CHƯƠNG 7
QUẢN LÝ BỘ NHỚ TRONG THỜI GIAN THỰC THI

7.1. Các phần tử yêu cầu cấp phát bộ nhớ trong thời gian thực thi

Tất cả các phần tử cần được cấp phát bộ nhớ, bao gồm:

1. Đoạn mã của chương trình được biên dịch.
2. Các chương trình hệ thống cần thiết trong thời gian thực thi.
3. Cấu trúc dữ liệu và hằng do người sử dụng định nghĩa.
4. Các điểm trở về của chương trình con.
5. Môi trường tham khảo.
6. Các vị trí nhớ tạm cho việc tính trị biểu thức.
7. Nhập, xuất bộ đệm.
8. Các bảng, trạng thái thông tin.

Ngoài dữ liệu và các chương trình được biên dịch, các tác vụ cũng cần bộ nhớ:

1) Gọi chương trình con và các tác vụ trở về.
2) Khởi tạo và hủy bỏ cấu trúc dữ liệu.
3) Tác vụ thêm vào hoặc loại bỏ các phần tử.

7.2. Các vấn đề về ngôn ngữ nguồn

Chương trình con

Mô phỏng 7.1. *Chương trình Pascal đọc và sắp xếp thứ tự các số nguyên*

```
(1)  program sort (input, output);
(2)          var a: array [0...10];
(3)      procedure readarray;
(4)          var i: integer;
(5)          begin
(6)              for i := 1 to 9 do read (a [1]);
(7)          end;
(8)      function partition (y, z: integer): integer;
(9)          var i, j, x, v: integer;
(10)         begin ...
(11)         end;
(12)     procedure quicksort (m, n: integer);
```

```
(13)                    var i: integer;
(14)                    begin
(15)                        if (n > m) then begin
(16)                            i := partition (m, n);
(17)                            quicksort (m, i – 1);
(18)                            quicksort (i + 1, n);
(19)                    end;
(20)            end;
(21)    begin
(22)            a[0] := -9999; a[10] := 9999;
(23)            readarray;
(24)            quicksort (1, 9);
(25)    end
```

Cây hoạt động *(activation tree)*

Cây hoạt động dùng để miêu tả con đường mà sự điều khiển đi vào và đi ra khỏi các hoạt động của chương trình. Một số tính chất của cây hoạt động:

1. Mỗi nút của cây tượng trưng cho một hoạt động của chương trình con.
2. Nút gốc (root) tượng trưng cho hoạt động của chương trình chính.
3. Nút a là cha của nút b nếu và chỉ nếu dòng điều khiển đi từ sự hoạt động a sang sự hoạt động b.
4. Nút a ở bên trái nút b nếu và chỉ nếu thời gian sống của a xuất hiện trước thời gian sống của b.

Mô phỏng 7.2. *Các phát biểu in của chương trình ở mô phỏng 7.1 miêu tả sự thực thi của nó.*

```
Sự thực thi chương trình bắt đầu
vào          readarray
ra khỏi      readarra
vào          quicksort (1,9)
vào          partition (1,9)
ra khỏi partition (1,9)
vào          quicksort (1,3)
.....
ra khỏi      quicksort (1,3)
vào          quicksort (5,9)
.....
ra khỏi      quicksort (5,9)
ra khỏi      quicksort (1,9)
Sự thực thi kết thúc
```

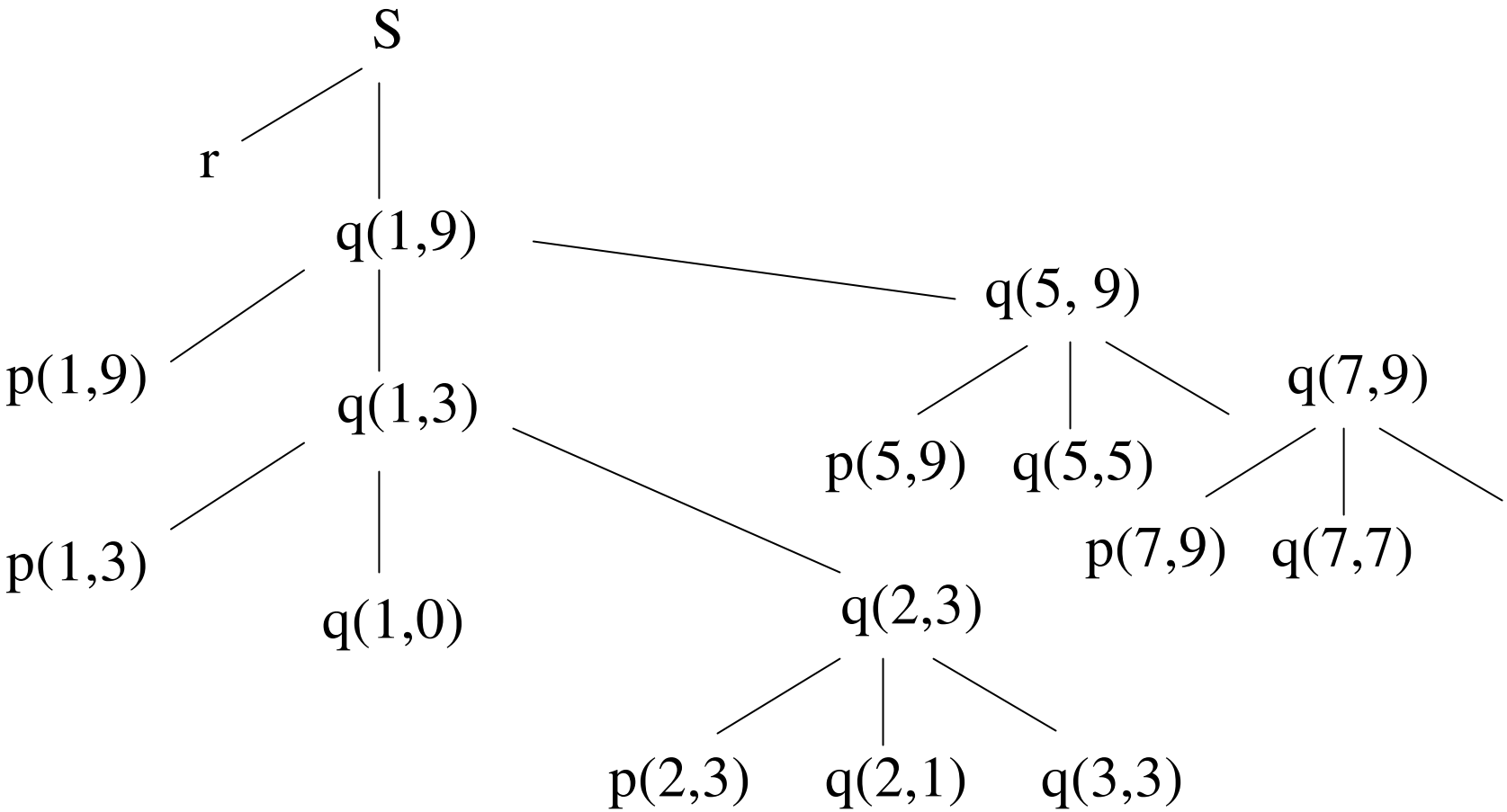
Thí dụ 6.1.

s: viết tắt cho sort

r: viết tắt cho readarray

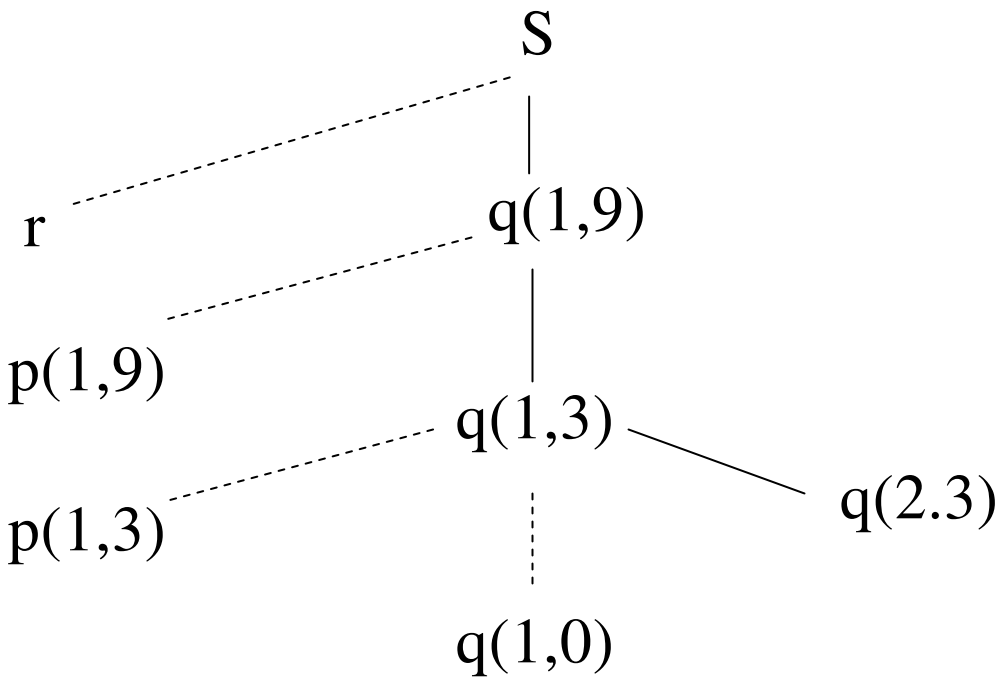
p: viết tắt cho partition

q: viết tắt cho quicksort



Hình 7.1. Cây hoạt động được xây dựng từ chuỗi xuất ở mô phỏng 7.2.

Stack điều khiển *(Control stack)*



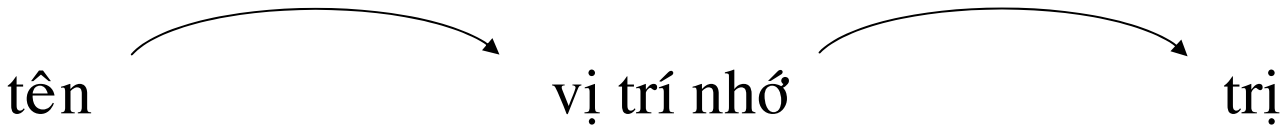
Hình 7.2. Stack điều khiển bao gồm các nút trên con đường từ s đến q (2,3) và trở về

Tầm vực của sự khai báo

Khai báo có thể tường minh, Var I: integer nhưng có thể là khai báo ngầm như Fortran, khi ta dùng tên biến i mà không khai báo, Fortran mặc nhiên hiểu i là biến nguyên. Tầm ảnh hưởng của các khai báo được quy tắc tầm vực quyết định.

Sự ràng buộc của tên

Môi trường là tên của hàm, ánh xạ tên đến vị trí nhớ và trạng thái là hàm ánh xạ từ vị trí nhớ đến trị mà nó lưu giữ.



Hình 7.3. Phép chiếu hai mức từ tên đến trị

Sự ràng buộc chính là bản sao động của khai báo, trong thời gian thực thi.

Bảng 7.1. *Các khái niệm tĩnh và động của chương trình con*

Khái niệm tĩnh	Bản sao động
Định nghĩa chương trình con	Sự hoạt động của chương trình con
Khai báo tên	Sự ràng buộc tên với vị trí nhớ
Tầm vực ý nghĩa của khai báo	Thời gian sống của sự ràng buộc tên

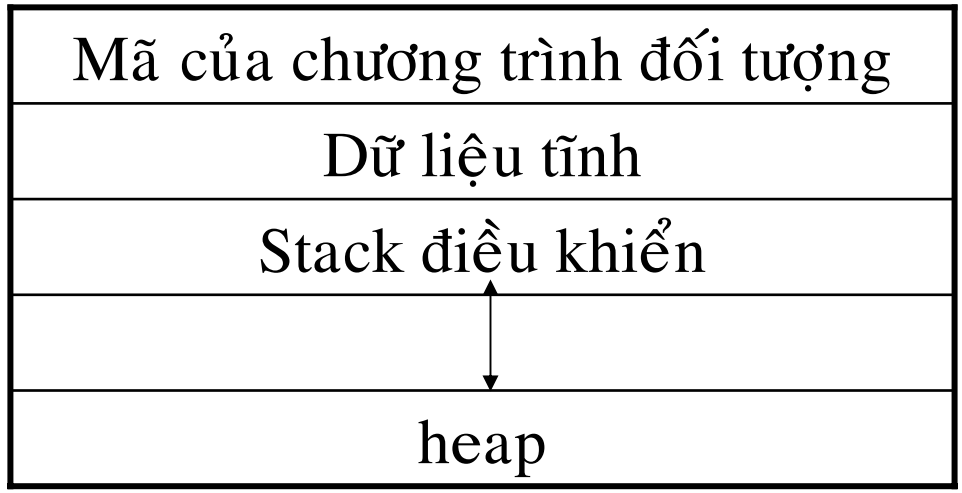
7.3. Tổ chức ký ức

Sự phân chia bộ nhớ trong thời gian thực thi

Trong thời gian dịch, trình biên dịch đã tính toán kích thước bộ nhớ dành cho chương trình đối tượng, nó bao gồm:

1. Mã của chương trình đối tượng.
2. Các đối tượng dữ liệu.
3. Một phần trong stack điều khiển (stack trung tâm) lưu giữ bản ghi hoạt động của chương trình con.

Mô phỏng 7.2. *Sự phân chia bộ nhớ trong thời gian thực thi cho vùng mã của chương trình và vùng dữ liệu.*



Không phải tất cả các ngôn ngữ lập trình đều dùng stack điều khiển và heap, nhưng Pascal và C thì dùng cả hai.

Bản ghi hoạt động *(Activation record)*

1. *Vùng giá trị khứ hồi*
2. *Vùng thông số*
3. *Đường liên kết động*
4. *Đường liên kết tĩnh*
5. *Các trạng thái máy*
6. *Vùng dữ liệu cục bộ*
7. *Vùng nhớ tạm*

Mô phỏng 7.3. *Dạng tổng quát của bản ghi hoạt động*

Giá trị khứ hồi
Thông số thực
Đường liên kết động
Đường liên kết tĩnh
Các trạng thái máy
Dữ liệu cục bộ
Vùng nhớ tạm

7.4. Chiến thuật cấp phát bộ nhớ

1. Cấp phát tĩnh
2. Quản trị bộ nhớ theo cơ chế stack
3. Cơ chế heap

1. Cấp phát tĩnh *(Static allocation)*

Cơ chế này sẽ dẫn đến một số hạn chế sau đây:

1) Kích thước và vị trí của đối tượng dữ liệu phải được xác định ngay trong thời gian biên dịch.

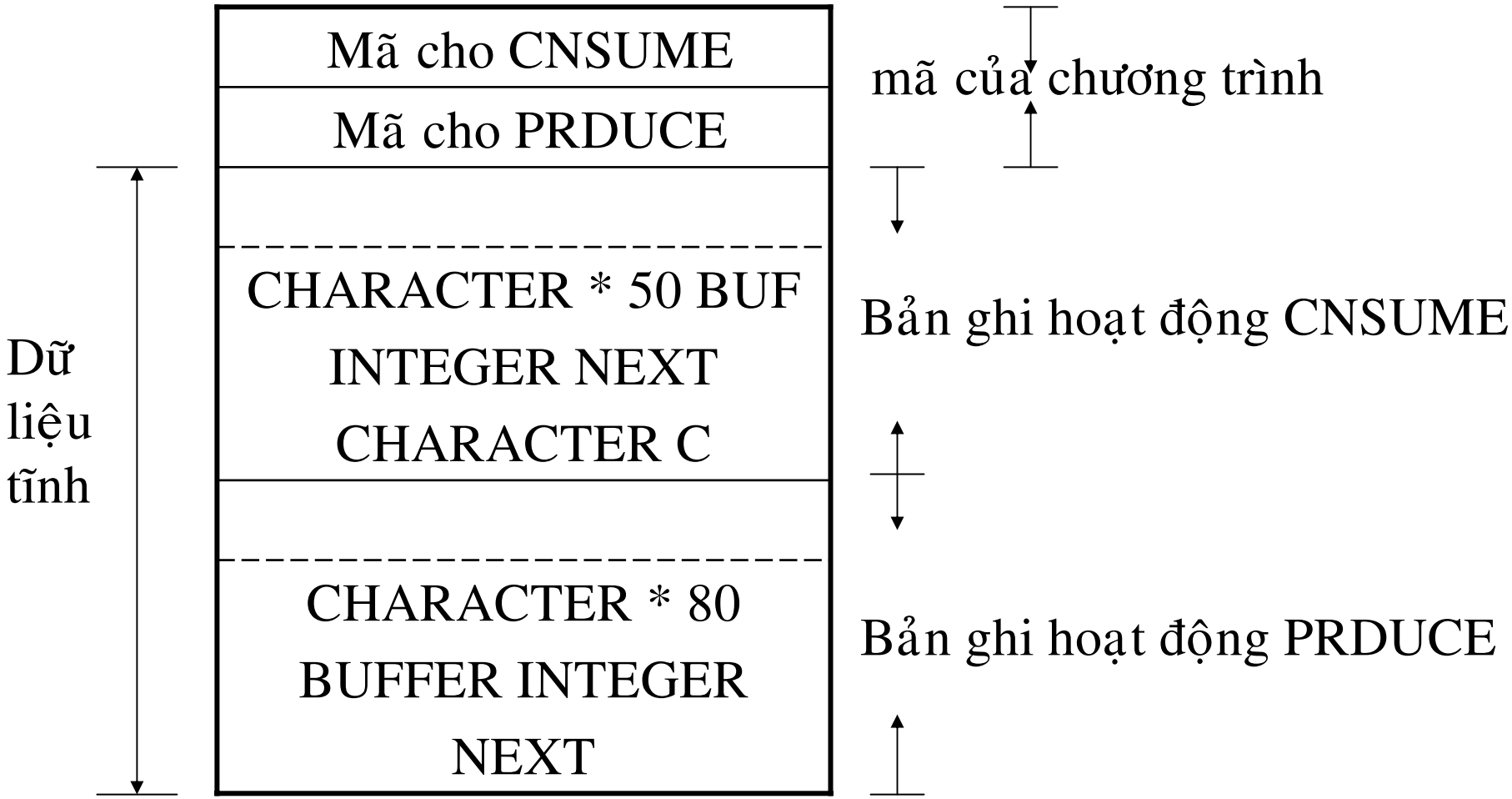
2) Không cho phép gọi đệ quy.

3) Không cho phép cấp phát động các đối tượng dữ liệu.

Mô phỏng 7.4. *Chương trình trong ngôn ngữ Fortran.*

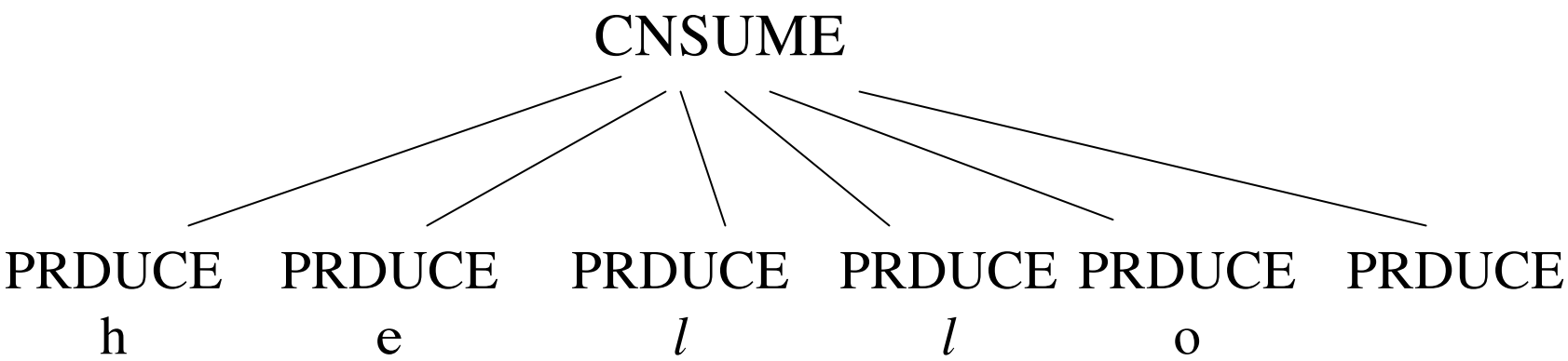
(1)	PROGRAM CNSUME
(2)	CHARACTER * 50 BUF
(3)	INTEGER NEXT
(4)	CHARACTER C, PRDUCE
(5)	DATA NEXT /1/, BUF /’’/
(6)	C = PRDUCE ()
(7)	BUF (NEXT: NEXT) = C
(8)	NEXT = NEXT + 1
(9)	IF (C. NE. ‘’) GOTO 6

```
(10)    WRITE (*, ‘(A)’) BUF
(11)    END
(12)    CHARACTER FUNCTION PRDUCE ()
(13)    CHARACTER * 80 BUFFER
(14)    INTEGER NEXT
(15)    SAVE BUFFER, NEXT
(16)    DATA NEXT /81/
(17)    IF (NEXT. GT. 80) THEN
(18)    READ (*, ‘’(A)’’) BUFFER
(19)    NEXT = 1
(20)    END IF
(21)    PRDUCE = BUFFER (NEXT: NEXT)
(22)    NEXT = NEXT + 1
(23)    END
```



Hình 7.4. Vị trí nhớ tĩnh cho các biến cục bộ cho chương trình Fortran 77

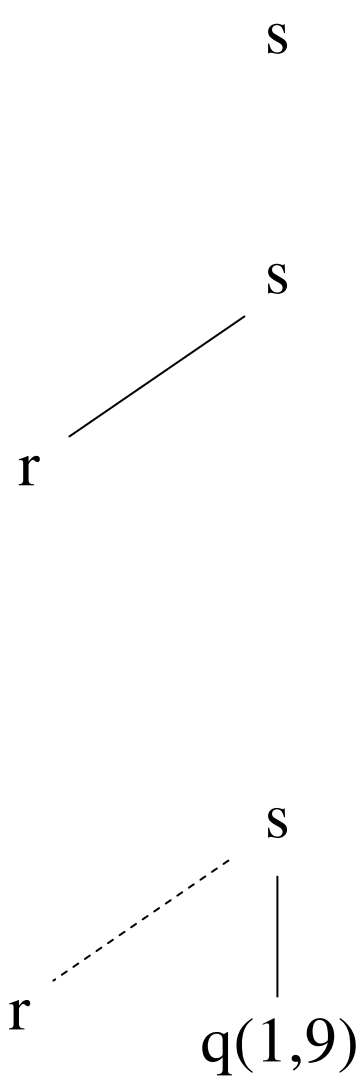
Thí dụ 7.2. Chương trình ở (mô phỏng 7.4) sẽ làm việc với các giá trị cục bộ được lưu lại qua các lần hoạt động. Các kýhiệu xuất ra trong chương trình chính CNSUME, được lấy từ bản ghi hoạt động của PRDUCE là hello, do CNSUME gọi PRDUCE 6 lần, như ở (H.7.5).



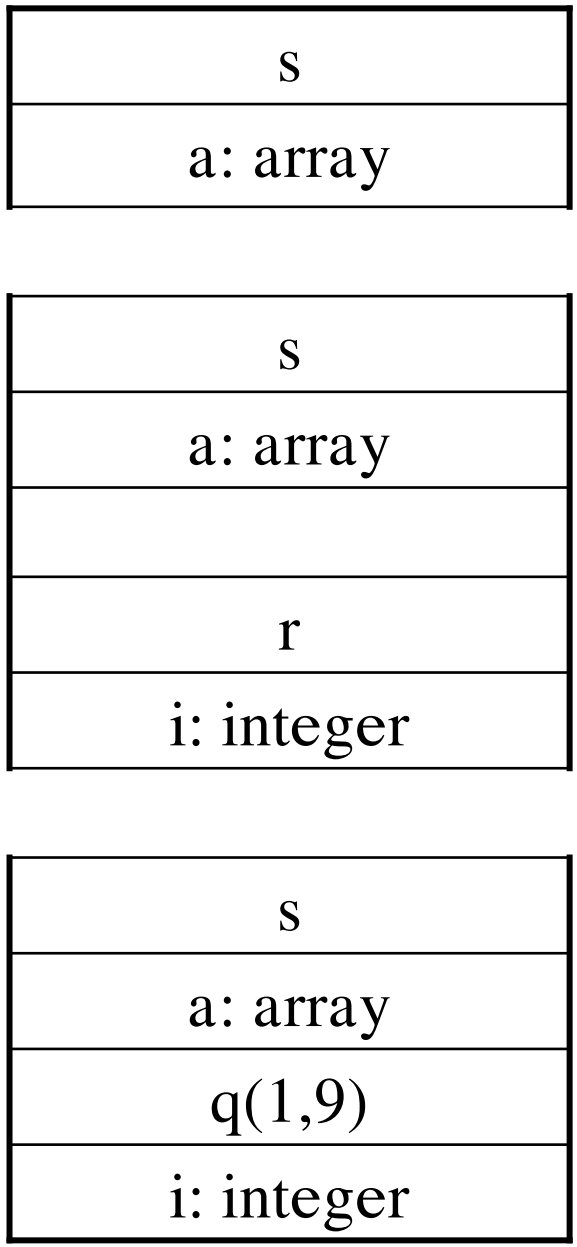
Hình 7.5. Các ký hiệu được trả về qua các lần hoạt động của PRDUCE

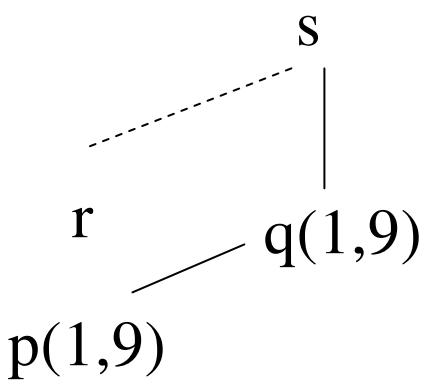
2. Cấp phát theo cơ chế stack

cây hoạt động

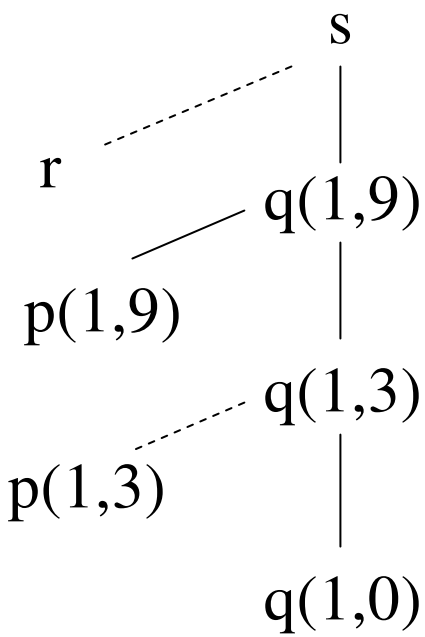


stack điều khiển

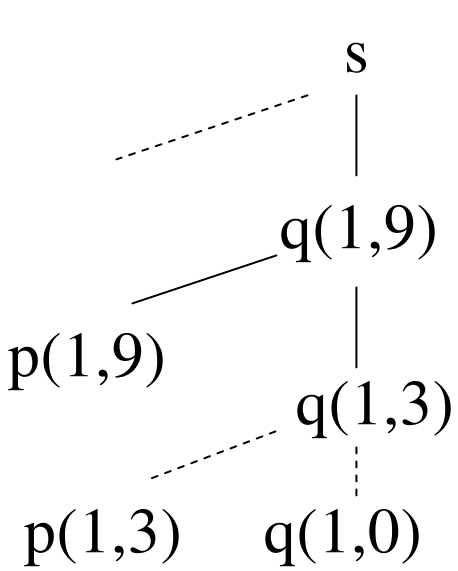




s
a: array
q(1,9)
i: integer
p(1,9)
i, j, x, v: integer



s
a: array
q(1,9)
i: integer
q(1,3)
i: integer
q(1,0)
i: integer



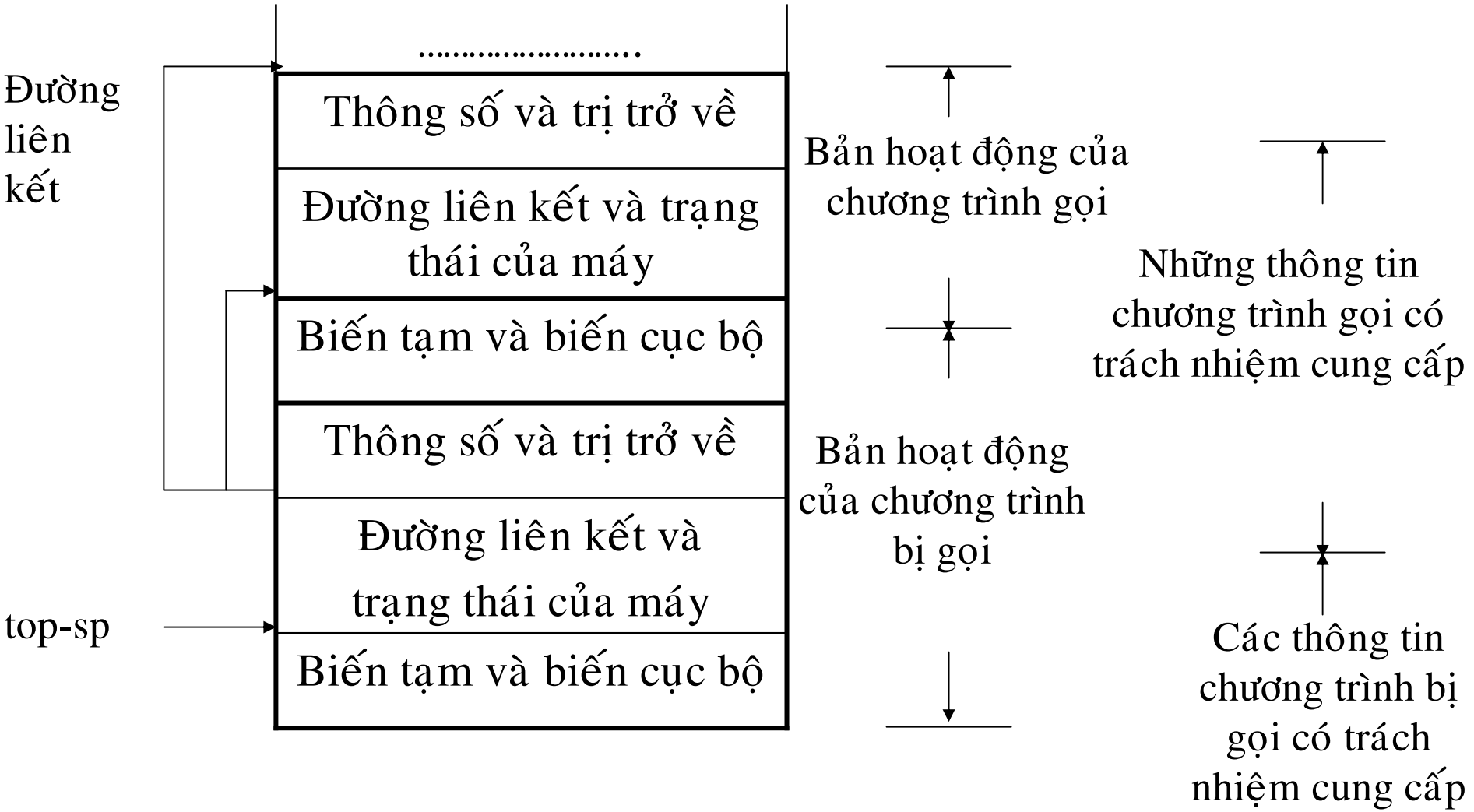
s
a: array
q(1,9)
i: integer
q(1,3)
i: integer

Hình 7.6. Các bản ghi hoạt động được cấp phát và loại bỏ khỏi stack điều khiển.

Sự gọi chương trình con

1. Chương trình gọi tính toán các thông số thực và cất vào vùng thông số của bản ghi hoạt động của chương trình bị gọi.
2. Chương trình bị gọi lưu giữ địa chỉ khứ hồi vào vùng trị trả về và trị top-sp vào vùng liên kết, tăng top-sp lên một khoảng vị trí nhớ, chính là kích thước của vùng biến tạm và biến cục bộ của nó với kích thước vùng thông số, trị trở về, đường liên kết và các trạng thái máy của chương trình bị gọi.

3. Chương trình bị gọi sẽ lưu bộ nhớ giá trị, các thanh ghi, đường liên kết và trạng thái khác.
4. Chương trình bị gọi khởi động các giá trị cục bộ của nó và bắt đầu thực thị.



Hình 7.7. Sự phân chia công việc giữa chương trình gọi và bị gọi

Chuỗi trở về có thể là các công việc sau

1. Chương trình bị gọi gửi các giá trị trở về vào bản ghi hoạt động của chương trình con.
2. Chương trình bị gọi xác lập lại trị: top-sp cho chương trình gọi, trị các thanh ghi, địa chỉ khứ hồi.
3. Chương trình gọi sẽ sử dụng các giá trị trong vùng biến tạm, giá trị trở về để tính toán các biểu thức sau này khi nó thực thi tiếp tục.

Dữ liệu có kích thước thay đổi

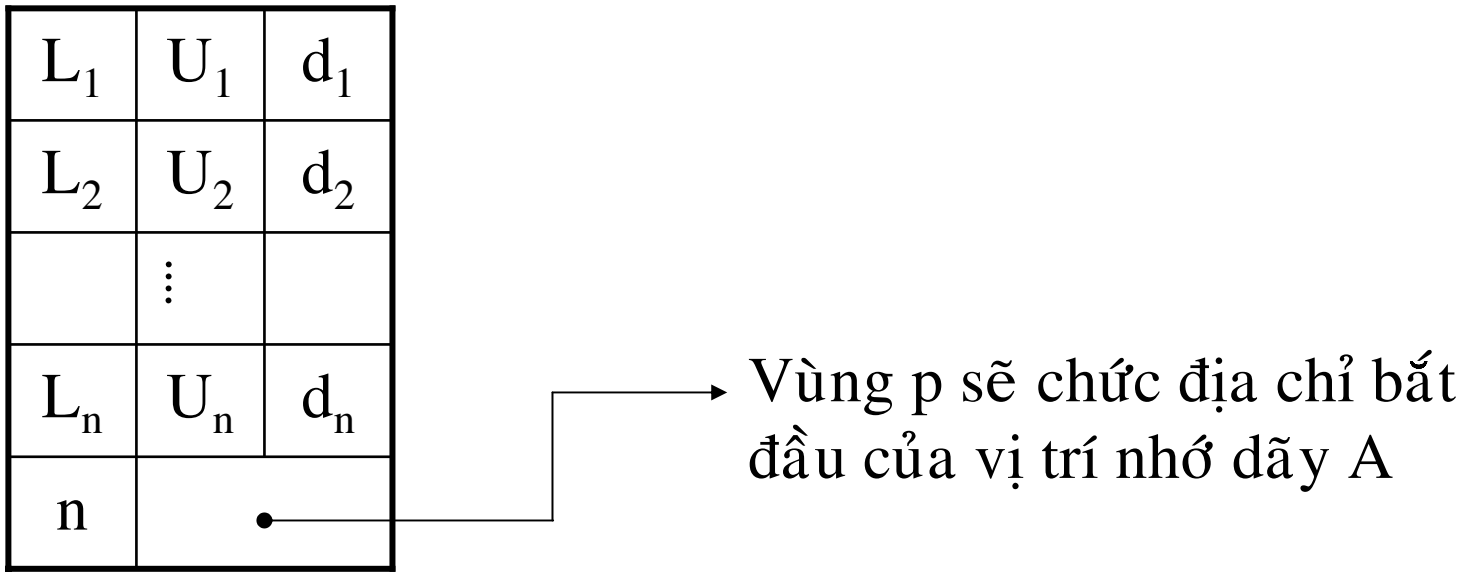
Ở một số ngôn ngữ như C, Algol, dãy được phép có kích thước thay đổi trong thời gian thực thi.

Thí dụ 7.4. Cho khai báo dãy trong Algol như sau:

DIMENSION A [L1 : U1, L2: U2, ...Ln: Un]

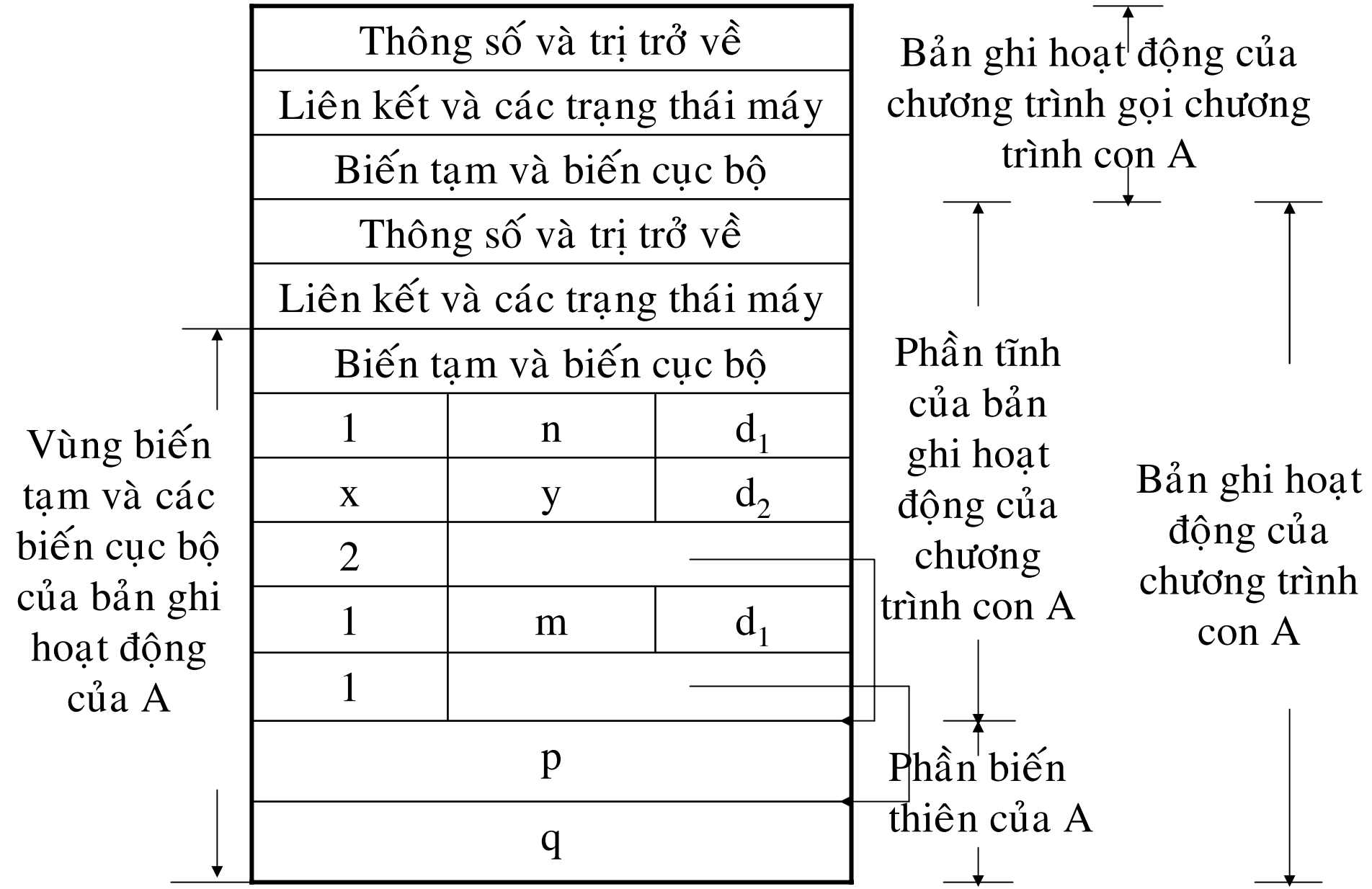
di là kích thước chiều thứ I, được tính: di = Ui – Li + 1

Vùng thông tin cho dãy A là:



Hình 7.8. Vùng thông tin của dãy trong bản ghi hoạt động

Thí dụ 7.5. DIMENSION p[1: n, x: y], q[1: m];

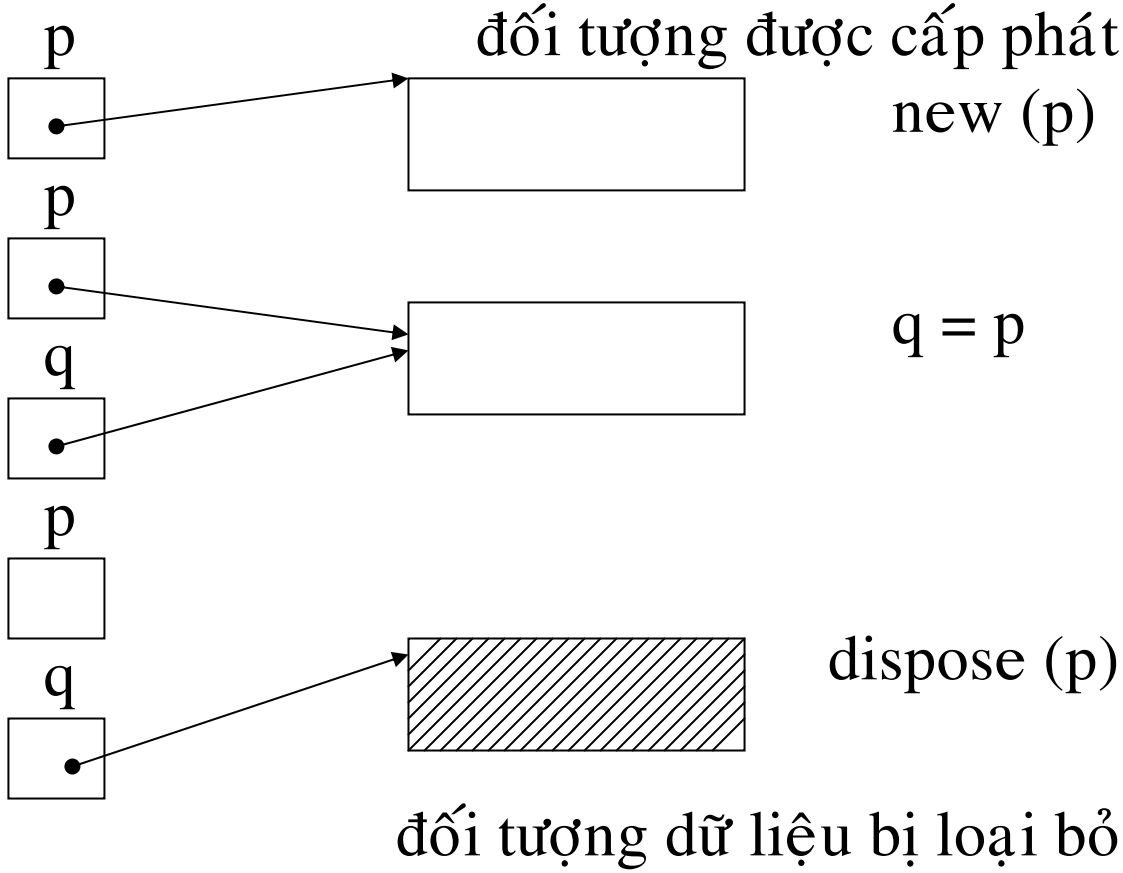


Hình 7.9. Bản ghi hoạt động của chương trình con A, có các biến dãy p, q với kích thước thay đổi

Tham chiếu treo *(Dangling reference)*

Mô phỏng 7.5. *Đoạn chương trình Pascal gây ra tham chiếu treo.*

```
var p, q: ^ integer;
begin...
        new(p);
        q: = p;
        dispose (p)
        ...
end;
```

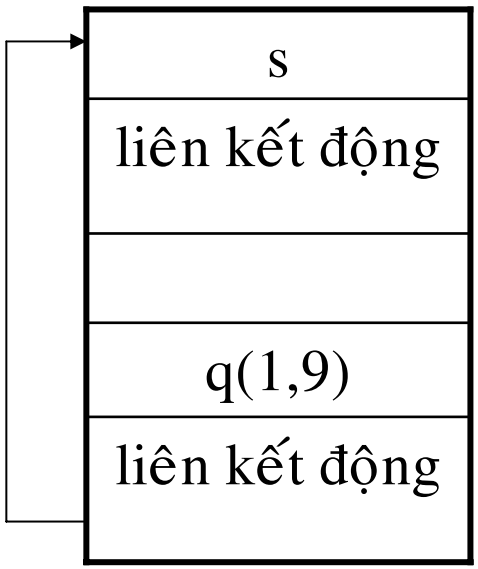
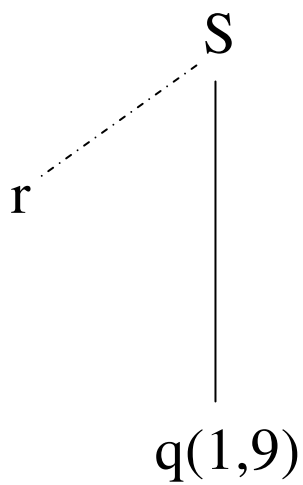
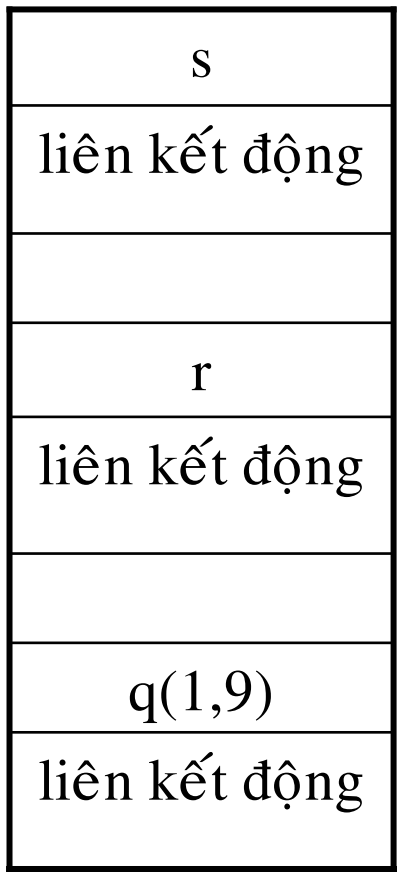


Hình 7.10. Tham chiếu treo q xuất hiện do lệnh dispose (p).

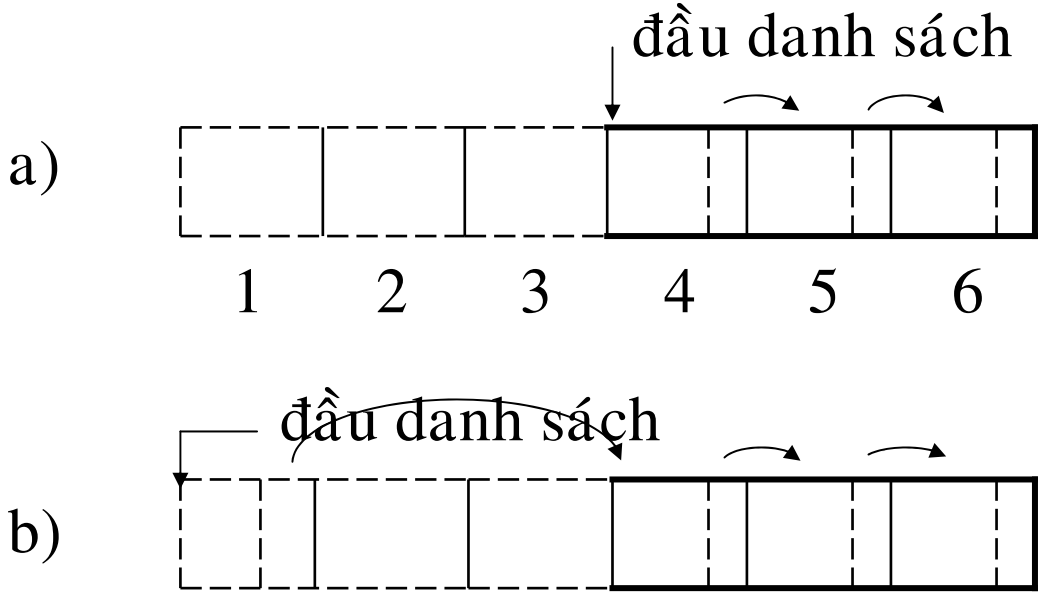
3. Cấp phát theo cơ chế heap

1. Trị của các biến cục bộ được lưu giữ ngay cả khi sự hoạt động của chương trình con tương ứng không còn nữa.
2. Sự hoạt động của chương trình bị gọi sống sau cả chương trình gọi.

Bảng 7.2. *Các bản ghi hoạt động của heap và stack cùng sự so sánh với cây hoạt động.*

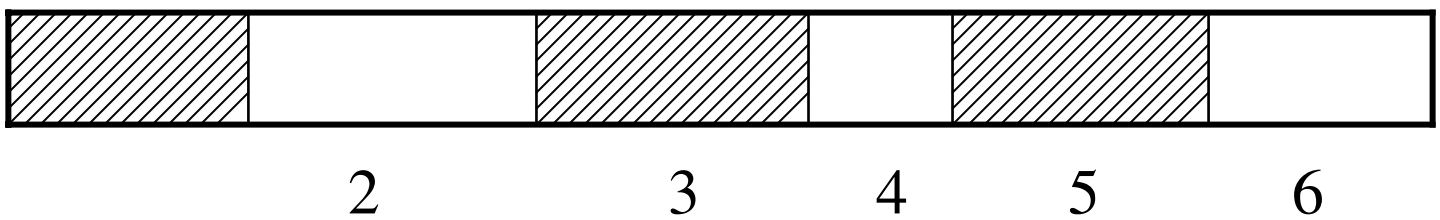
Các bản ghi hoạt động trên stack	Vị trí trên cây hoạt động	Các bản ghi hoạt động trong heap	Ghi chú
			Theo cơ chế heap r đã hết thực thi nhưng bản ghi hoạt động của nó vẫn còn tồn tại

Cấp phát vị trí nhớ cho các khối có kích thước cố định



Hình 7.11. Các khối bị loại bỏ sẽ được thêm vào danh sách của các khối chưa sử dụng.

Cấp phát vị trí nhớ cho các khối có kích thước thay đổi



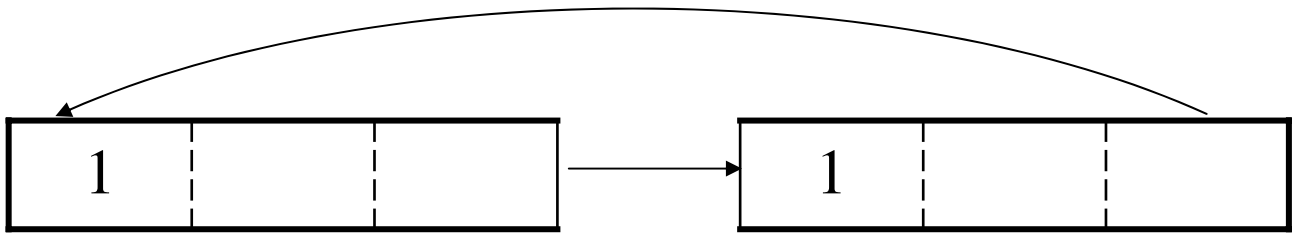
Hình 7.12. Các khối đang được sử dụng và đang trống

Loại bỏ ngầm vị trí nhớ

Mô phỏng 7.6. *Dạng của một khối*

kích thước khối
số lượng con trỏ tham khảo tới
đánh dấu
các con trỏ chỉ đến các khối
thông tin của người sử dụng

1. Đếm các tham khảo



*Hình 7.13. Hai khối này là **rác** mặc dù vẫn có số đếm tham khảo là 1*

2. Kỹ thuật đánh dấu

7.5. Truy xuất biến không cục bộ

Mô phỏng 7.7. *Chương trình dùng để minh họa việc truy xuất biến không cục bộ.*

```
program MAIN
    var x: integer;
        procedure sub1;
            var x: real;
            ⋮
        begin
            read (x)
            sub2;
        end;
        procedure sub2;
            (Không có khai báo x)
            ⋮
```

```
        begin
          ⋮
        write (x);
          ⋮
        end
begin {main}
          ⋮
        sub1;
end.
```

Cấu trúc khối

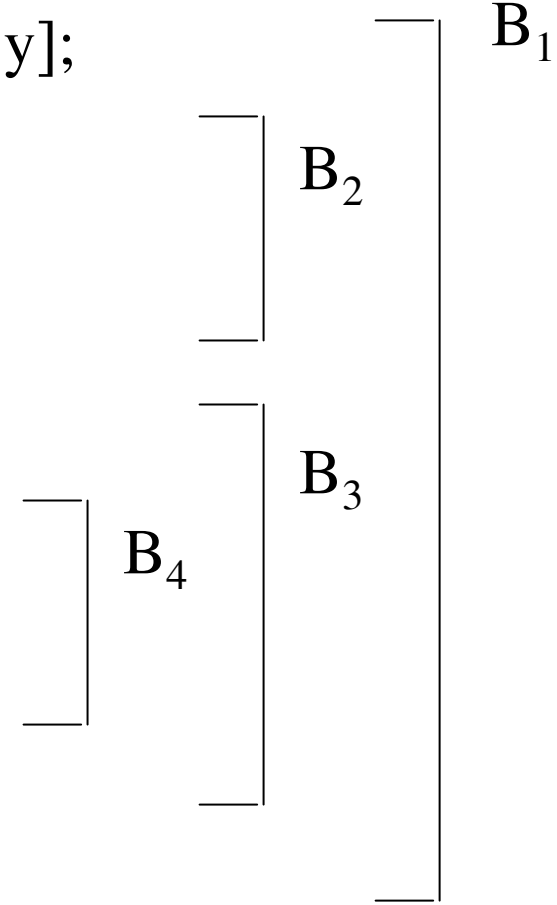
Quản trị bộ nhớ và việc cấp phát vị trí nhớ cho các khối của Algol

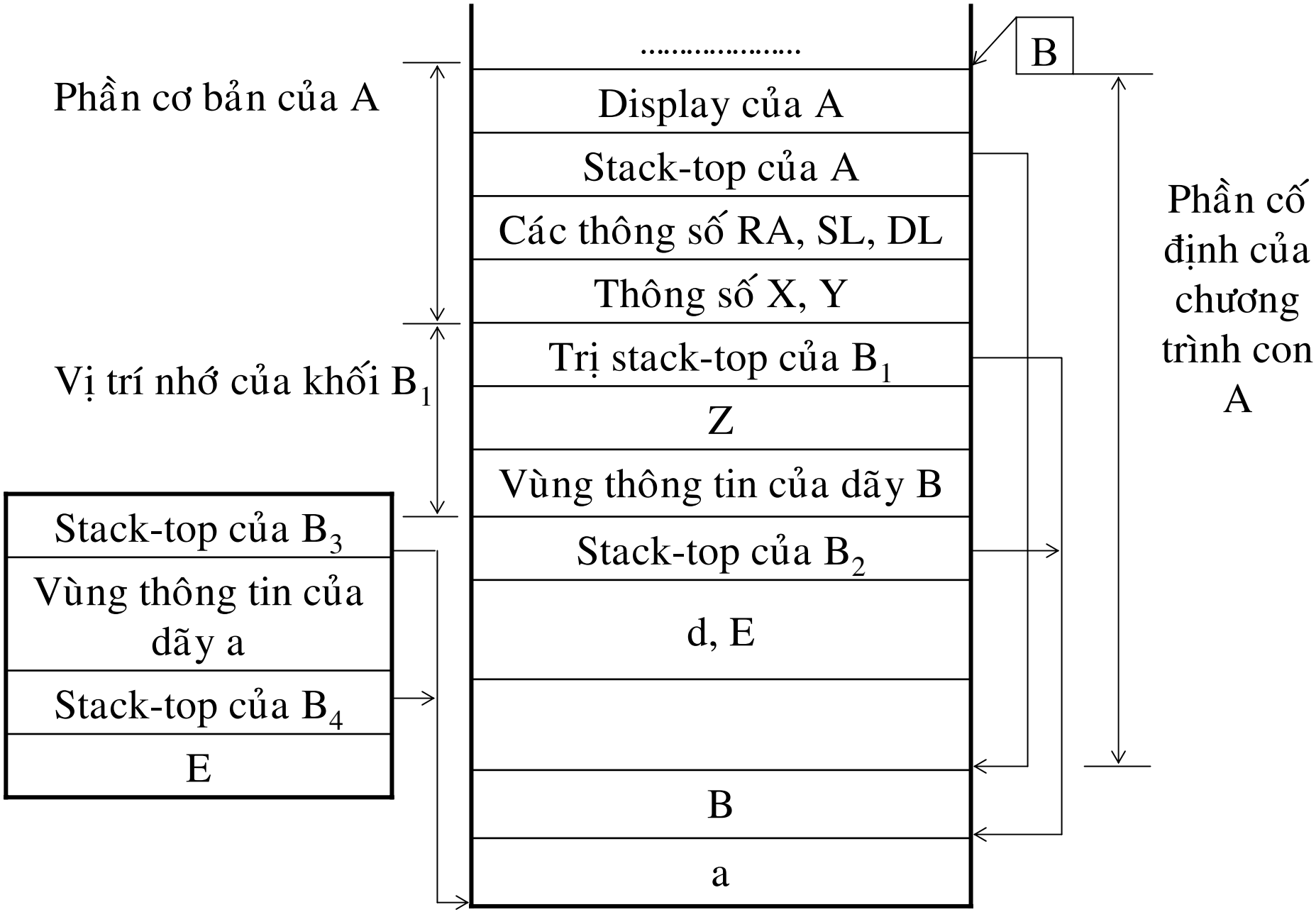
Mỗi bản ghi hoạt động gồm các thành phần chính sau đây:

1. Dãy display của chương trình con. Nếu chương trình ở cấp i thì display chiếm i + 1 ô nhớ.
2. Vị trí nhớ chứa trị stack top của chương trình con.
3. Các thông tin về địa chỉ khứ hồi, liên kết tĩnh và liên kết động, stack_top của chương trình con gọi.
4. Các vị trí nhớ dành cho các thông số của chương trình con. Các phần 1, 2, 3, 4 tạo thành phần cơ bản của chương trình con.
5. Mỗi chương trình con có thể có nhiều khối, mỗi khối được cấp phát một khoảng ký ức để chứa các thành phần sau:
   1) Vị trí nhớ chứa trị stack top của khối.
   2) Vị trí nhớ dành cho các biến cục bộ là biến đơn.
   3) Các thông tin về dãy (nếu khối có khai báo dãy).
   4) Vị trí nhớ chứa các biến tạm.

Thí dụ 7.9. Cho chương trình con trong Algol.

```
procedure A (x, y);
        integer x, y;
L1:         begin real z; array B [x: y];
L2:             begin real: D, E;
                .....
                .....
                end;
L3:         begin array a[a: x];
L4:             begin real E;
                ........
                ....
                end;
            end;
        end;
```





Hình 7.16. Bản ghi hoạt động của chương trình con A có chứa các khối

Các hành vi thâm nhập vào một khối và ra khỏi khối

- *Hành vi thâm nhập vào một khối*

- *Hành vi ra khỏi khối*

Tầm vực tĩnh với các chương trình con không lồng nhau

Tầm vực tĩnh với các chương trình con lồng nhau

Bảng tầm vực *(display)*

Để truy xuất biến không cục bộ, người ta sử dụng bảng tầm vực. Tuy nhiên, liên kết tĩnh vẫn tồn tại trong các bản ghi hoạt động, dùng để phục hồi hình ảnh bảng tầm vực khi chương trình con cấp i gọi chương trình con cấp j, với i > j và sau khi chương trình con cấp j hoàn tất sự thực thi.

Thí dụ 7.12. Cho chương trình sau:

Mô phỏng 7.10. *Chương trình Pascal có cấu trúc khối*

```
program M;
        :
        procedure P;
           :
           procedure Q;
                 begin
                 : P ;
                 end;
           procedure R;
                 begin
                 Q;
                 end;
           begin;
           R;
           end;
           begin
           P;
           end;
```

Mức tầm vực của các chương trình con là:

M

P

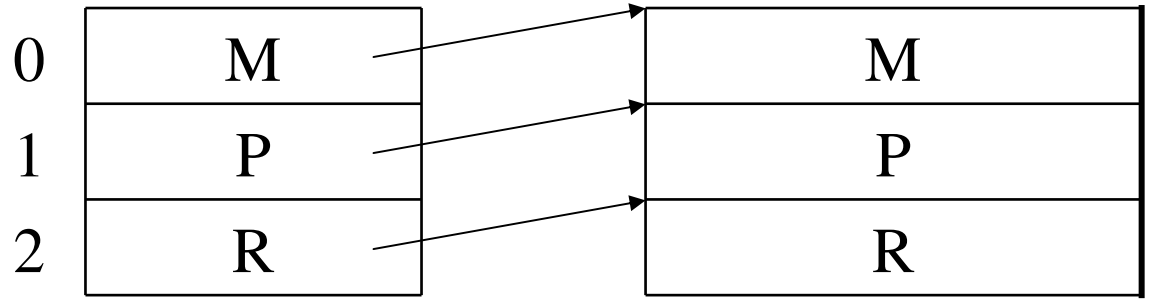
Q

R

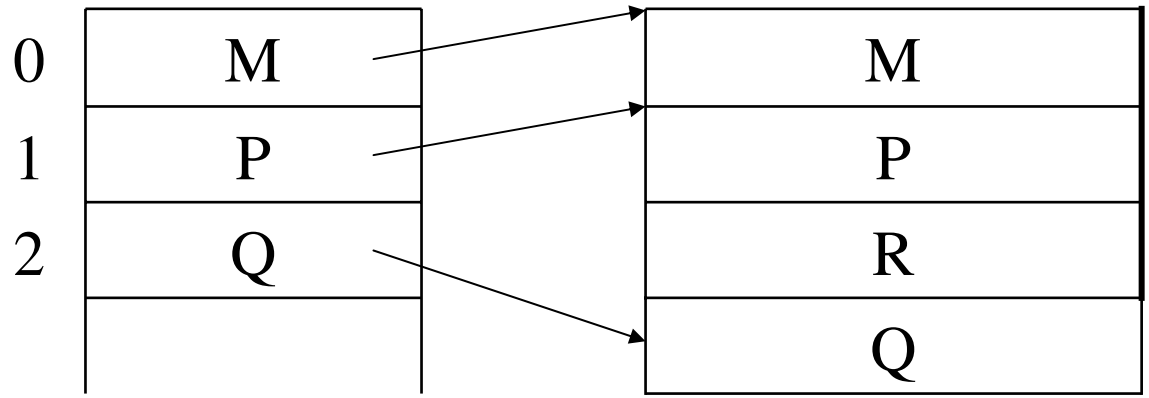
và M gọi P gọi R gọi Q gọi P

	Các bước thực thi		Display	Stack điều khiển
1	M	0	M	M
2	M gọi P	0	M	M
		1	P	P

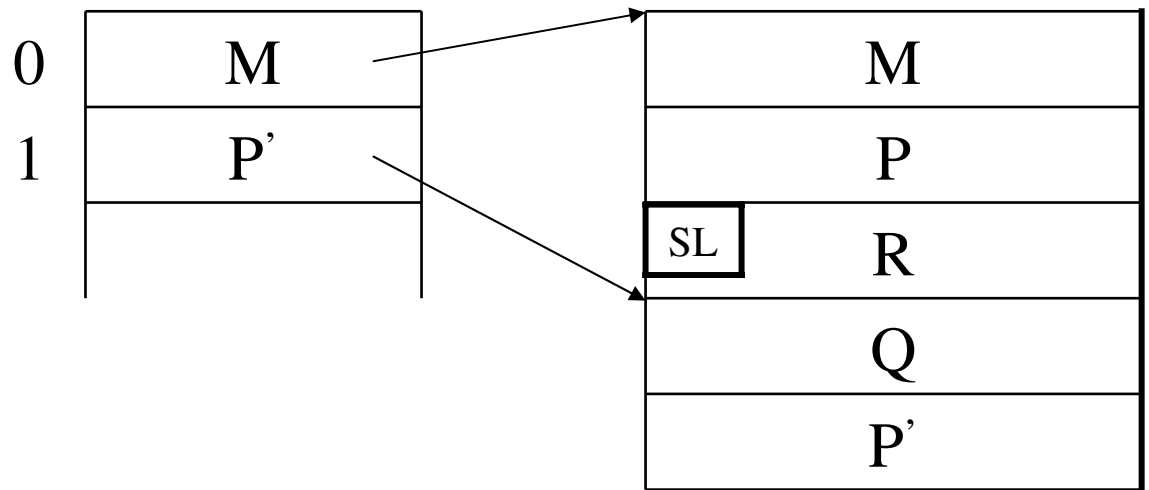
3 P gọi R

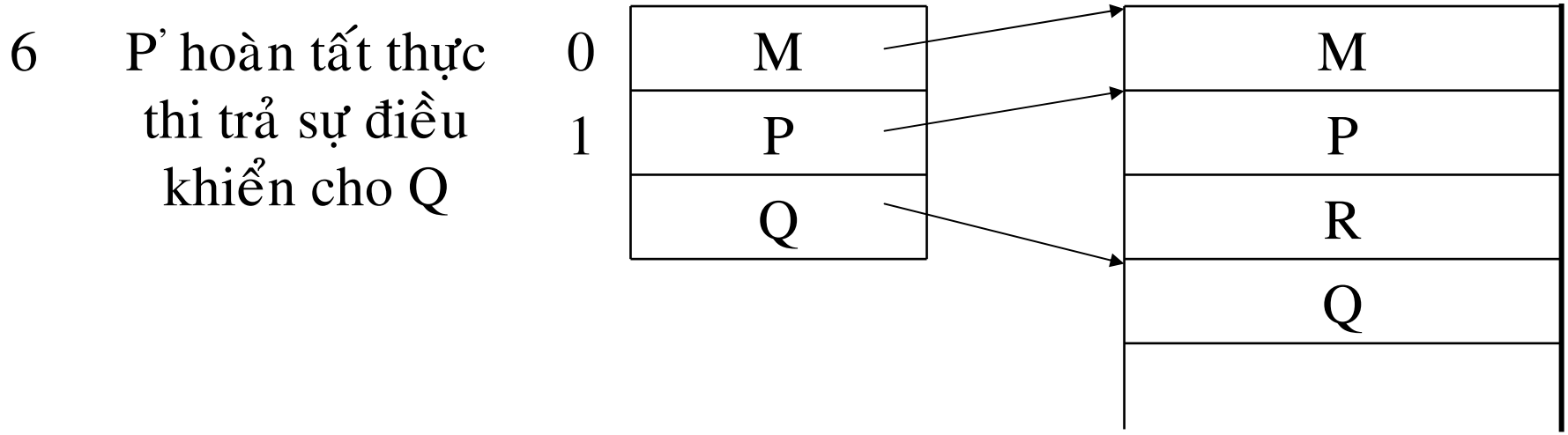


4 R gọi Q



5 Q gọi P





Hình 7.20. Các bước gọi chương trình con cùng với sự thay đổi nội dung của display và stack điều khiển

Tầm vực động

7.6. Truyền thông số

1. Thông số nhập – xuất

- Truyền bằng tham khảo
- Truyền bằng trị

2. ***Thông số chỉ nhập***

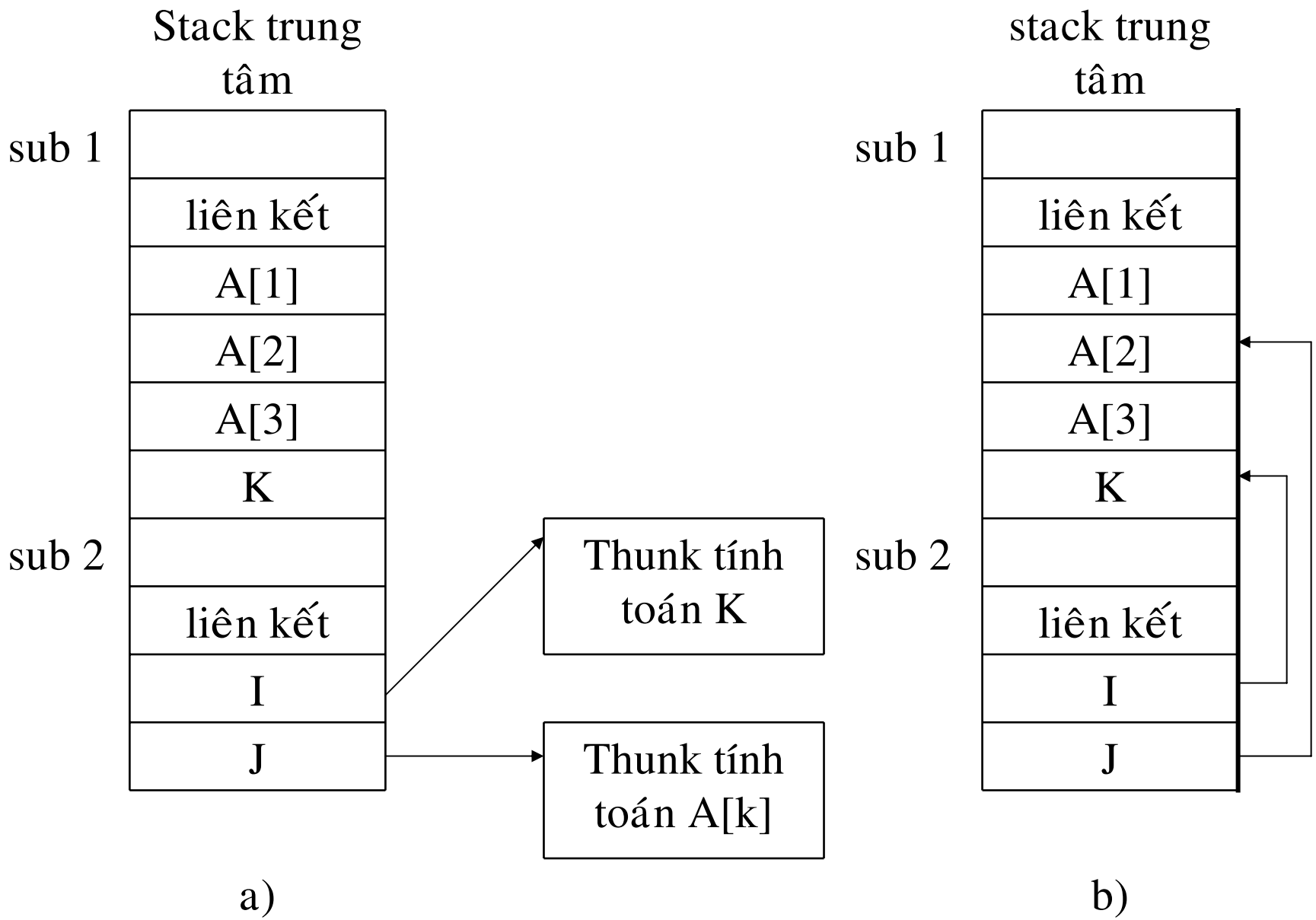
- Truyền bằng trị
- Truyền bằng trị hằng

3. ***Thông số chỉ xuất***

- Truyền thông số bằng tên

Thí dụ 7.6. Cho chương trình

```
type VECT = array [1 .. 3] of integer;
    procedure SUB2 (var I, J: integer);
        begin
            I := I + 1;
            J := J + 1; write (I, J);
        end;
procedure SUB1;
    var A: VECT;
        K :integer;
    begin A[1] := 7; A[2] := 8; A[3] := 9;
            K :=2: SUB2 (K, A[K]);
            for K := 1 to 3 do write (A [K]);
    end;
```



Hình 7.23. Phương pháp truyền thông số bằng tên và bằng tham khảo

Chương trình con đóng vai trò thông số

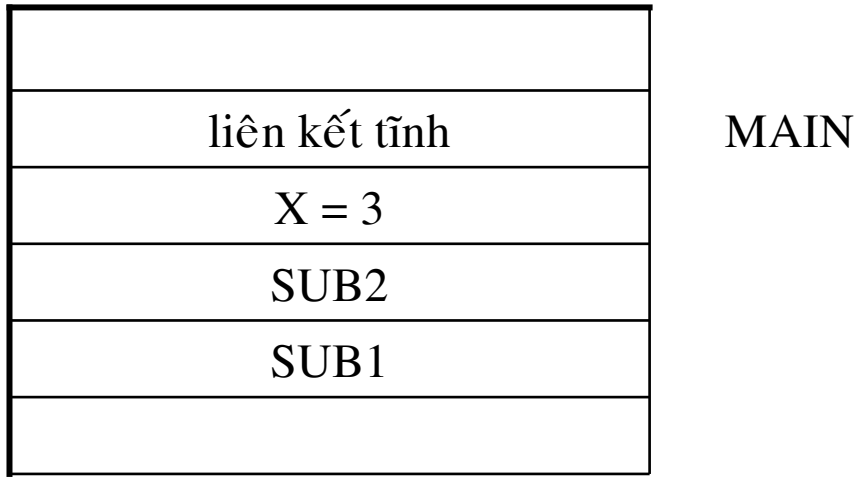
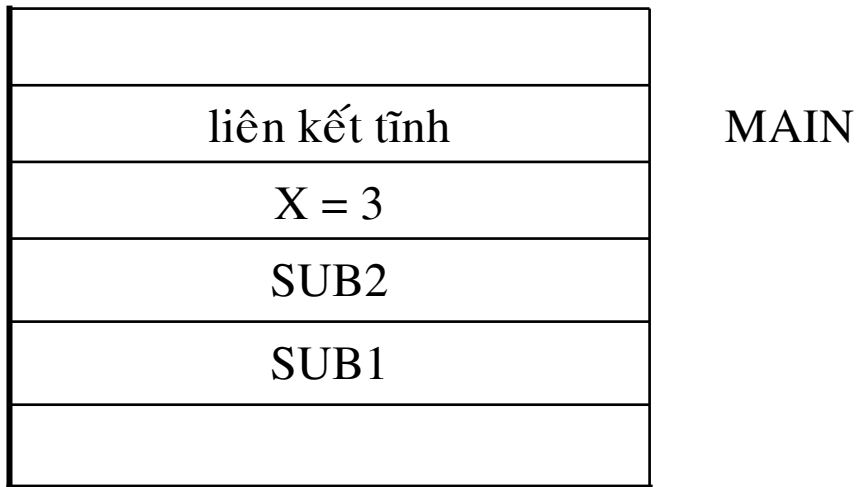
Thí dụ 7.7. Cho chương trình

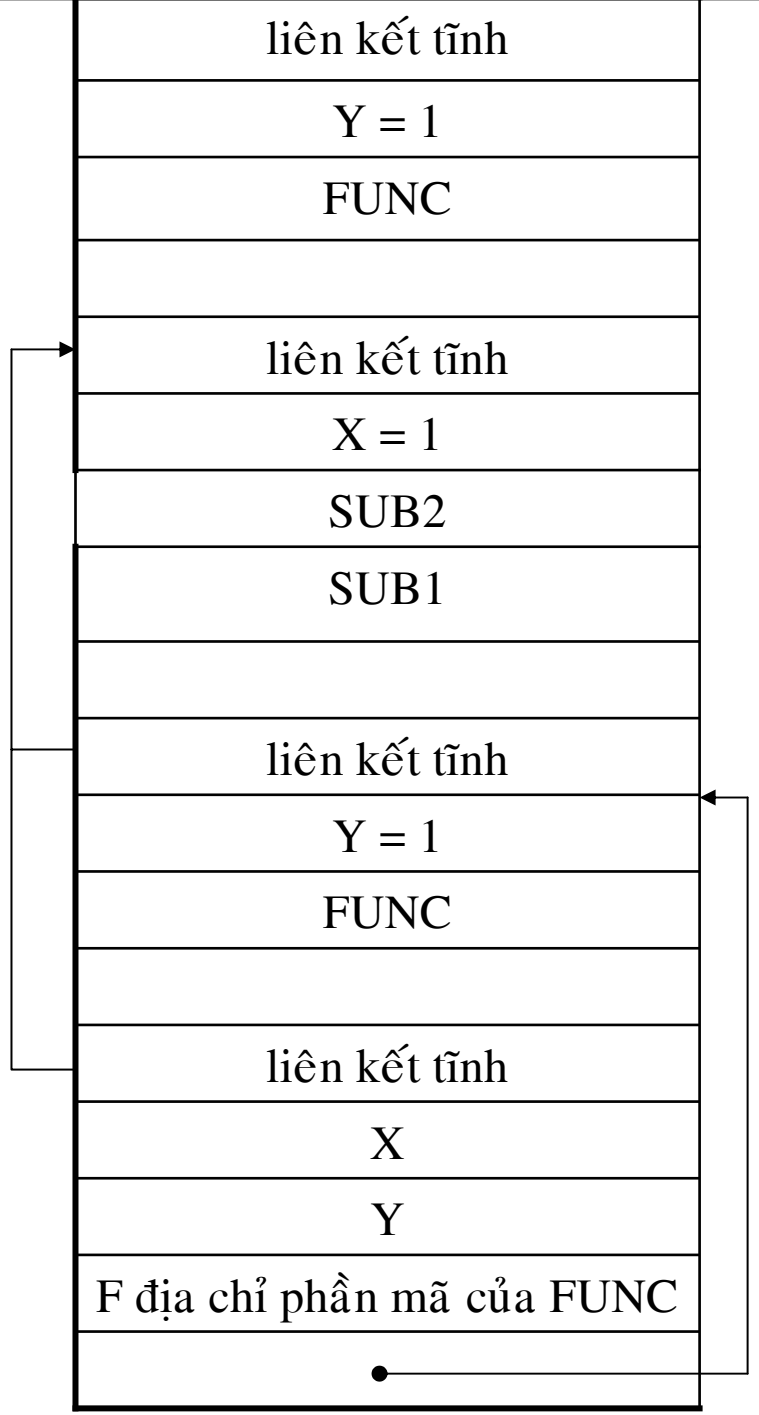
```
program MAIN;
 var X : real;
      procedure SUB2 (X, Y: real; function F (u: real): real);
var z: real;
begin
      z := abs (Y - X);
      z := (F (X) + F (Y)) * Z/2;
      write (Z);
end;
      procedure SUB1;
var Y :real;
```

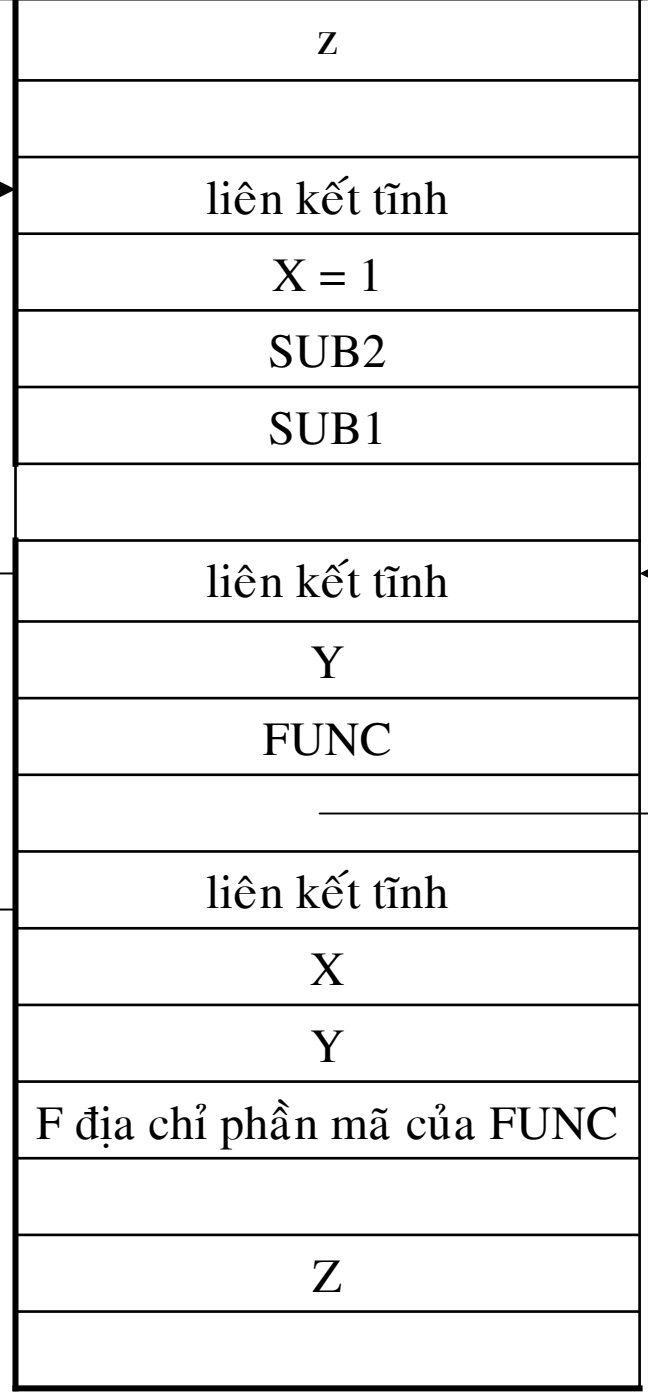
```
        function FUNC: (V: real): real;
        begin
            FUNC := X + V + Y
        end;
begin
        Y := 1
        SUB2 (0, 1, FUNC)
end;
        begin
            X := 3;
            SUB1;
        end.
```

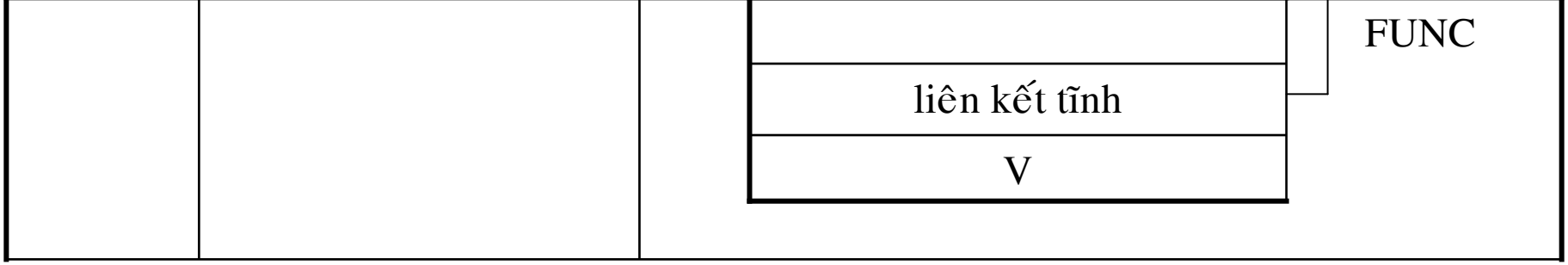
Nhìn vào chương trình trên chúng ta thấy trình tự thực thi của chương trình như sau: MAIN gọi SUB1 gọi SUB2 (0, 1, FUNC) gọi FUNC

Bảng 7.3. *Stack trung tâm khi một chương trình con gọi chương trình con khác thông qua thông số hình thức*

Bước 1	Sự thực thi	Stack trung tâm
1	2	3
1	MAIN	
2	MAIN gọi SUB1	

3	SUB1 gọi SUB2		

4	SUB2 gọi FUNC		


FUNC
liên kết tĩnh
V