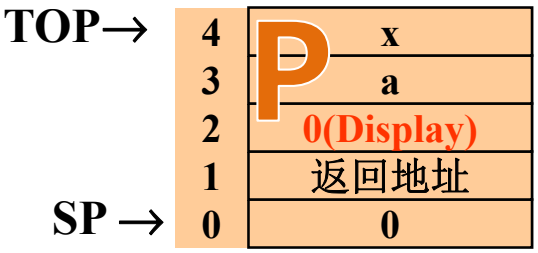
示例程序

主程序P → 过程 S → 过程 Q
→ 过程 R →过程 R

过程运行时通过Display表知道它的所有外层过程的当前活动记录的起始地址

```
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);        [a(0, 3)、b(1, 4)]
          ……
        end {R}
    begin
        ……
        R(1, x);                 [x(0, 4)]
        ……
    end {Q}
    procedure S;
        var c, i:integer;
    begin                        [a(0, 3) ]
        a:=1;
        Q(c);
        ……
    end {S}
begin                            [a(0, 3) ]
    a:=0;
    S;
    ……
end. {P}
```





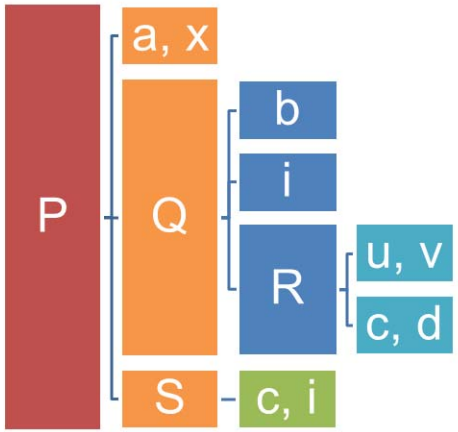

```
program P;
    var a, x : integer;
    procedure Q(b: integer);
      var i: integer;
      procedure R(u: integer; var v: integer);
        var c, d: integer;
      begin
        if u=1 then R(u+1, v)
        ……
        v:=(a+c)*(b-d);
        ……
      end {R}
    begin
      ……
      R(1, x);
      ……
    end {Q}
    procedure S;
      var c, i:integer;
    begin
      a:=1;
      Q(c);
      ……
    end {S}
begin
    a:=0;
    S;
    ……
end. {P}
```

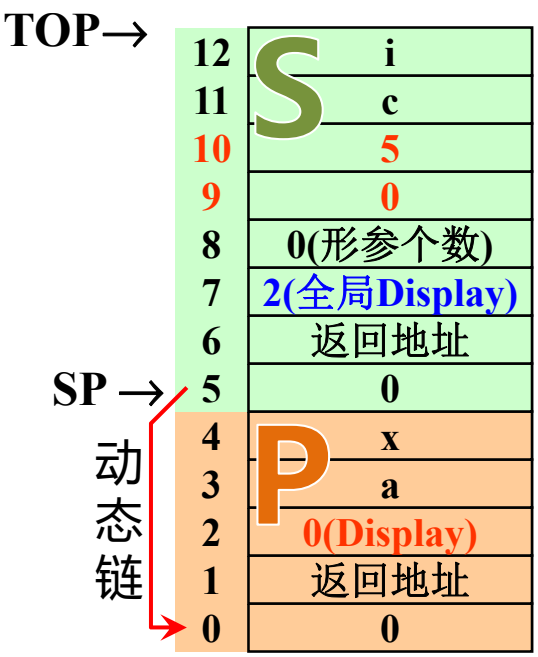
[a(0, 3)、b(1, 4)]

[x(0, 4)]

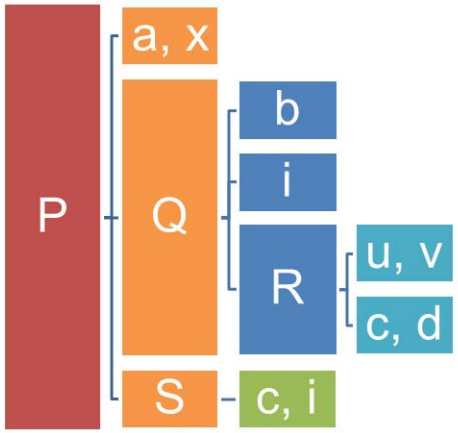
[a(0, 3)]

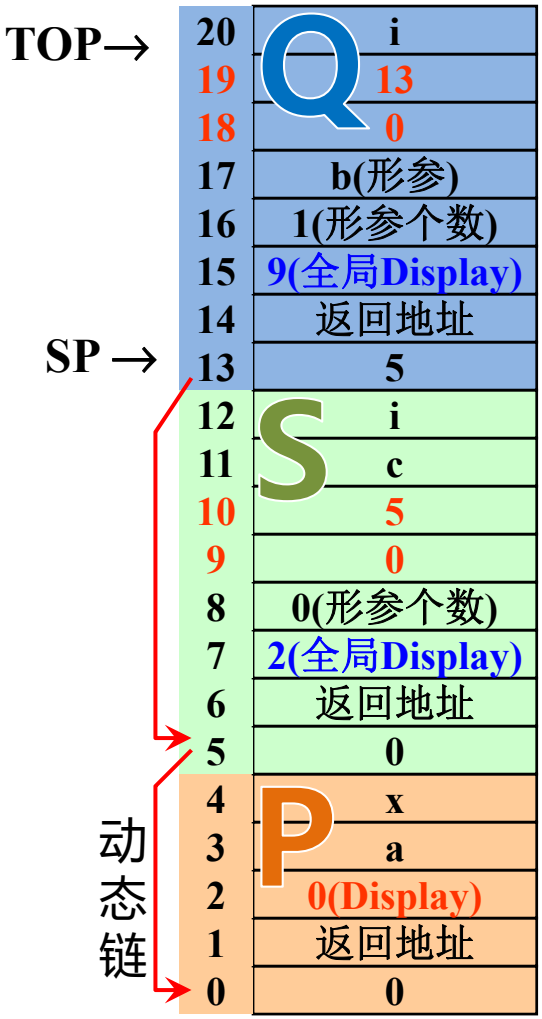
[a(0, 3)]



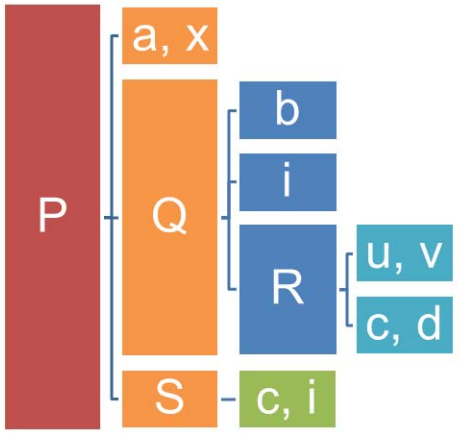


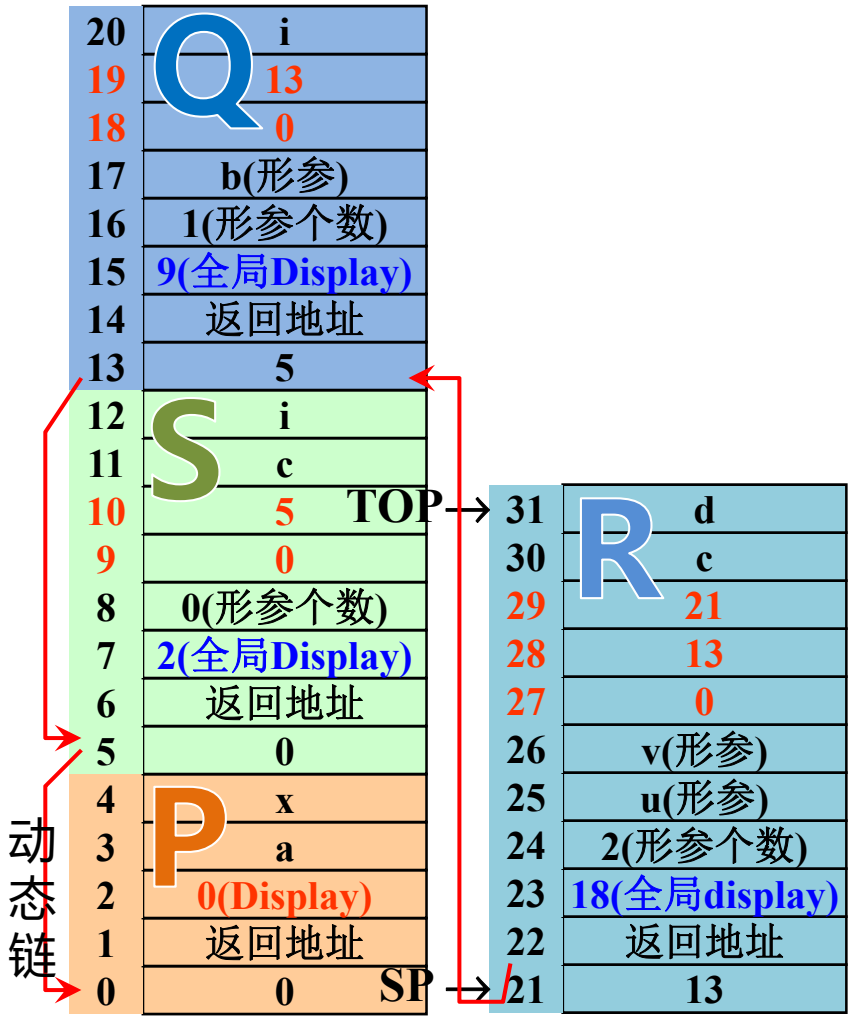

```
program P;
    var a, x : integer;
    procedure Q(b: integer);
      var i: integer;
      procedure R(u: integer; var v: integer);
        var c, d: integer;
      begin
        if u=1 then R(u+1, v)
        ……
        v:=(a+c)*(b-d);          [a(0, 3)、b(1, 4)]
        ……
      end {R}
    begin
      ……
      R(1, x);                   [x(0, 4)]
      ……
    end {Q}
    procedure S;
      var c, i:integer;
    begin                        [a(0, 3) ]
      a:=1;
      Q(c);
      ……
    end {S}
begin                            [a(0, 3) ]
    a:=0;
    S;
    ……
end. {P}
```



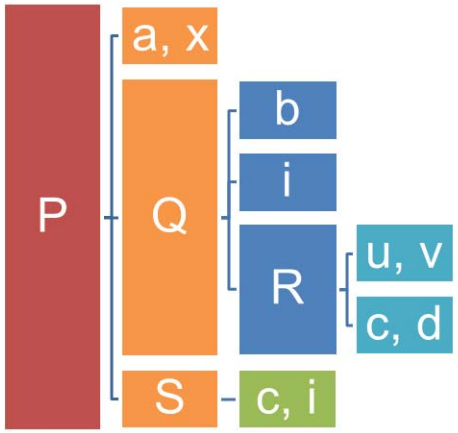



```
program P;
    var a, x : integer;
    procedure Q(b: integer);
      var i: integer;
      procedure R(u: integer; var v: integer);
        var c, d: integer;
      begin
        if u=1 then R(u+1, v)
        ……
        v:=(a+c)*(b-d);          [a(0, 3)、b(1, 4)]
        ……
      end {R}
    begin
      ……
      R(1, x);                   [x(0, 4)]
      ……
    end {Q}
    procedure S;
      var c, i:integer;
    begin
      a:=1;                      [a(0, 3) ]
      Q(c);
      ……
    end {S}
begin
    a:=0;                        [a(0, 3) ]
    S;
    ……
end. {P}
```

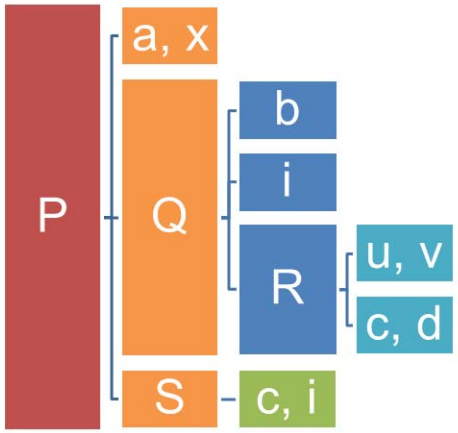





```
program P;
    var a, x : integer;
    procedure Q(b: integer);
      var i: integer;
      procedure R(u: integer; var v: integer);
        var c, d: integer;
      begin
        if u=1 then R(u+1, v)
        ……
        v:=(a+c)*(b-d);          [a(0, 3)、b(1, 4)]
        ……
      end {R}
    begin
      ……
      R(1, x);                   [x(0, 4)]
      ……
    end {Q}
    procedure S;
      var c, i:integer;
    begin
      a:=1;                      [a(0, 3) ]
      Q(c);
      ……
    end {S}
begin
    a:=0;                        [a(0, 3) ]
    S;
    ……
end. {P}
```




```
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);        [a(0, 3)、b(1, 4)]
        ……
      end {R}
    begin
      ……
      R(1, x);                 [x(0, 4)]
      ……
    end {Q}
    procedure S;
      var c, i:integer;
    begin                      [a(0, 3) ]
      a:=1;
      Q(c);
      ……
    end {S}
begin                          [a(0, 3) ]
    a:=0;
    S;
    ……
end. {P}
```



过程调用和过程返回

- 过程调用的语句

$$P(T_1, T_2, \ldots, T_n)$$

- 翻译成四元式序列

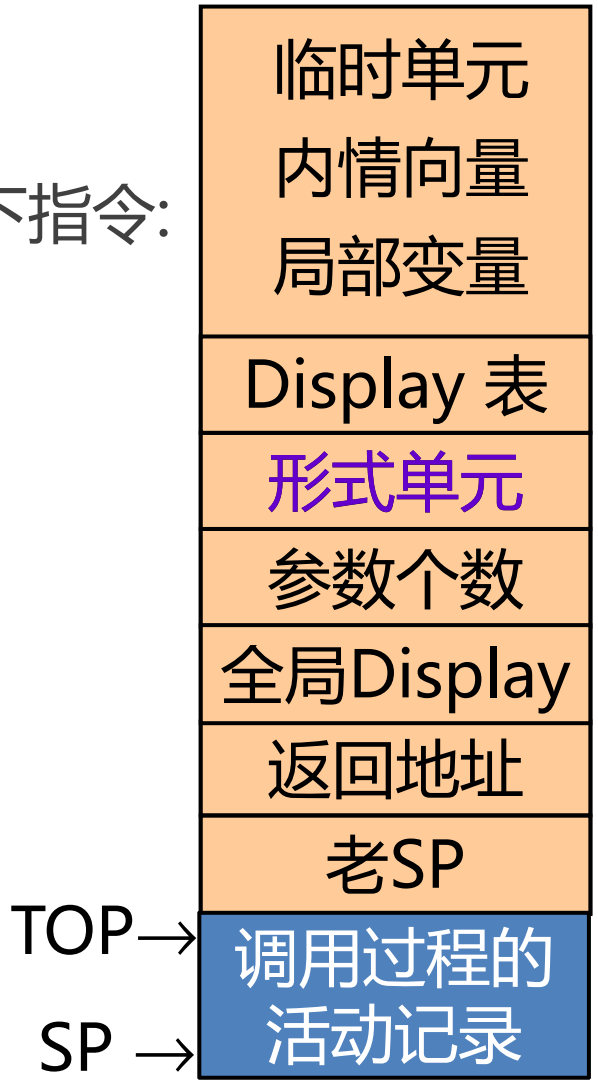
par T_1
par T_2
...
par T_n
call P, n

过程调用和过程返回

- 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)



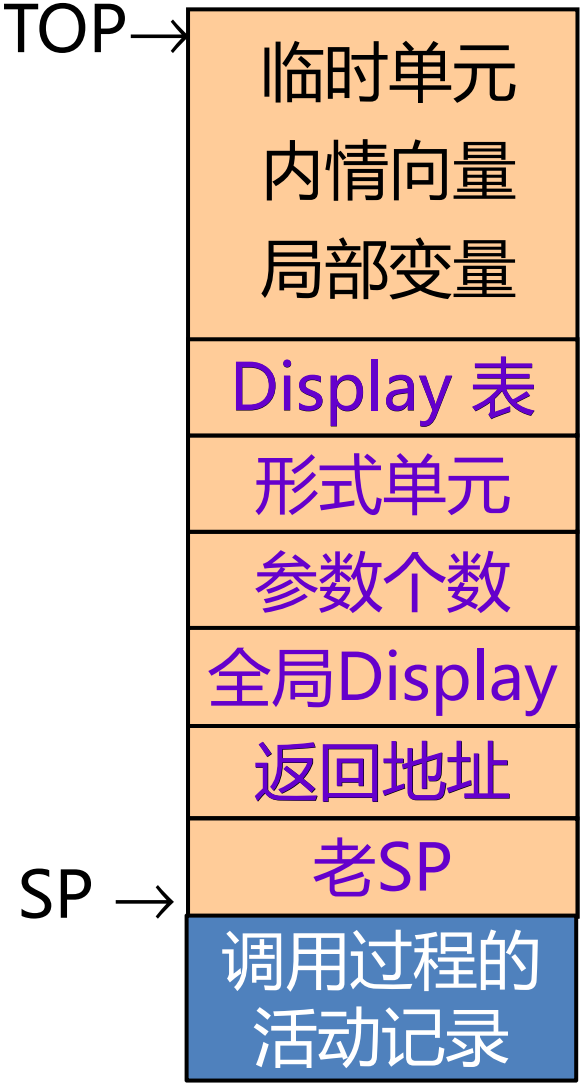
过程调用和过程返回

- 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)

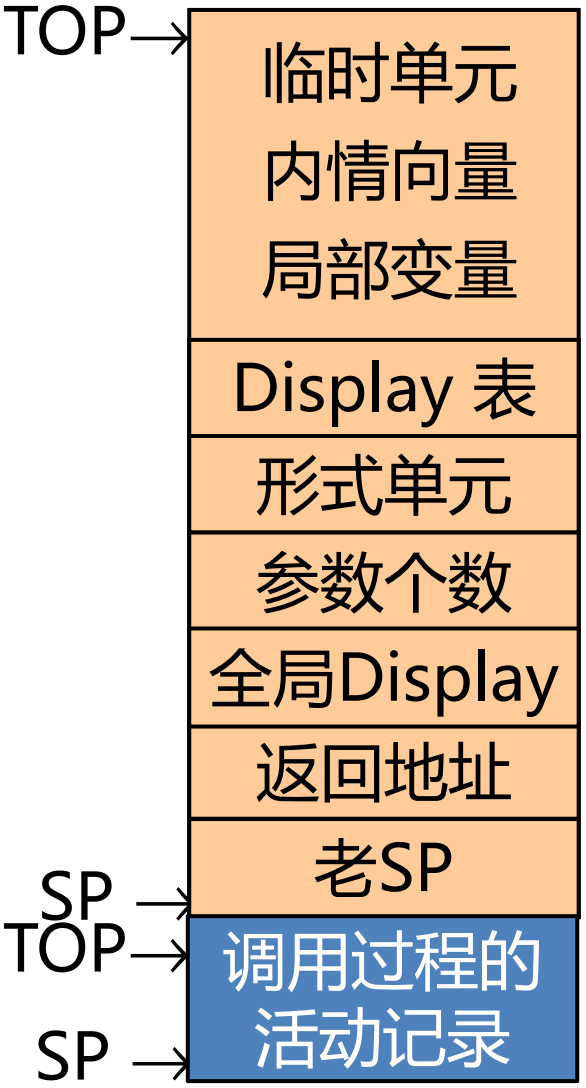
过程调用和过程返回

- 转进过程P后，首先定义新的SP和TOP，保存返回地址。
- 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/个单元，在添上进入P后新建立的SP值就构成了P的Display。



过程调用和过程返回

- 过程返回时，执行下述指令:
  TOP:=SP-1
  SP:=0[SP]
  X:=2[TOP]
  UJ 0[X]



小结

- 非局部名字的访问的实现
  - Display表和活动记录
  - 根据调用过程的信息建立被调用过程的Display表
- 过程调用和过程返回

小结

- 嵌套过程语言的动态存储管理
  - 嵌套过程语言的特点
  - 非局部名字的访问的实现：静态链、Display表

小结

- 运行时存储器的划分
  - 活动记录
- 非嵌套过程语言的动态存储管理
  - 栈式管理与活动记录
  - 过程调用和过程返回
- 嵌套过程语言的动态存储管理
  - 非局部名字的访问的实现：静态链、Display表
  - 过程调用和过程返回