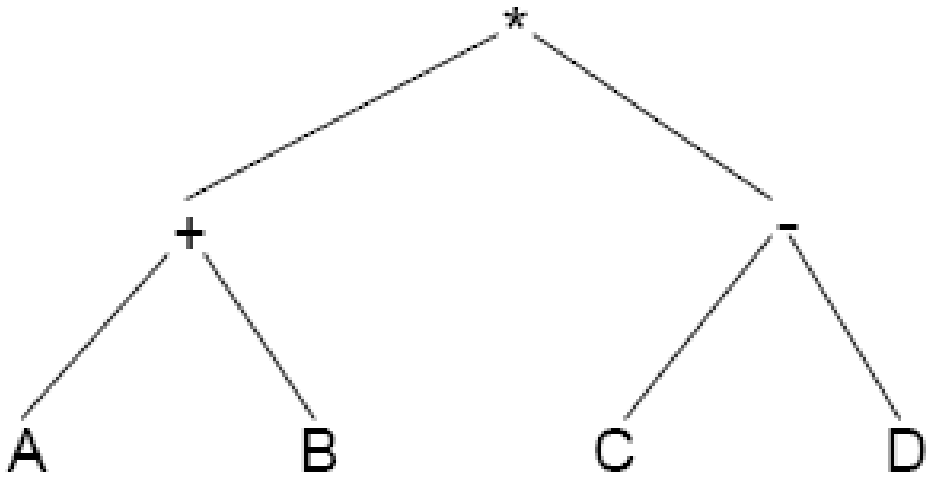
CHƯƠNG 6: ĐIỀU KHIỂN TRÌNH TỰ

Nội dung

- Cấp biểu thức
- Cấp phát biểu
- Cấp chương trình con

ĐIỀU KHIỂN TRÌNH TỰ TRONG BIỂU THỨC

- Chồng chất hàm: kết quả của tác vụ này là toán hạng của tác vụ khác



(A + B) * (C – D)

VẤN ĐỀ LIÊN QUAN

- Quan hệ giữa thứ tự thực hiện phép toán và phép tham khảo dữ liệu
  - a + b * a
- Hiệu ứng lề: a + F(x) * a

  Giả sử ban đầu a = 1 và F(x) trả về 3 và làm a tăng 1.

  a+ → F(x) → a* = 1 + 3 * 2 = 6

  a* → F(x) → a+ = 2 + 3 * 1 = 5

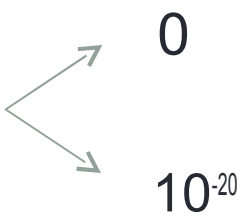
  F(x) → a+ → a* = 2 + 3 * 2 = 8

  a+ → a* → F(x) = 1 + 3 * 1 = 4

VẤN ĐỀ LIÊN QUAN

- Vượt tầm trị

  $A * B * C = 10^{-20} * 10^{-20} * 10^{20}$



- Rút ngắn việc tính toán biểu thức luận lý

  true or X = true

  false and X = false

  if (a = 0) or (b/a>c) then ...

  while (i >= lb) and (i<=ub) and (v[i]>0) do ...

CÚ PHÁP BIỂU THỨC

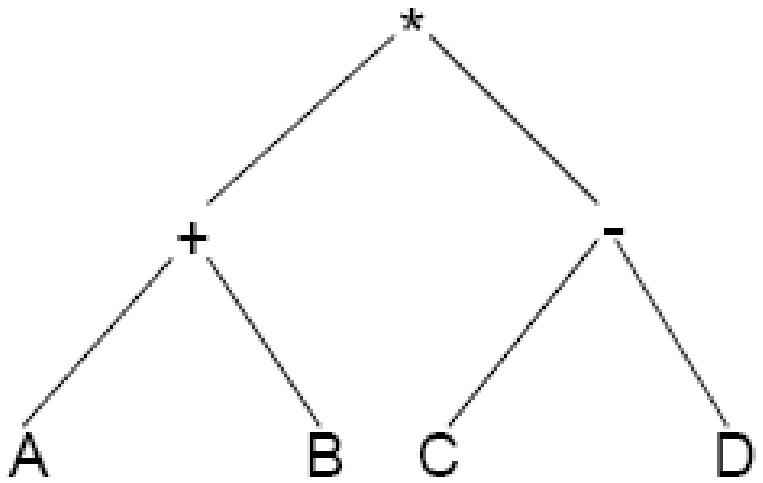
- Dạng trung tố: a + b
  - quen thuộc
  - đòi hỏi qui định độ ưu tiên và tính kết hợp
- Dạng tiền tố
  - tiền tố thường +(a,b)
  - tiền tố Cambridge Polish (+ a b)
  - tiền tố Polish + a b

CÚ PHÁP BIỂU THỨC

- Dạng hậu tố
  - hậu tố thường (a,b)+
  - hậu tố Cam. Pol. (a b +)
  - hậu tố Polish a b +
- Hậu tố thường + Cam. Pol.
  - cho phép số toán hạng tùy ý
  - nhiều dấu ngoặc
- Hậu tố Polish
  - số toán hạng cố định
  - không cần dấu ngoặc

DẠNG THỰC THI CỦA BIỂU THỨC

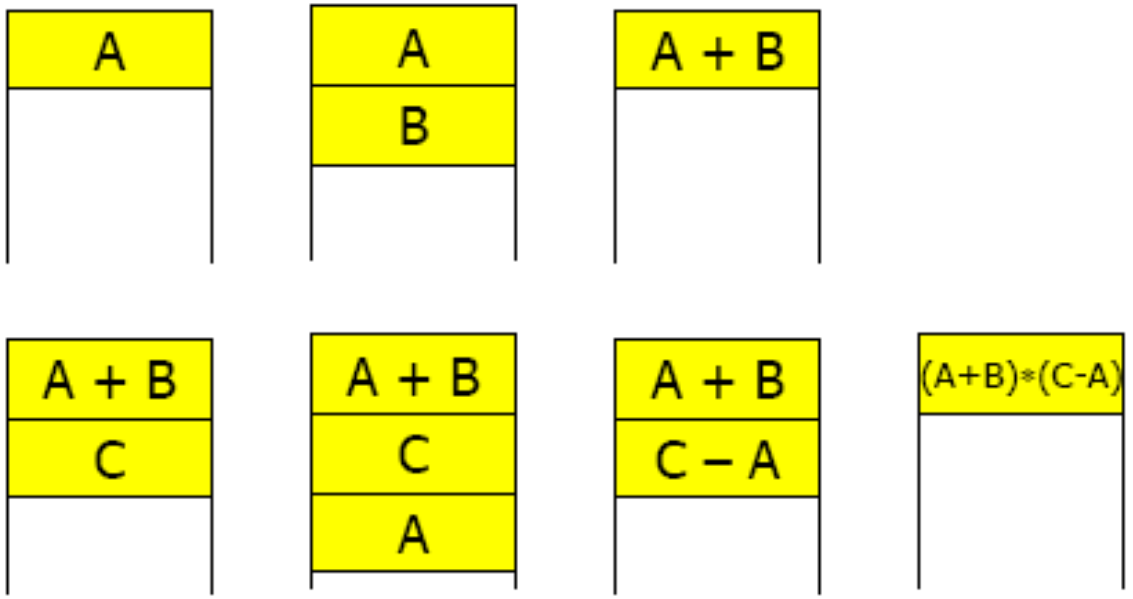
- Biên dịch: mã máy
- Thông dịch
  - Cây → duyệt cây để tính kết quả



DẠNG THỰC THI CỦA BIỂU THỨC

– hậu tố A B + C A – *

push A
push B
add
push C
push A
sub
mul



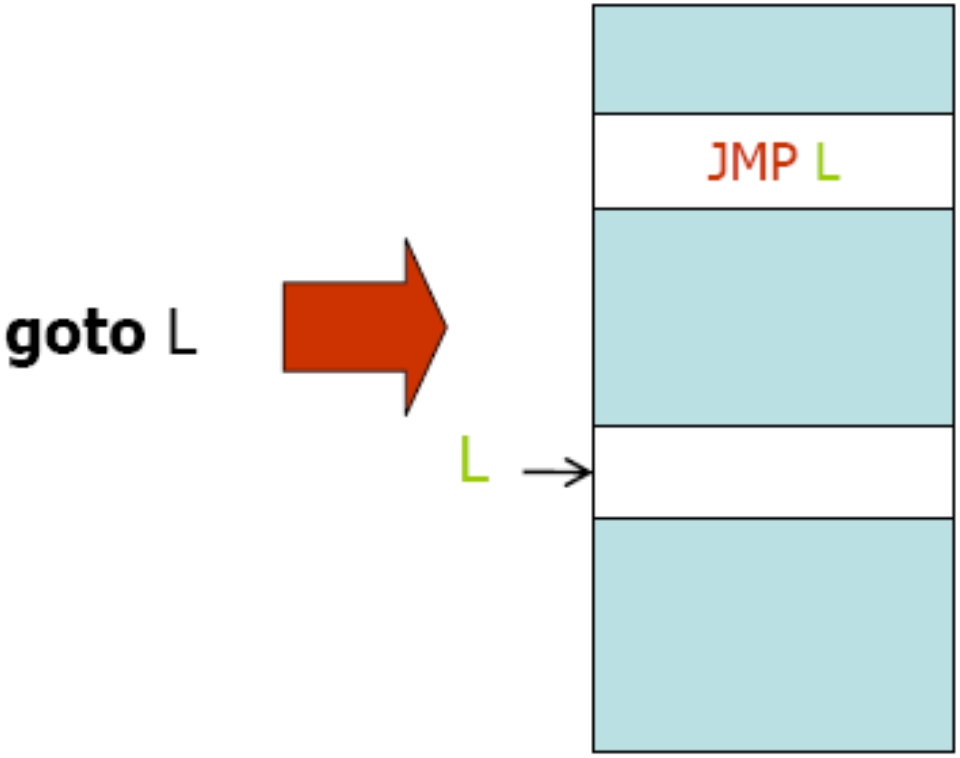
- tiền tố→ đệ qui

ĐIỀU KHIỂN TRÌNH TỰ GIỮA CÁC PHÁT BIỂU

- Goto
- Ghép
- Rẽ nhánh
  - if
  - case (switch)
- Lặp
  - while
  - repeat
  - for

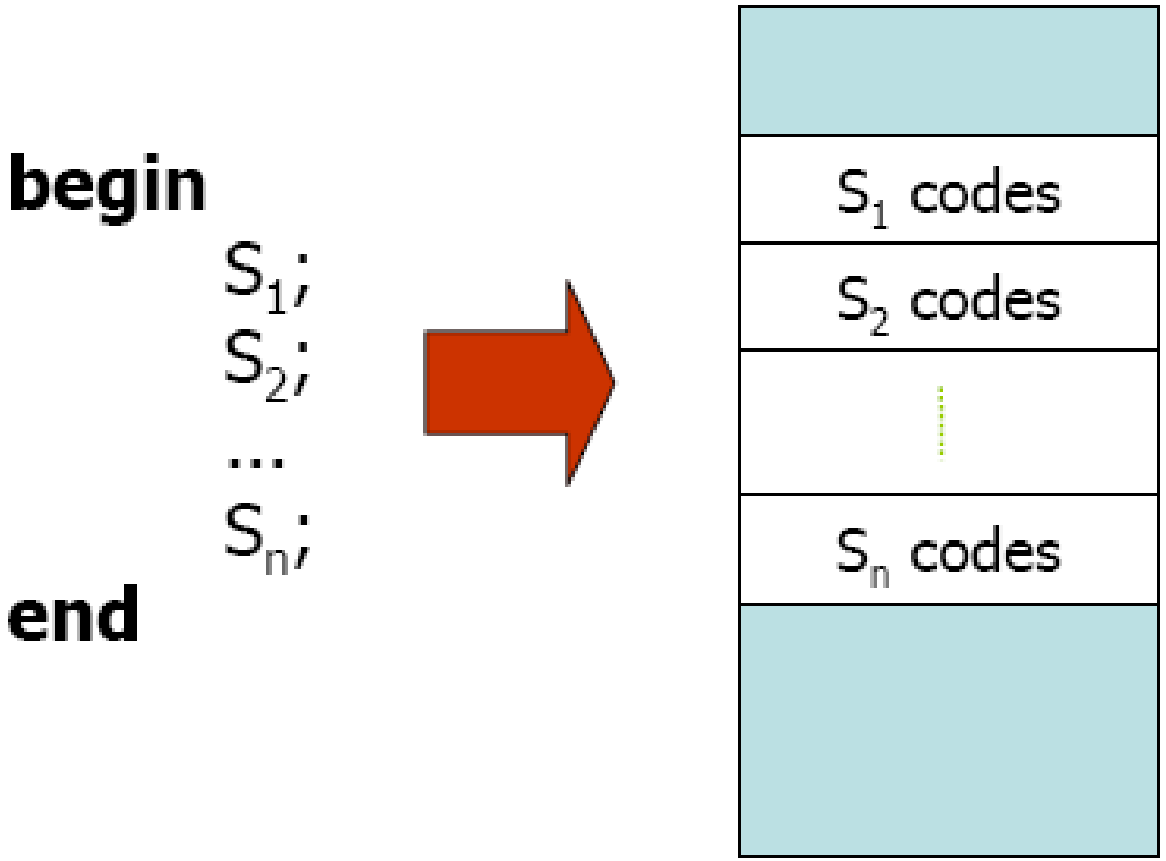
GOTO

- thuận tiện
- mất tính cấu trúc
- khó đọc
- khó sửa sai

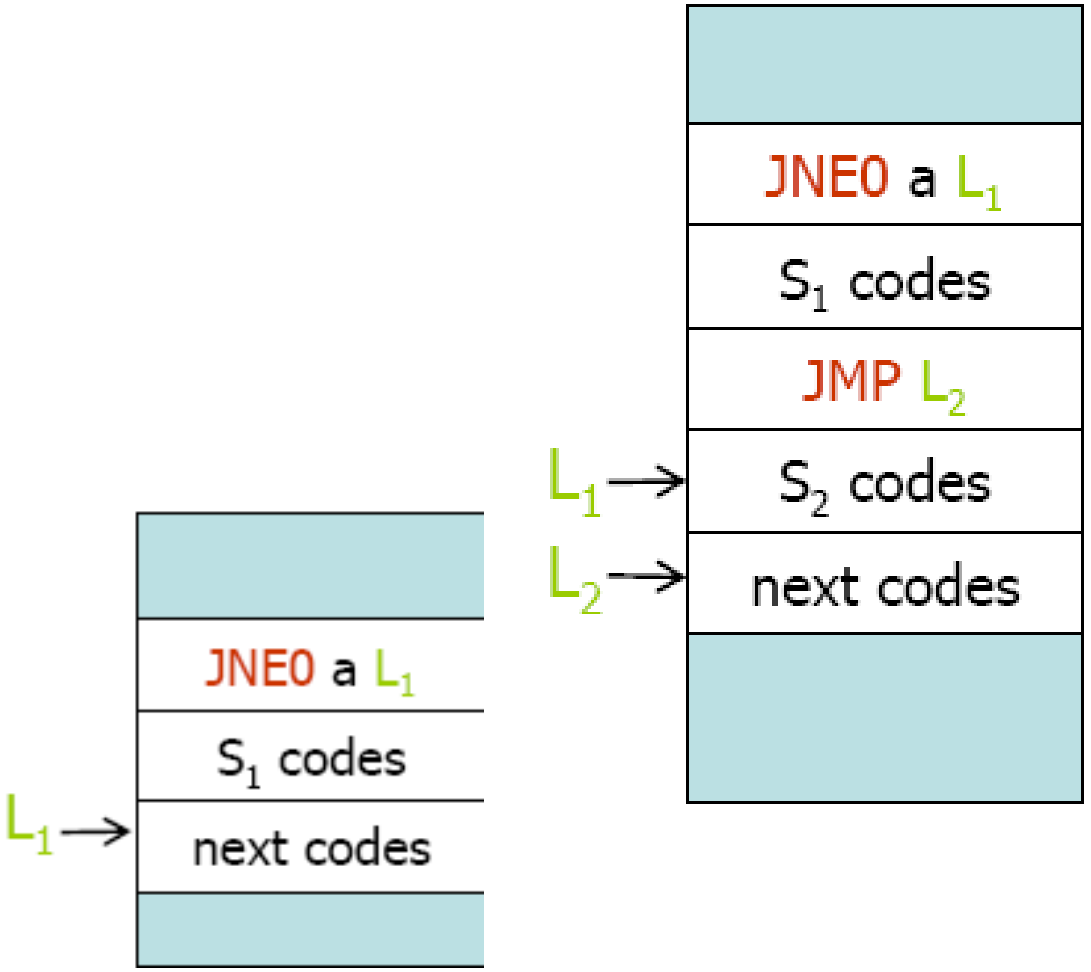


GHÉP

- chuỗi nhiều phát biểu được xử lý như 1 phát biểu
- cơ chế điều khiển cơ bản



ĐIỀU KIỆN


JNE0 a L1
S1 codes
next codes
JNE0 a L1
S1 codes
JMP L2
L1
S2 codes
L2
next codes

ĐIỀU KIỆN

- CASE

```
case bt of
   tri1: S1;
   tri2: S2;
   ...
   else Sn;
end
```

```
switch (bt) {
   case tri1: S1;
   case tri2: S2;
   ...
   default: Sn;
}
```

```
docase
   case bt1 S1
   case bt2 S2
   ...
   otherwise Sn
endcase
```

LẶP

L_0: if not dk goto L_1

S

goto L_0

L_1:

LẶP

- FOR
  - for i := bt1 to/downto bt2 do S
  - for (bt1; bt2; bt3) S
  - →
  - tầm vực biến chỉ số
  - thay đổi trị của bt2 và bt3 sau mỗi lần lặp

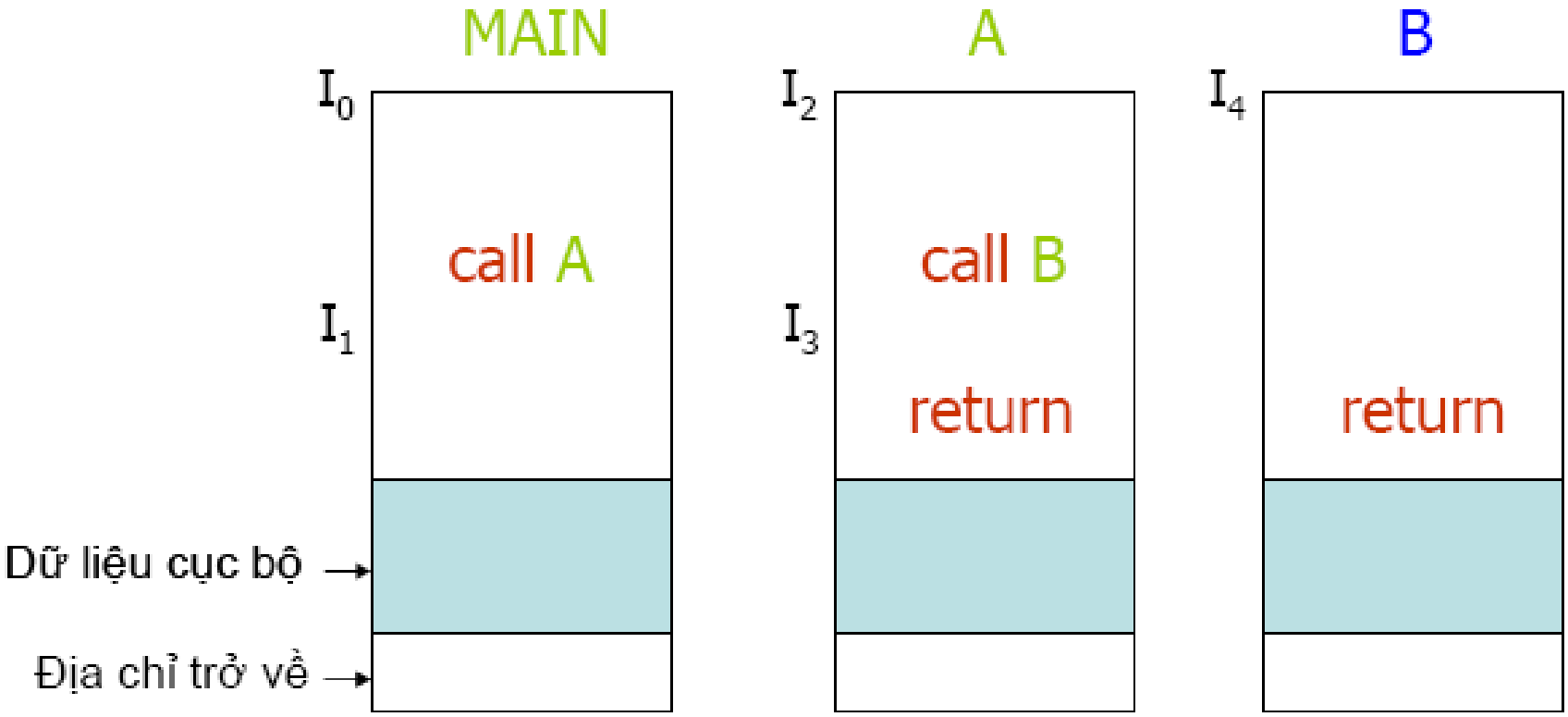
CẤU TRÚC ĐIỀU KHIỂN TRÌNH TỰ CẤP ĐƠN VỊ

- Cấu trúc gọi - trở về đơn giản
- Cấu trúc gọi đệ qui
- Biến cố và trình xử lý biến cố
- Trình cộng hành
- Trình định thời
- Công tác và sự thực thi đồng thời

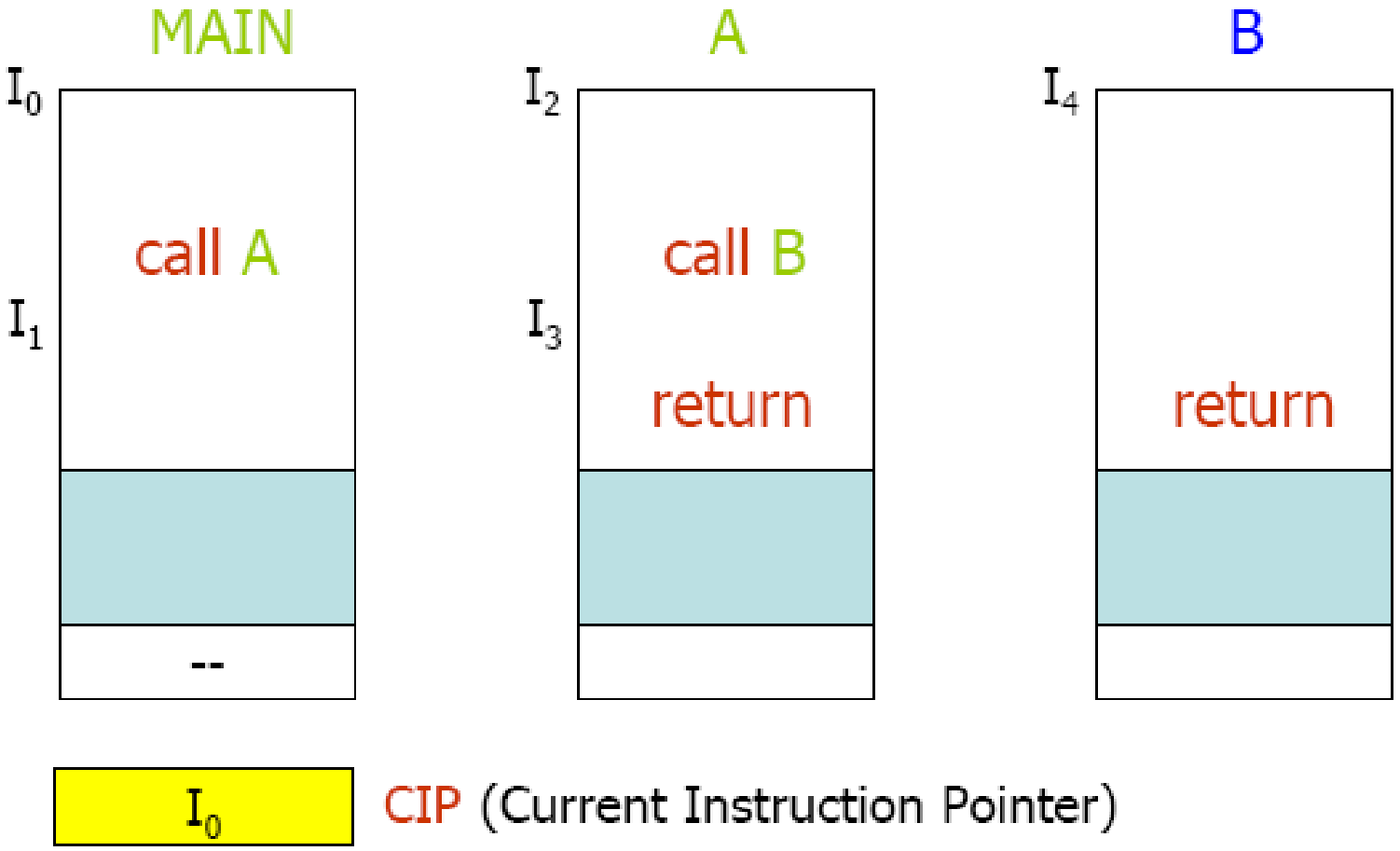
CẤU TRÚC GỌI - TRỞ VỀ ĐƠN GIẢN

- Không gọi đệ qui (trực tiếp hay gián tiếp)
- Phép gọi tường minh
- Thực thi hoàn toàn
- Chuyển điều khiển tức thời tại thời điểm gọi
- Chuỗi thực thi đơn

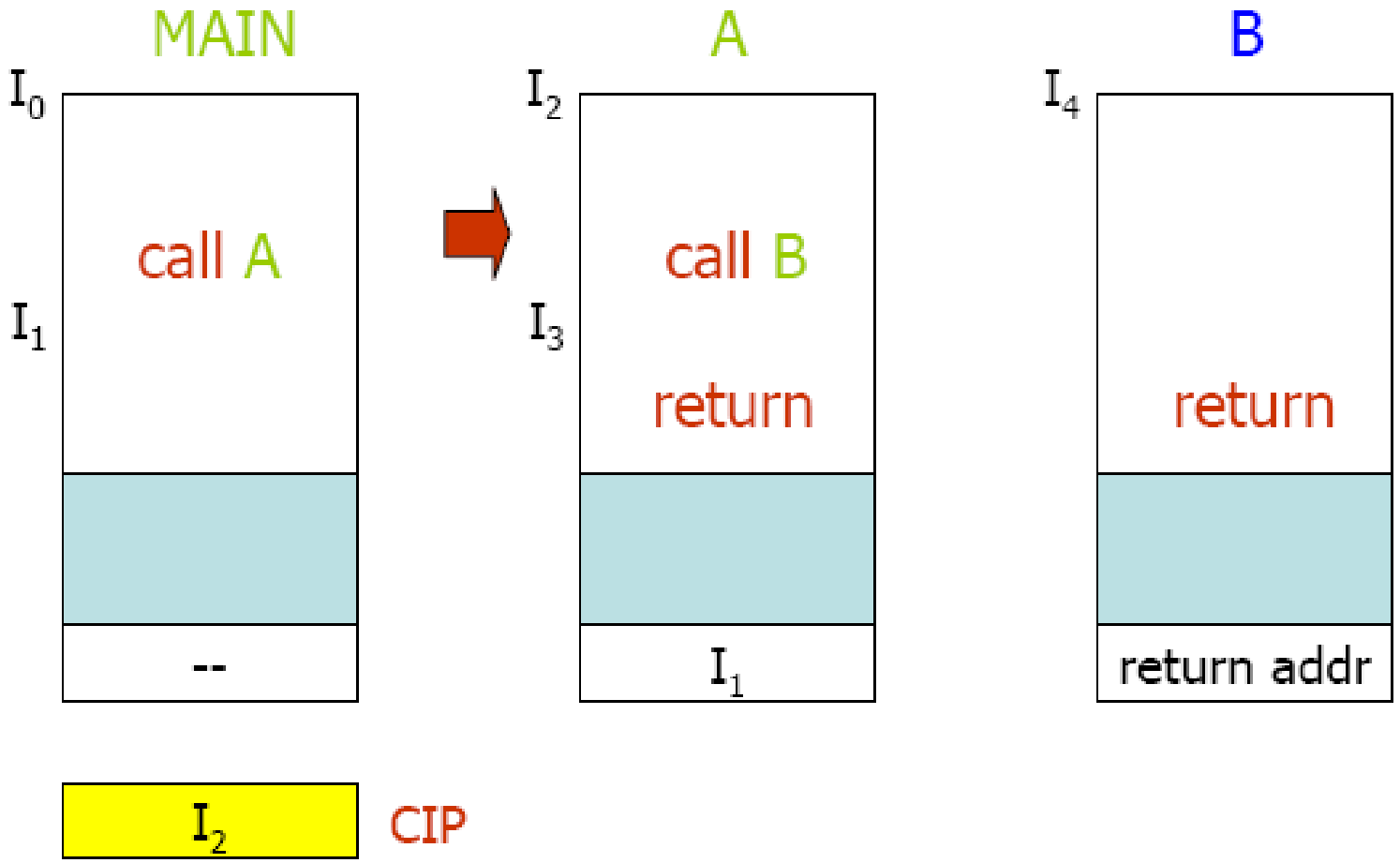
CẤU TRÚC GỌI - TRỞ VỀ ĐƠN GIẢN



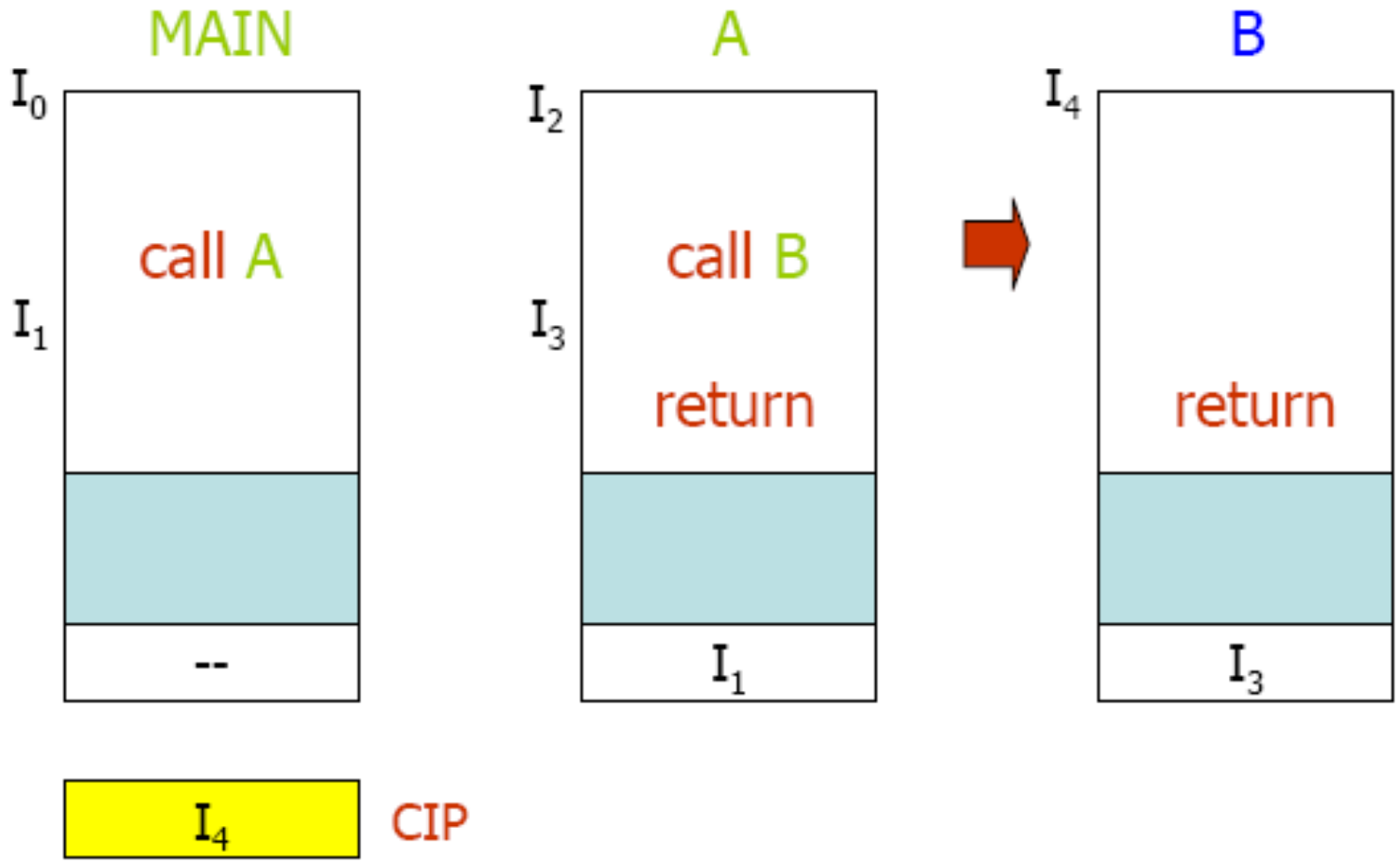
CẤU TRÚC GỌI - TRỞ VỀ ĐƠN GIẢN



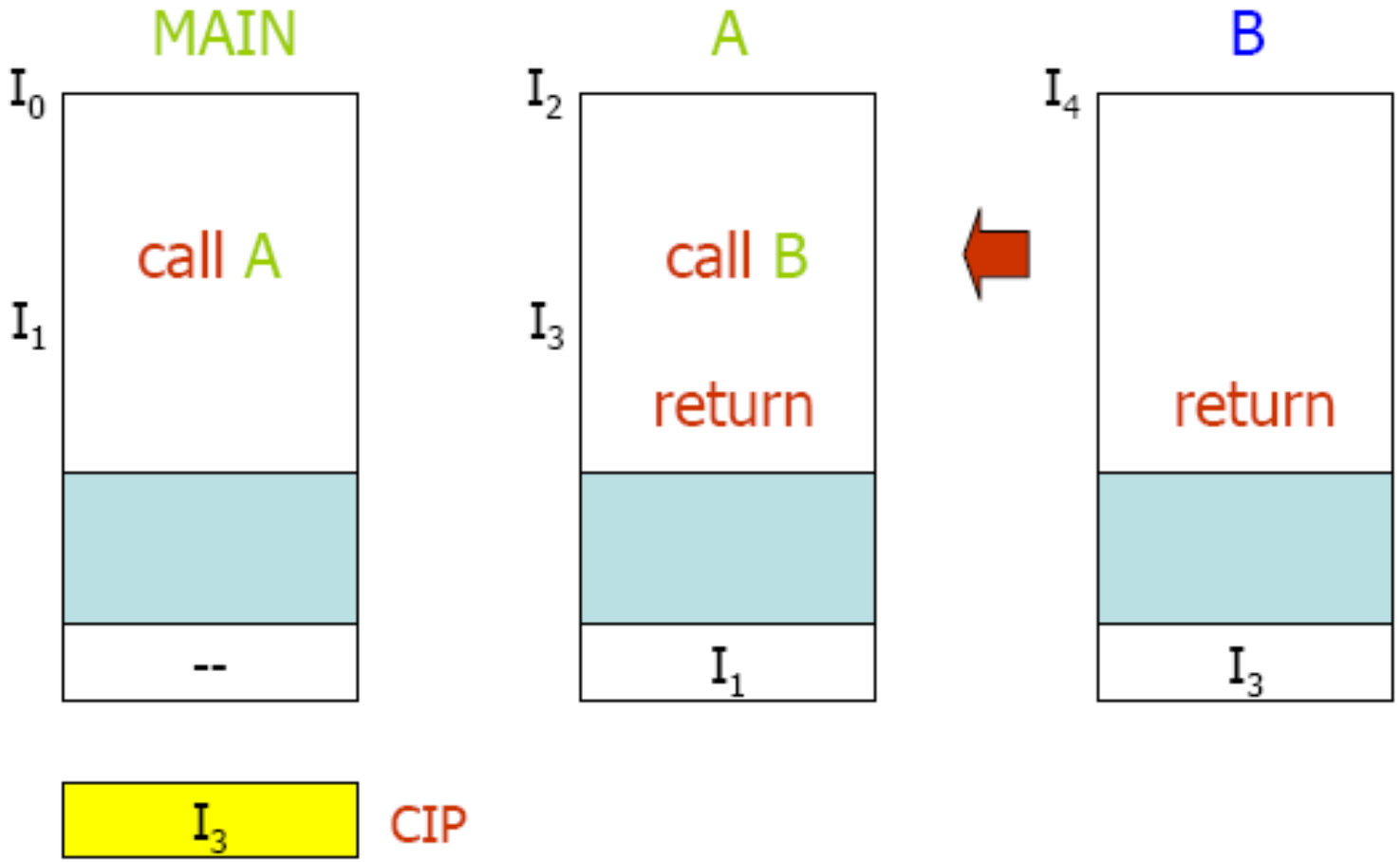
CẤU TRÚC GỌI - TRỞ VỀ ĐƠN GIẢN



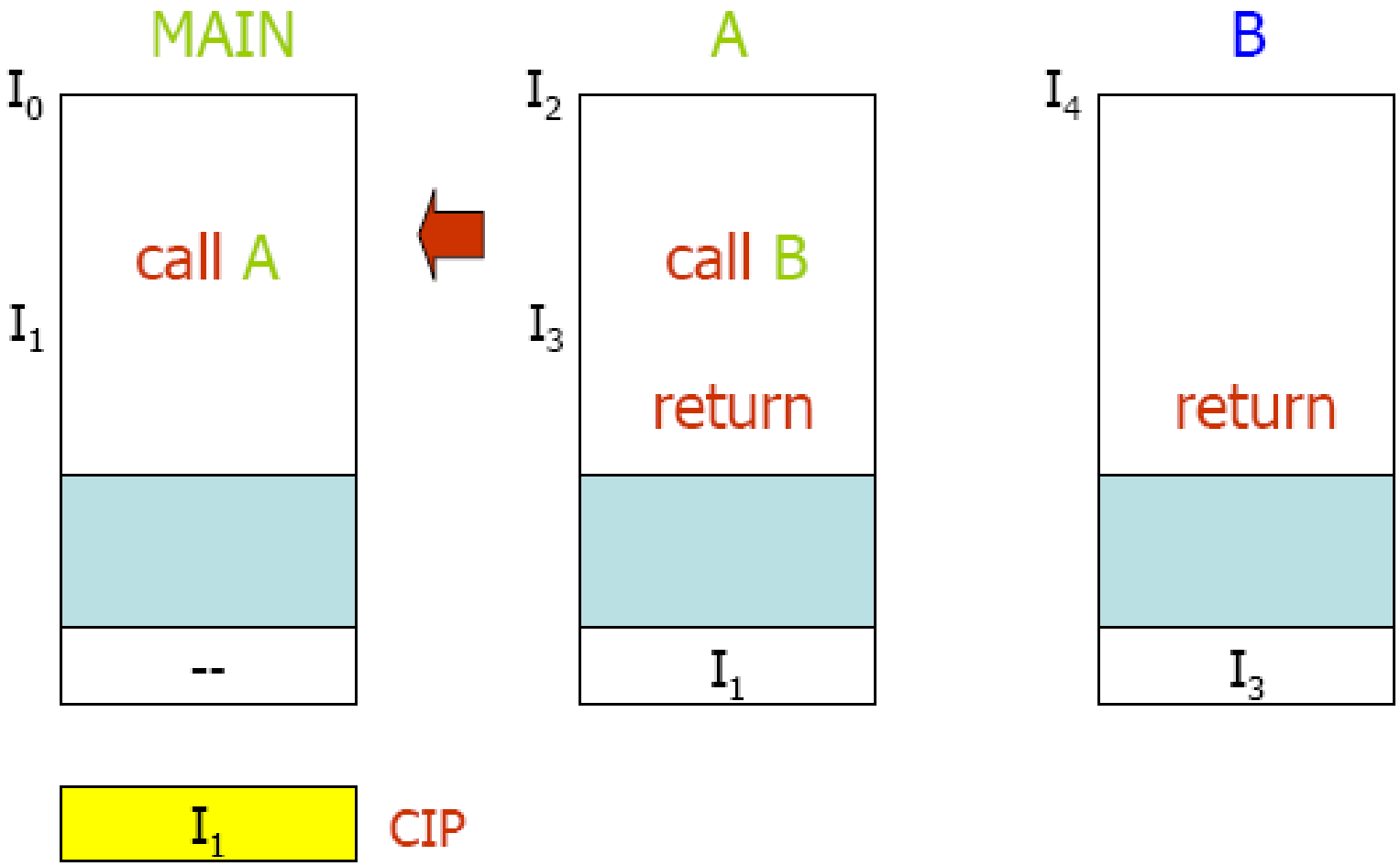
CẤU TRÚC GỌI - TRỞ VỀ ĐƠN GIẢN



CẤU TRÚC GỌI - TRỞ VỀ ĐƠN GIẢN



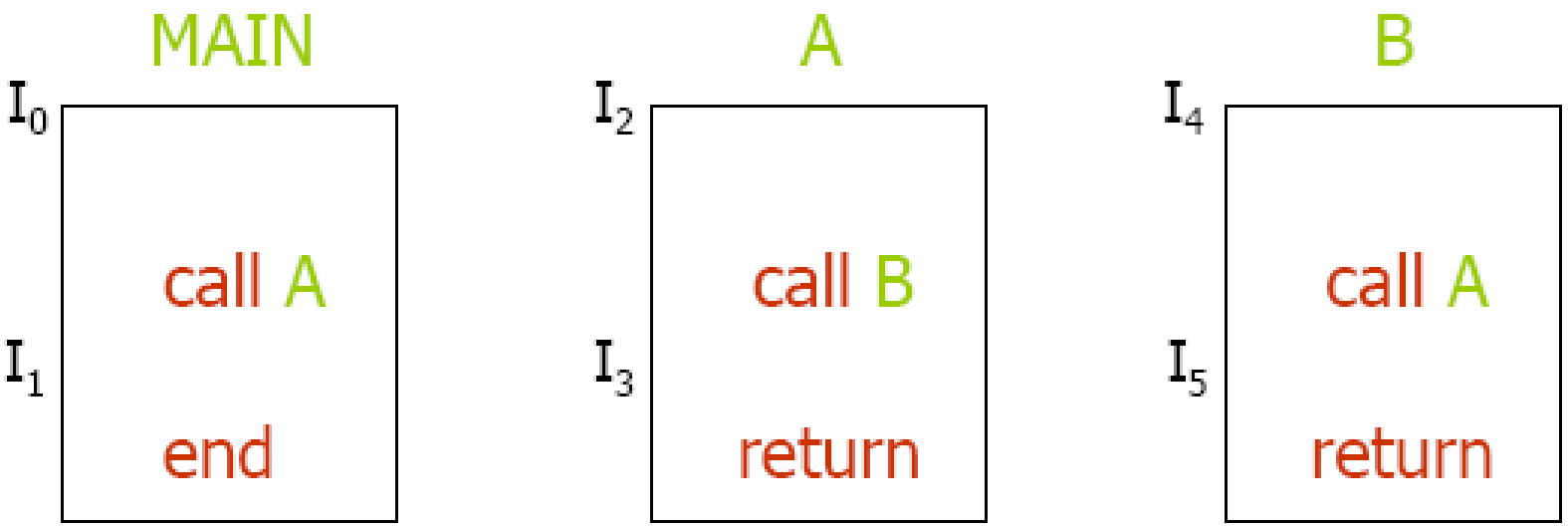
CẤU TRÚC GỌI - TRỞ VỀ ĐƠN GIẢN



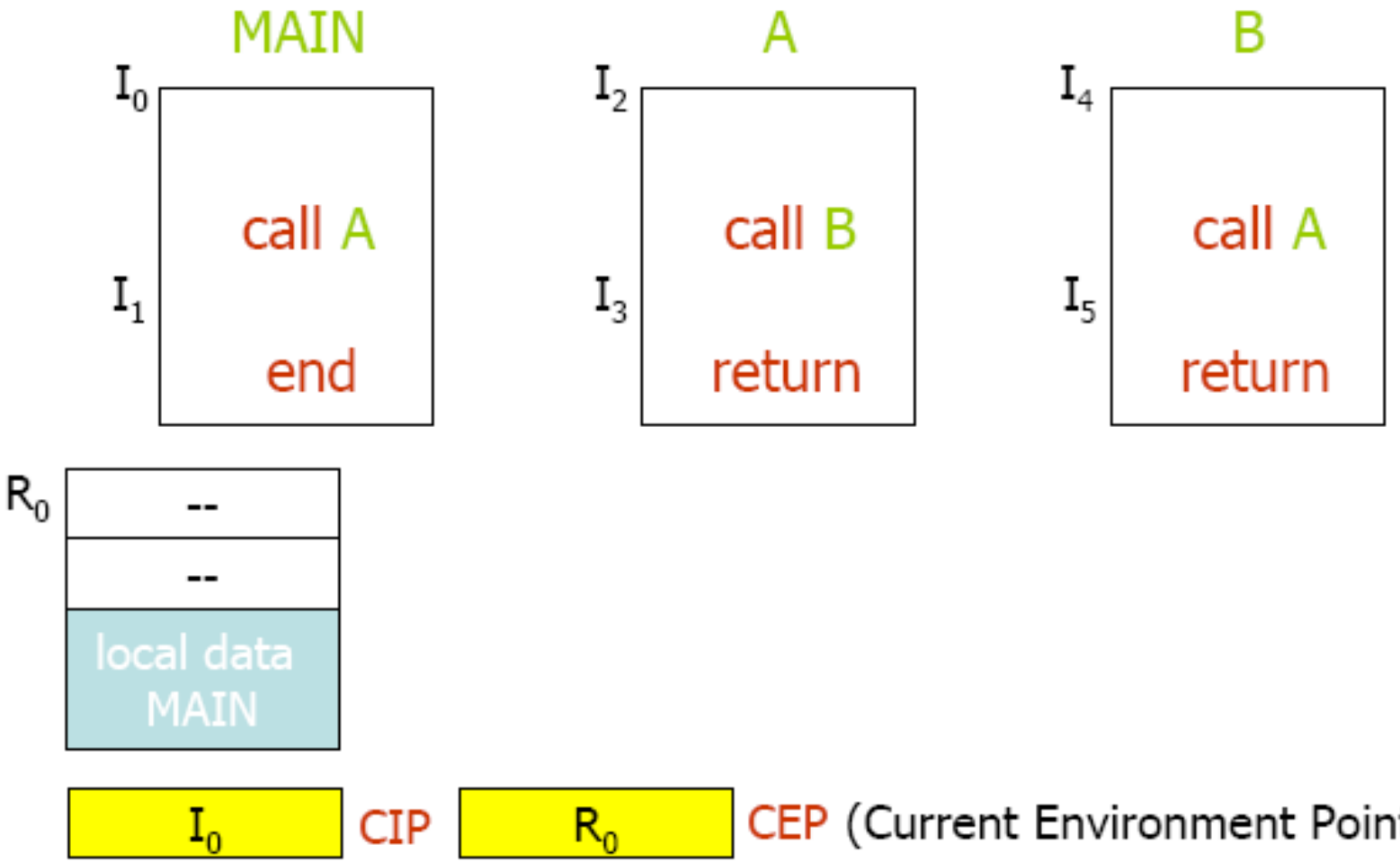
CẤU TRÚC GỌI ĐỆ QUI

- Cho phép gọi đệ qui trực tiếp và gián tiếp
- Các chức năng khác tương tự cấu trúc gọi
  - trở về đơn giản
  - Phép gọi tường minh
  - Thực thi hoàn toàn
  - Chuyển điều khiển tức thời tại thời điểm gọi
  - Chuỗi thực thi đơn

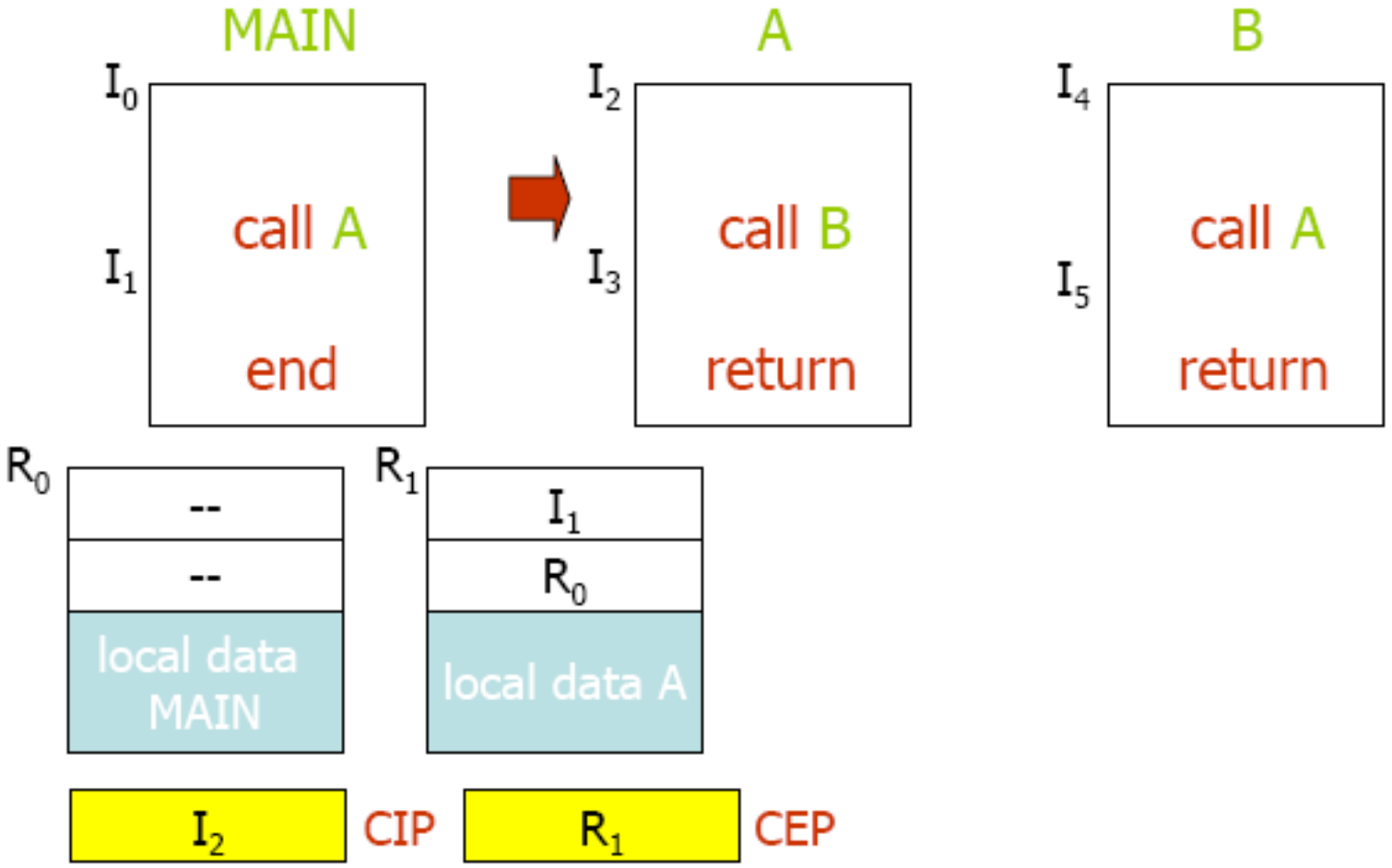
CẤU TRÚC GỌI ĐỆ QUI



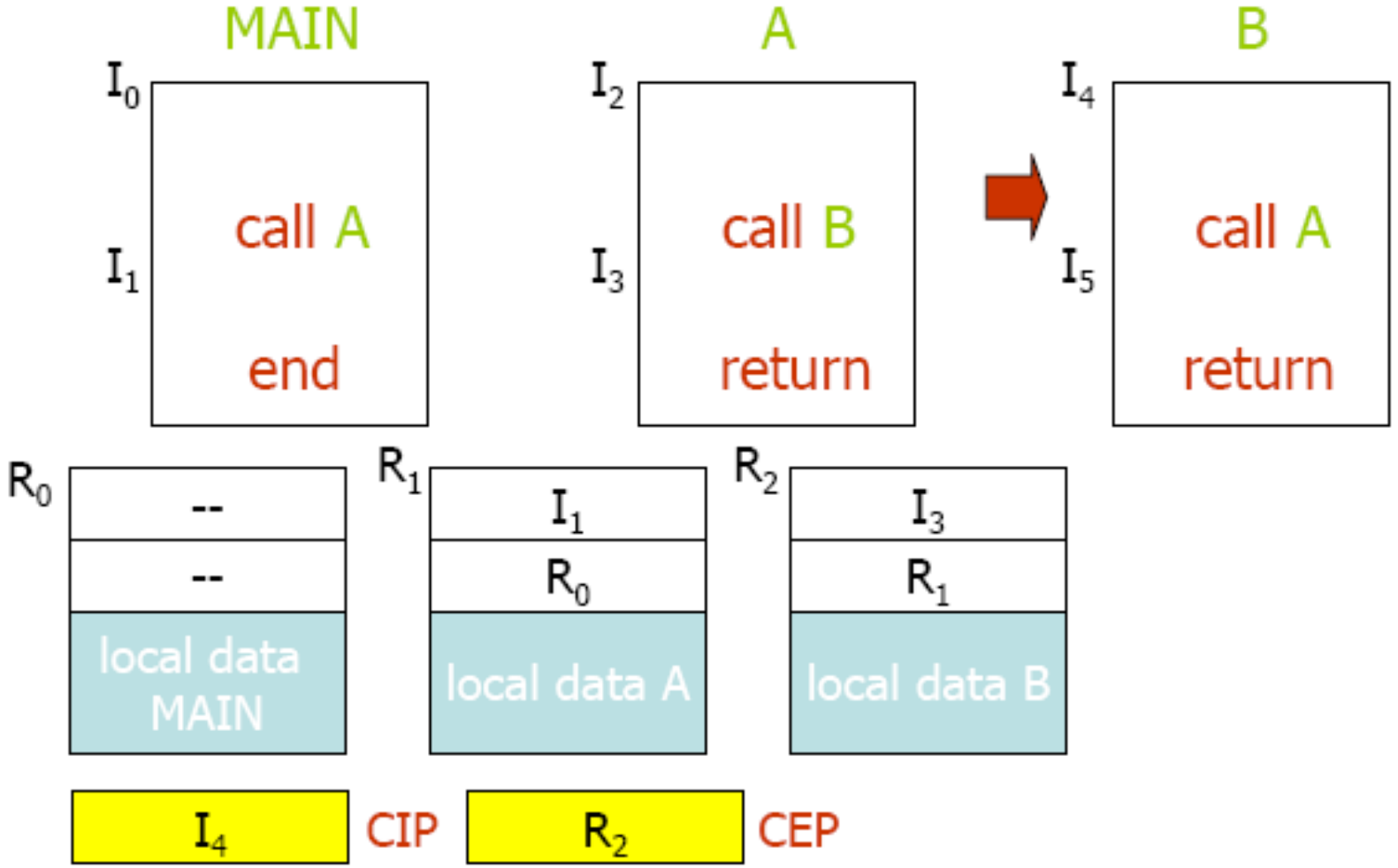
CẤU TRÚC GỌI ĐỆ QUI



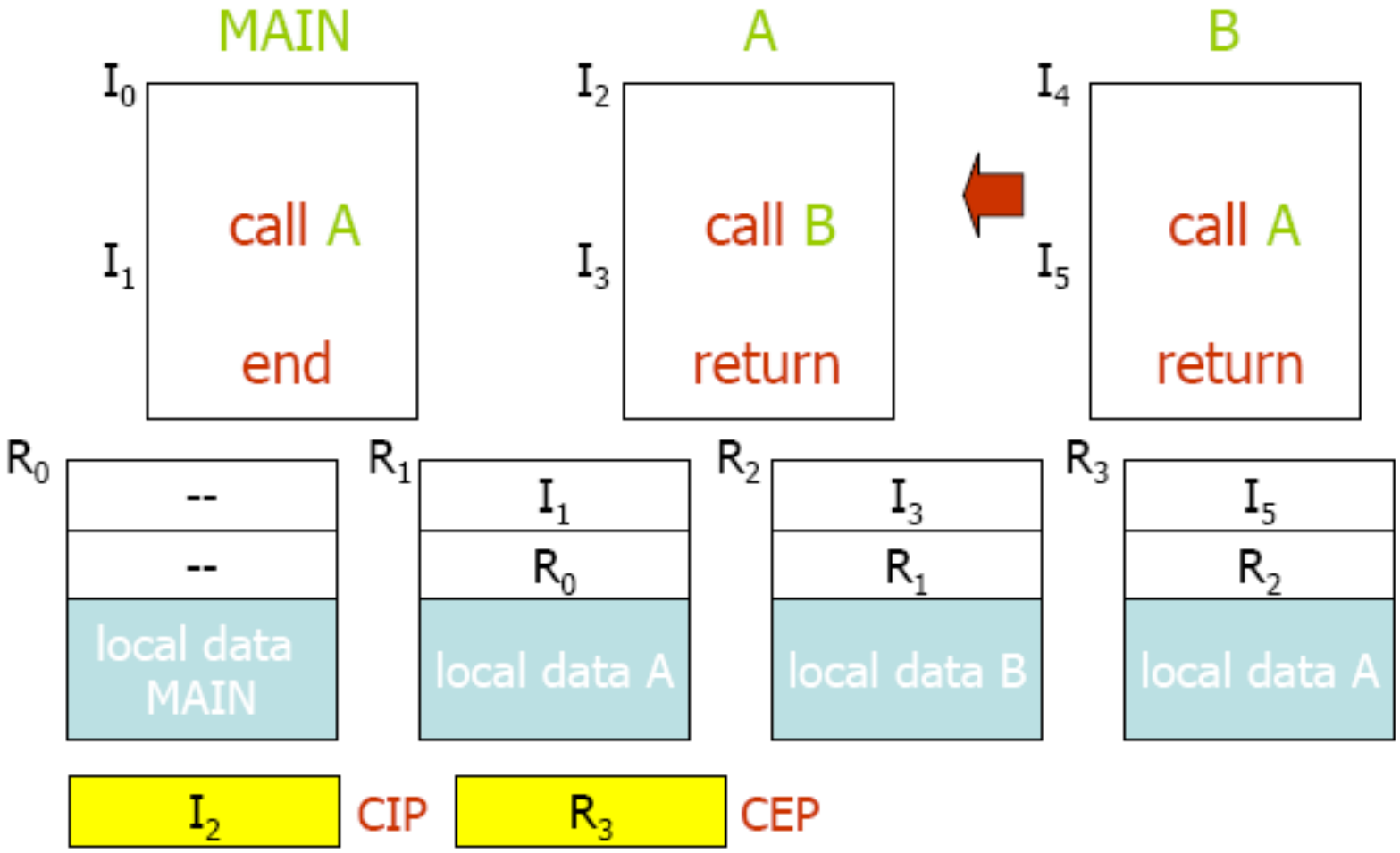
CẤU TRÚC GỌI ĐỆ QUI



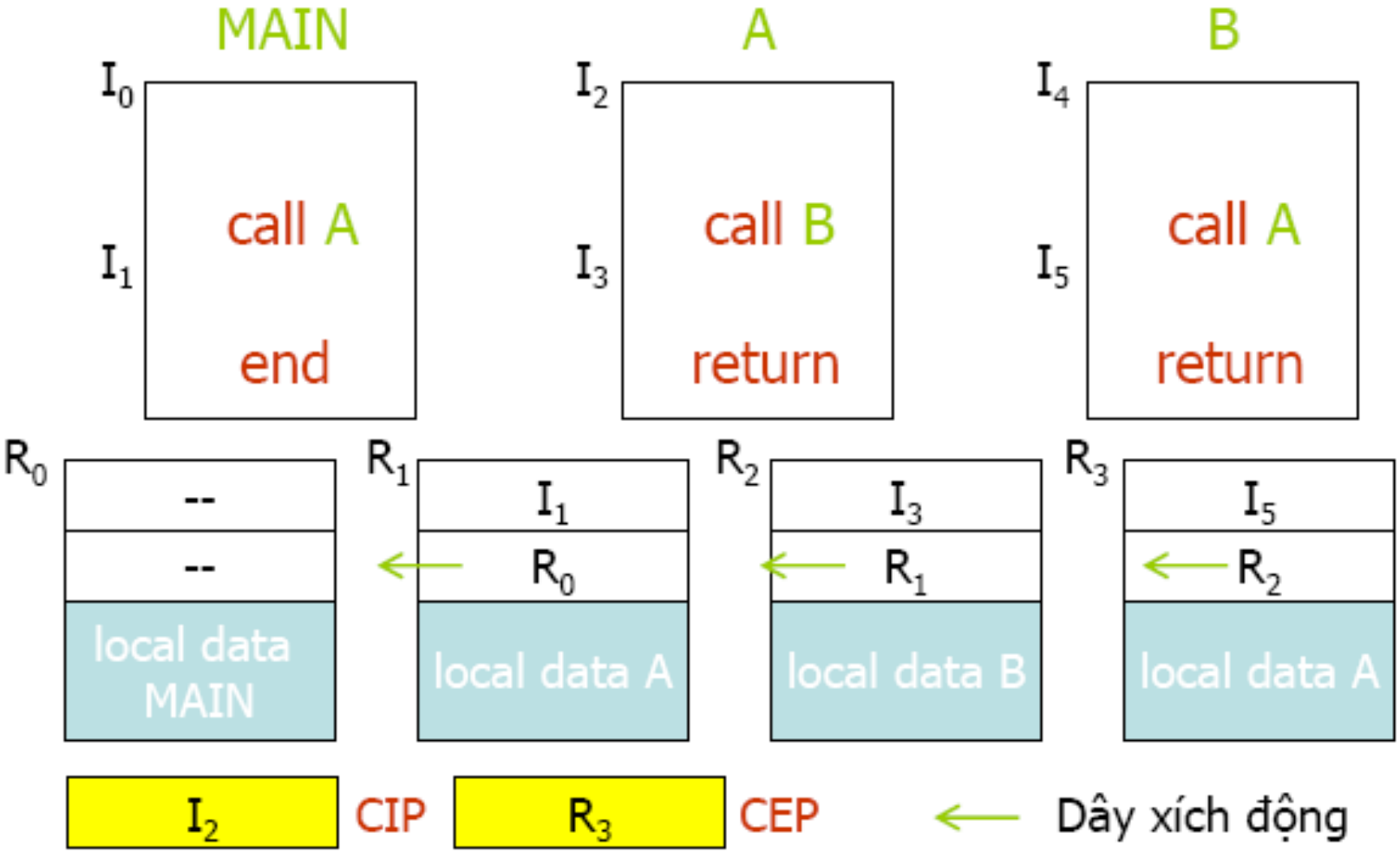
CẤU TRÚC GỌI ĐỆ QUI



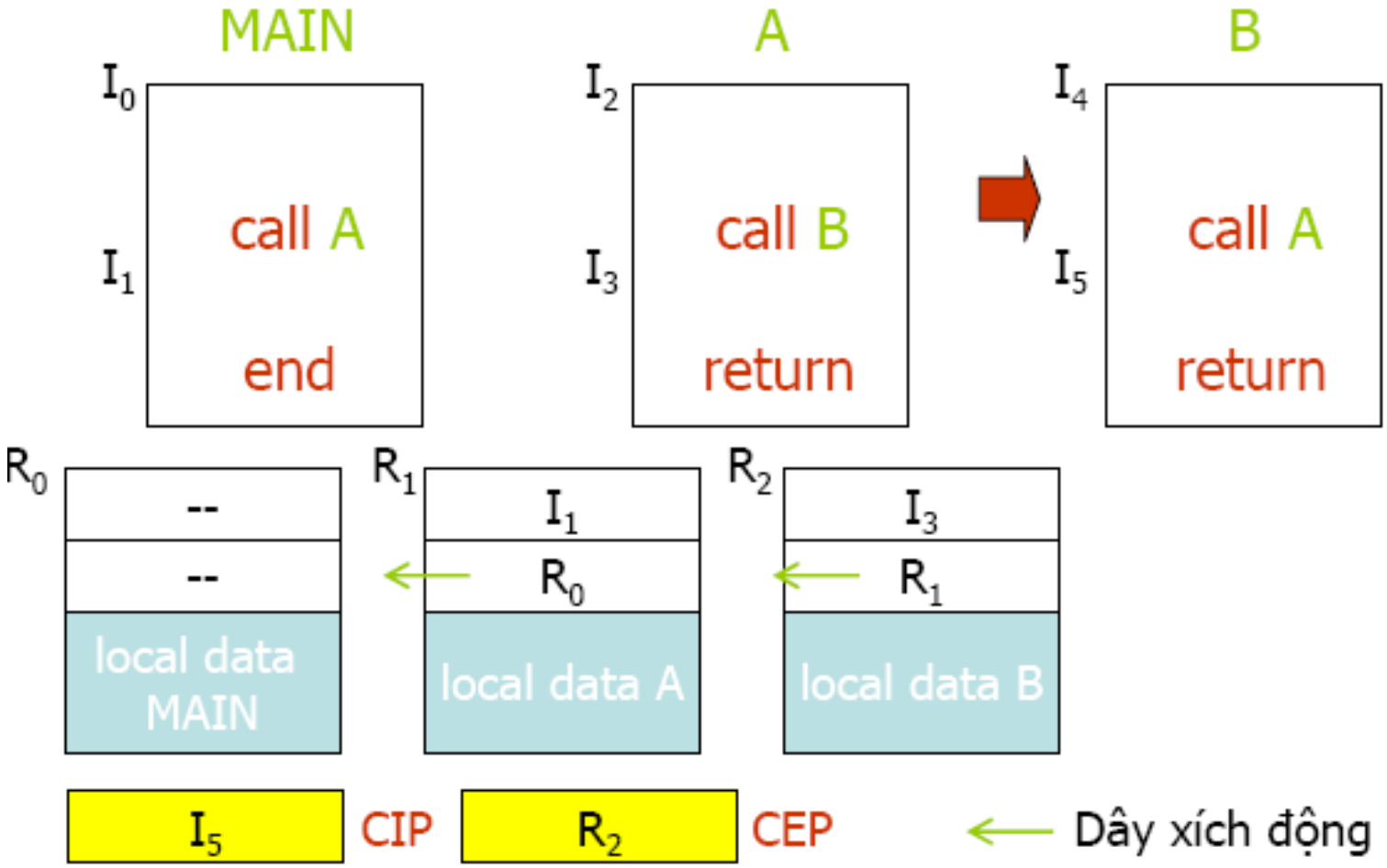
CẤU TRÚC GỌI ĐỆ QUI



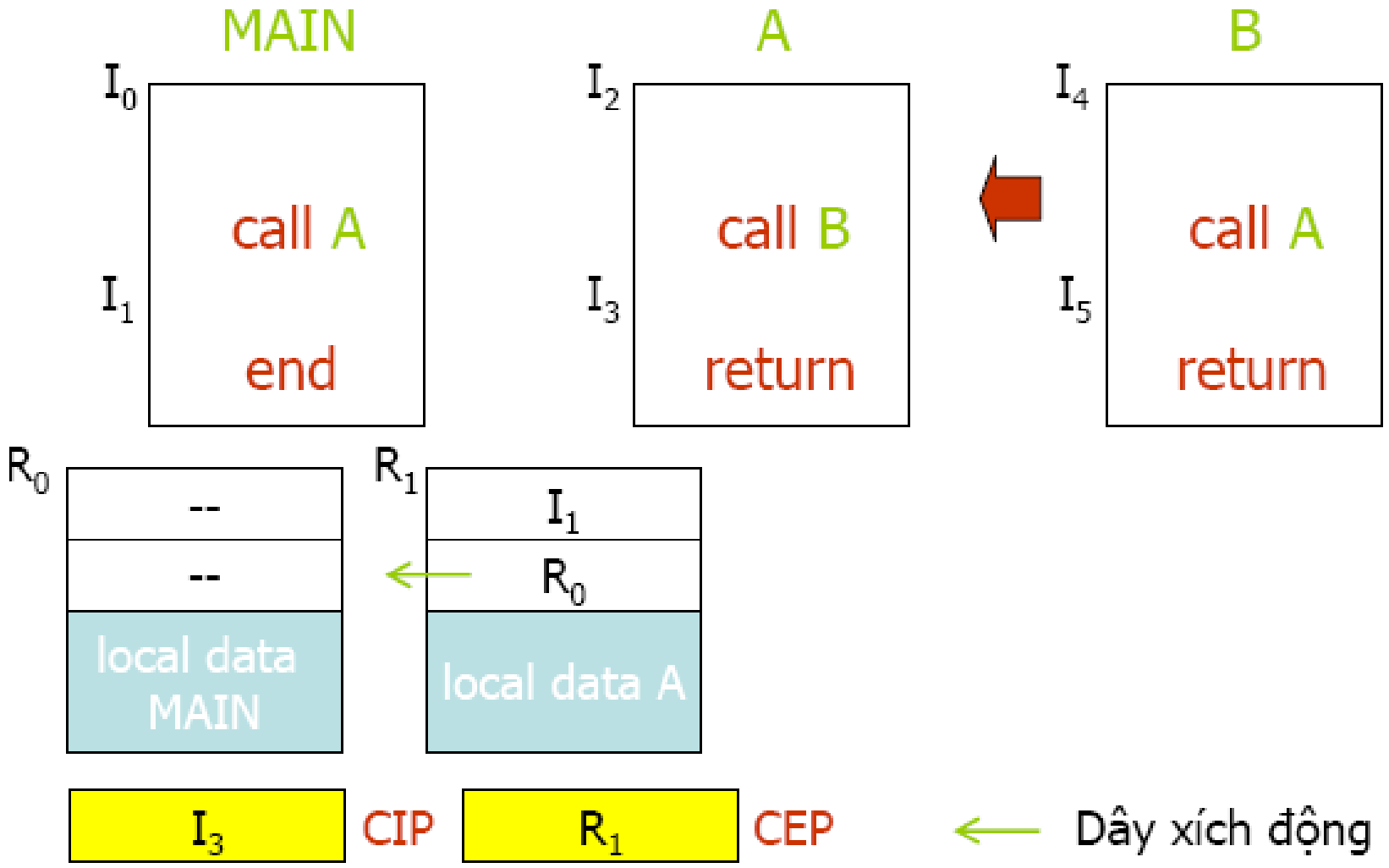
CẤU TRÚC GỌI ĐỆ QUI



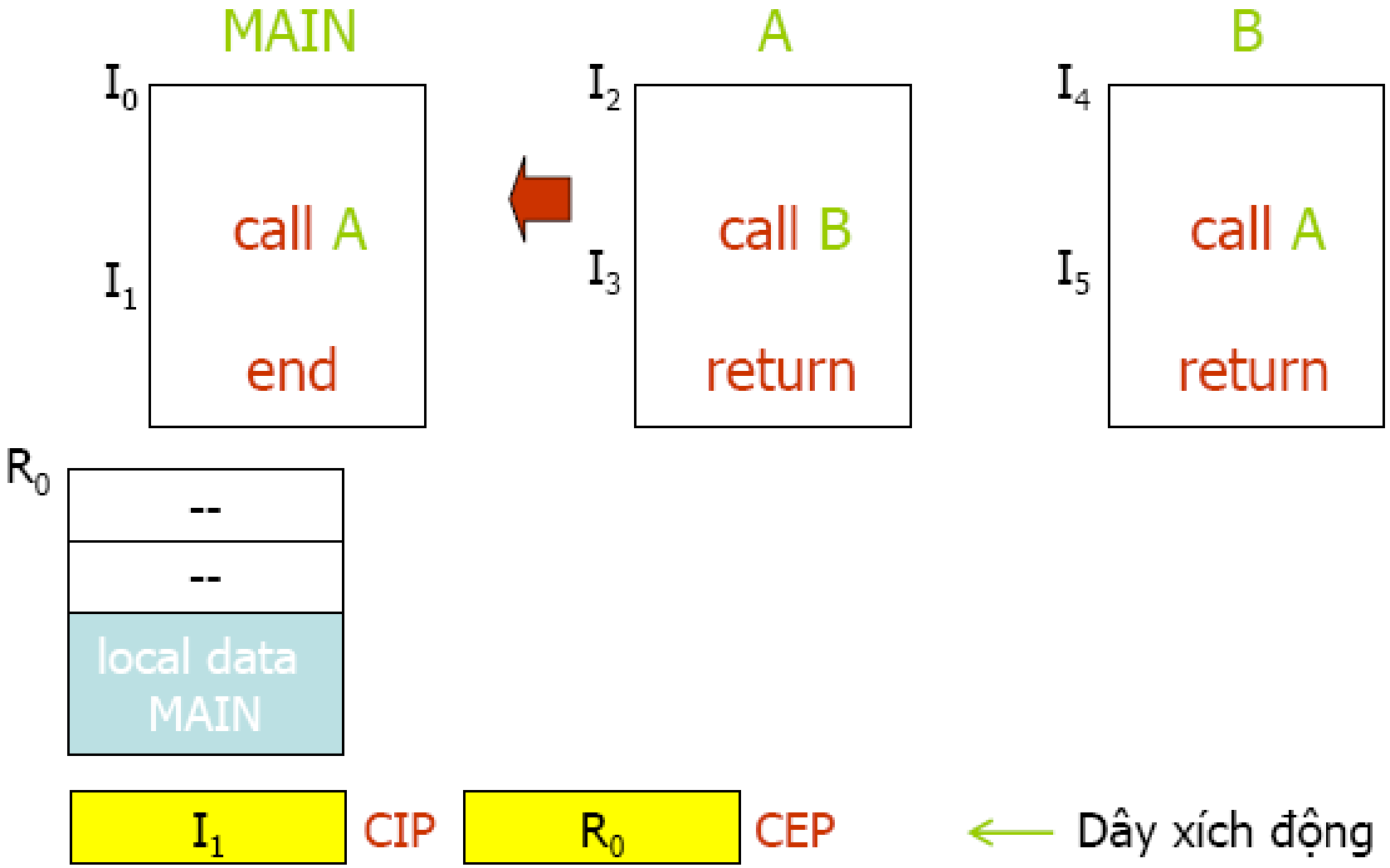
CẤU TRÚC GỌI ĐỆ QUI



CẤU TRÚC GỌI ĐỆ QUI



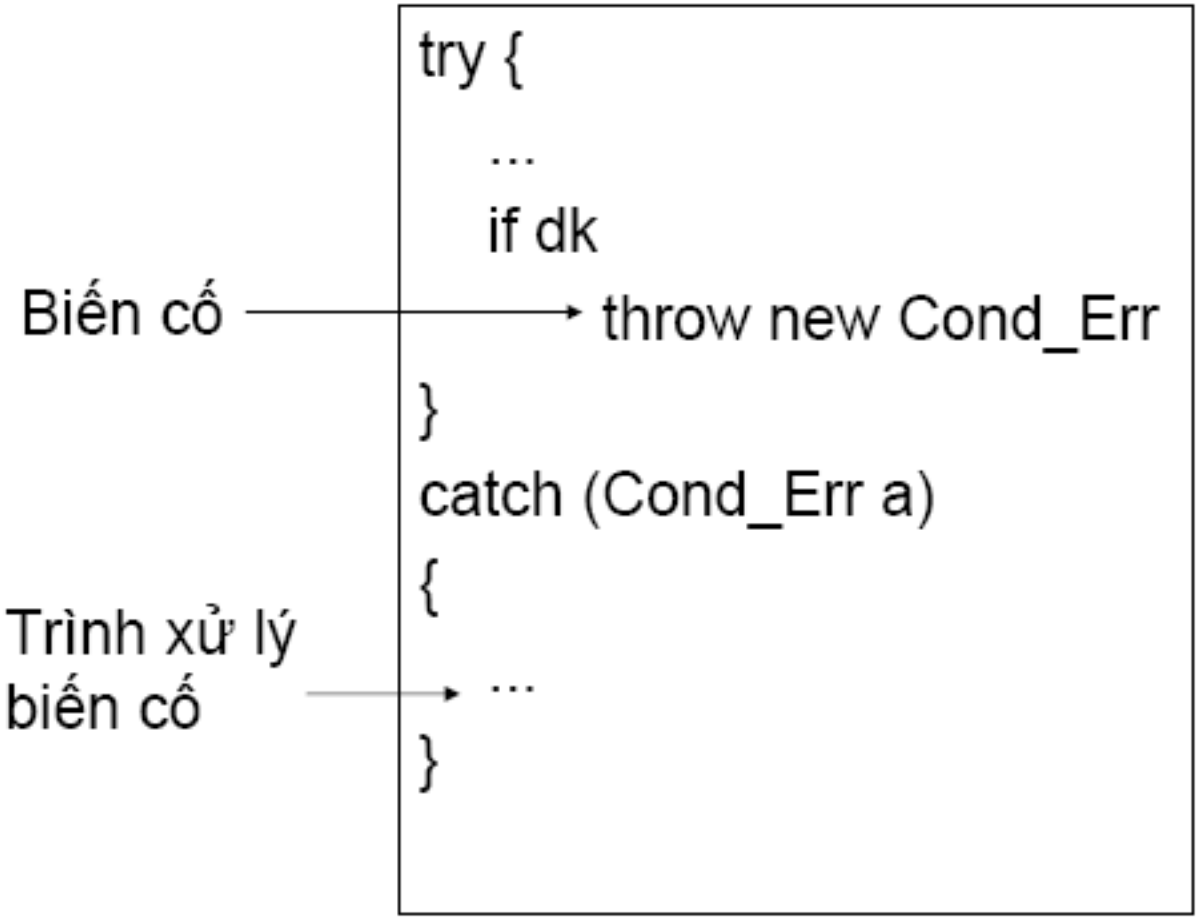
CẤU TRÚC GỌI ĐỆ QUI



BIẾN CỐ VÀ TRÌNH XỬ LÝ BIẾN CỐ

- Phục vụ cho event-driven programming
- Tách rời phần nội dung chương trình và xử lý lỗi
- Biến cố (exception) là những sự kiện ngắt quãng sự thực thi bình thường của chương trình
- Trình xử lý biến cố
  - chỉ được gọi khi biến cố tương ứng xảy ra

VÍ DỤ


try {
...
if dk
Biến cố
throw new Cond_Err
}
catch (Cond_Err a)
{
Trình xử lý
biến cố
...
}

PHÁT SINH BIẾN CỐ

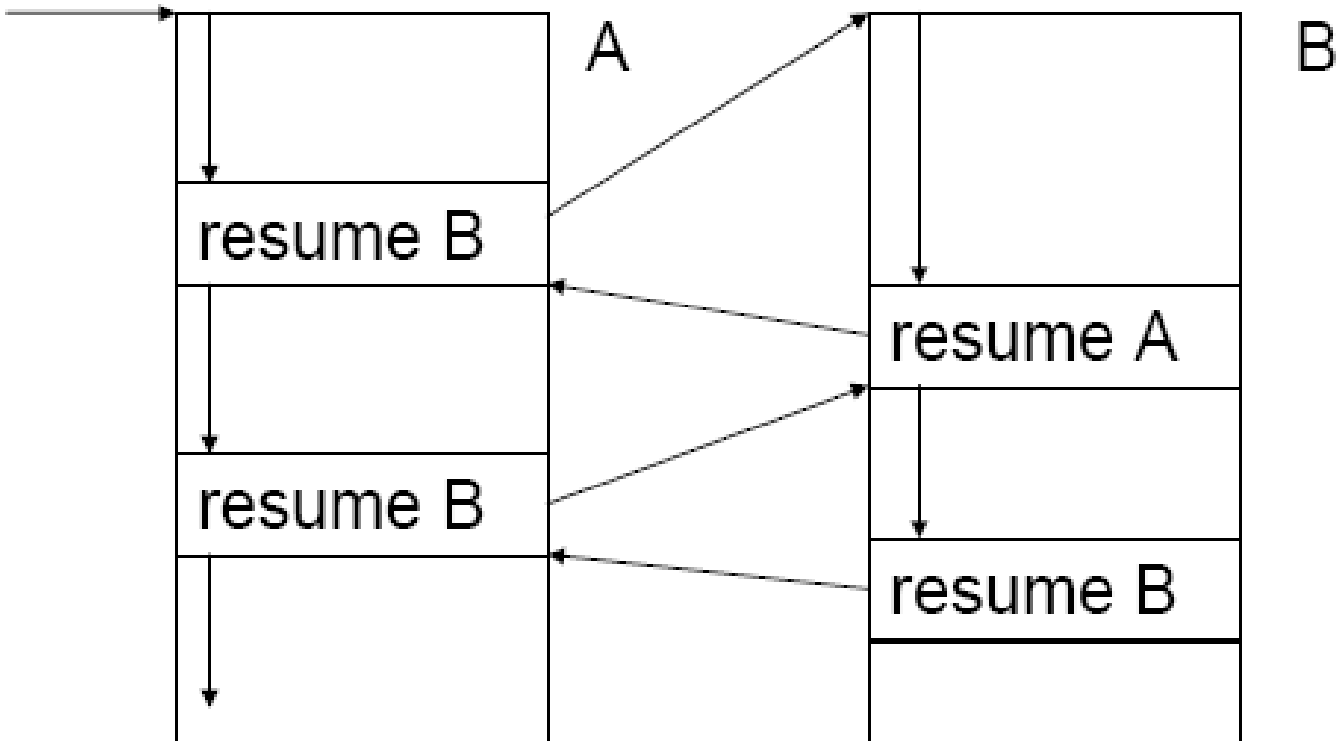
- Lỗi trong thời gian thực thi
  - Chia cho 0
  - Sai kiểu
- Sự kiện không dự đoán trước
  - Phím Esc được nhấn
- Được tạo ra khi thỏa mãn điều kiện

LAN TRUYỀN BIẾN CỐ

- Khi trình xử lý biến cố không thuộc cùng 1 chương trình con với nơi xảy ra biến cố ⇒ lan truyền biến cố
- Biến cố thường được lan truyền theo chuỗi xích động
- Khi trình xử lý biến cố thực thi xong thì điều khiển có thể chuyển về
  - Nơi phát sinh biến cố
  - Kết thúc chương trình
  - Nơi chứa trình xử lý biến cố

TRÌNH CỘNG HÀNH

- Trình cộng hành là các chương trình con luân phiên thực hiện từng phần của chúng



TRÌNH ĐƯỢC ĐỊNH THỜI

- Chương trình con được định thời là chương trình con mà sự thực thi của nó có thể không bắt đầu ngay khi được gọi
  - call A at time = current_time + 10
  - call B with priority 7
- Một chương trình định thời sẽ điều khiển việc thực thi

CÔNG TÁC VÀ SỰ THỰC THI ĐỒNG THỜI

- Công tác (task) là chương trình con có thể thực thi đồng thời với các chương trình con khác
- Đòi hỏi phải đồng bộ hoạt động của các công tác
  - khóa chết
  - chờ đợi vô thời hạn