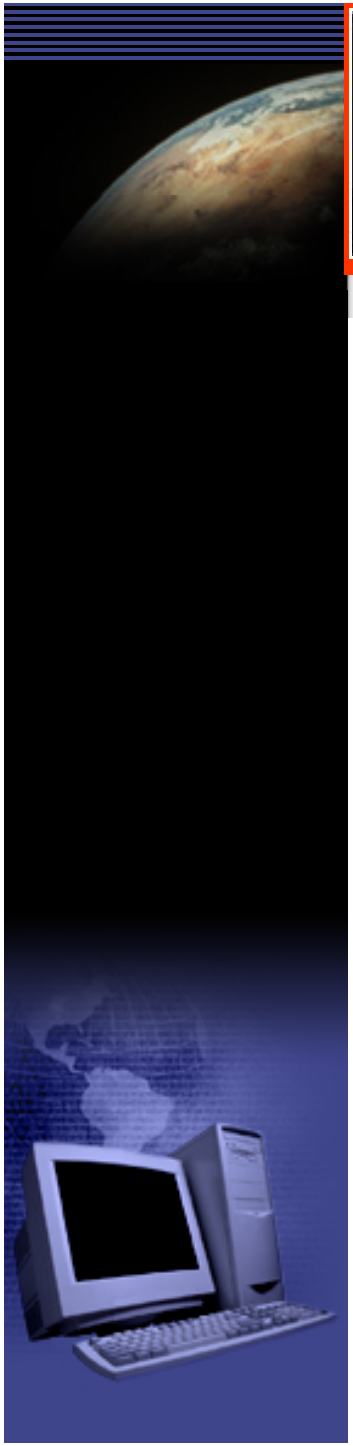
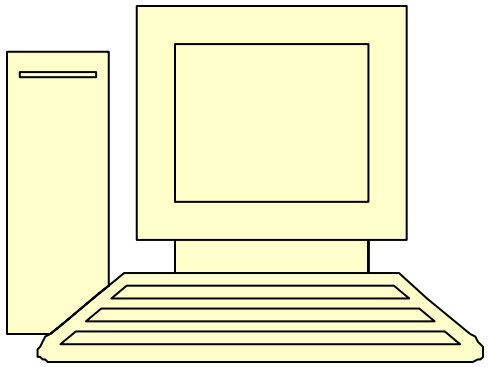



# ĐẠI HỌC ĐÀ NẴNG

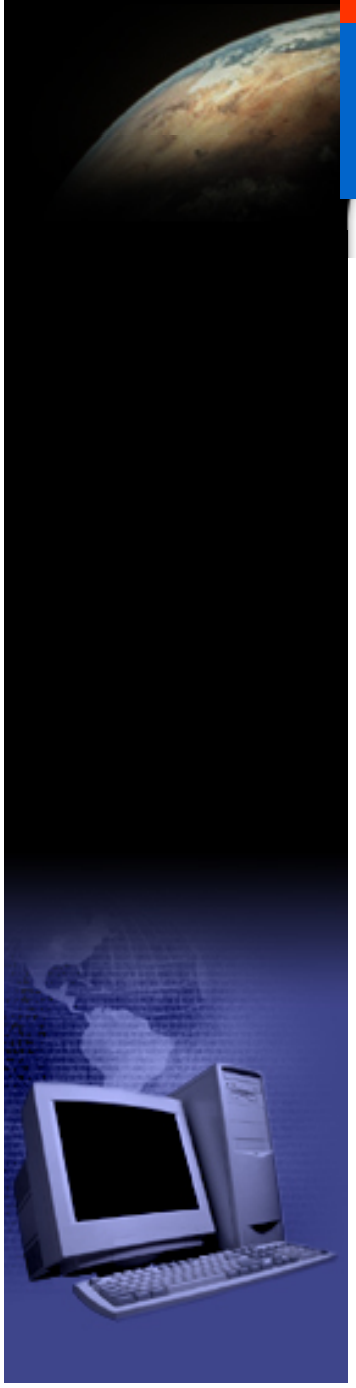
## TRƯỜNG ĐẠI HỌC BÁCH KHOA

## KHOA CÔNG NGHỆ THÔNG TIN

# NGUYÊN LÝ HỆ ĐIỀU HÀNH



# Giới thiệu



## Nội dung giáo trình

**CHƯƠNG 1. MỞ ĐẦU**

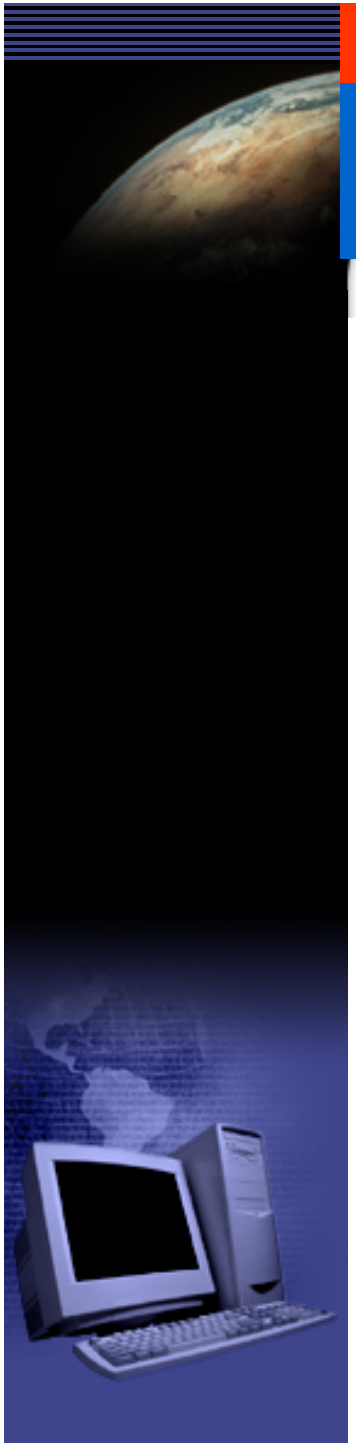
**CHƯƠNG 2. TIẾN TRÌNH**

**CHƯƠNG 3. VÀO/RA**

**CHƯƠNG 4. QUẢN LÝ BỘ NHỚ**

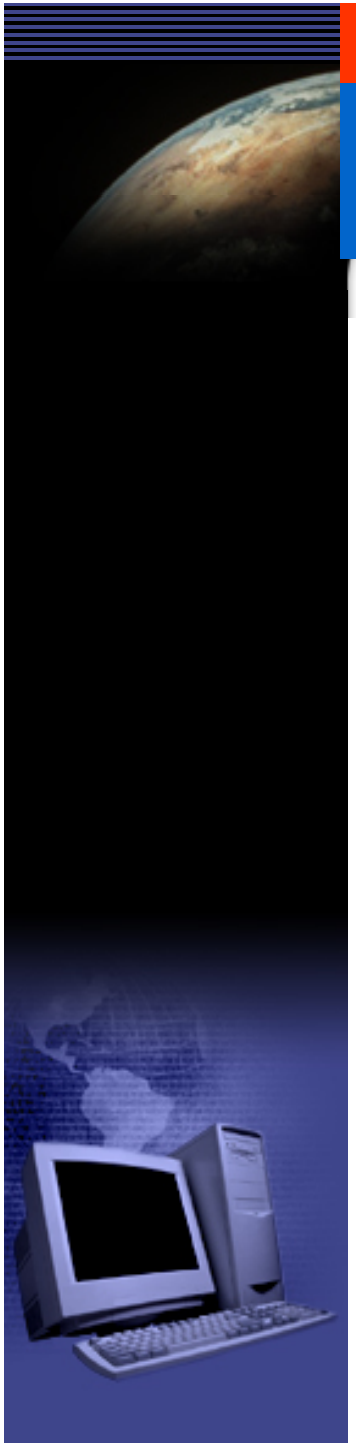
**CHƯƠNG 5. HỆ THỐNG FILE**

## CHƯƠNG 1. MỞ ĐẦU



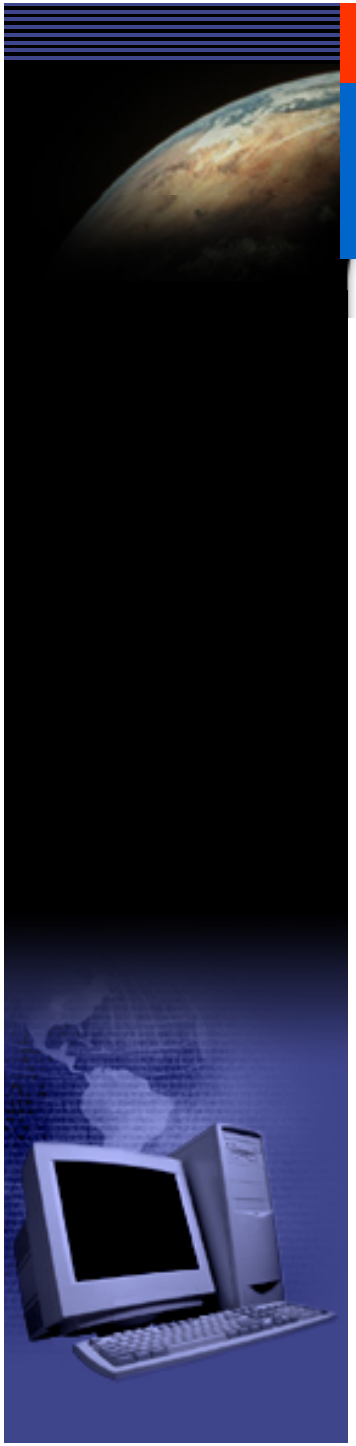
# Các vấn đề

1. Khái niệm hệ điều hành
2. Chức năng của hệ điều hành
3. Vị trí của hệ điều hành
4. Các thành phần của hệ điều hành
5. Cấu trúc của hệ điều hành



# Khái niệm hệ điều hành

**Hệ điều hành (HĐH) là phần gắn bó trực tiếp với phần cứng và là môi trường cho các chương trình ứng dụng chạy trên nó.**



# Chức năng của hệ điều hành

- Quản lý và phân phối tài nguyên 1 cách hợp lý
- Giả lập một máy tính mở rộng và tạo giao diên tiện lợi với người sử dụng

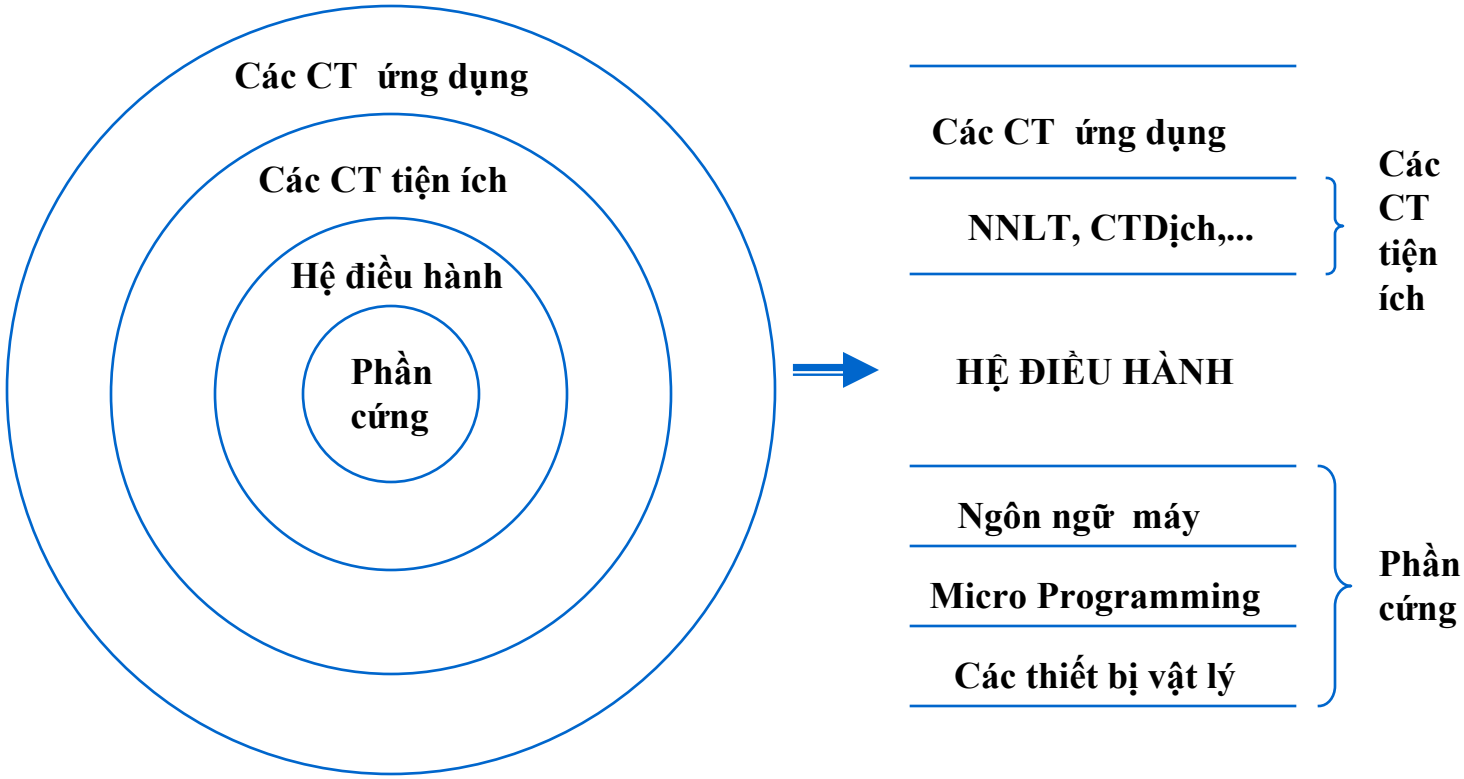
# Tài nguyên

- **Tài nguyên phần cứng**
  - **Bộ xử lý**
  - **Bộ nhớ**
  - **Các thiết bị nhập xuất**
- **Tài nguyên phần mềm**

**Các file, chương trình dùng chung,...**

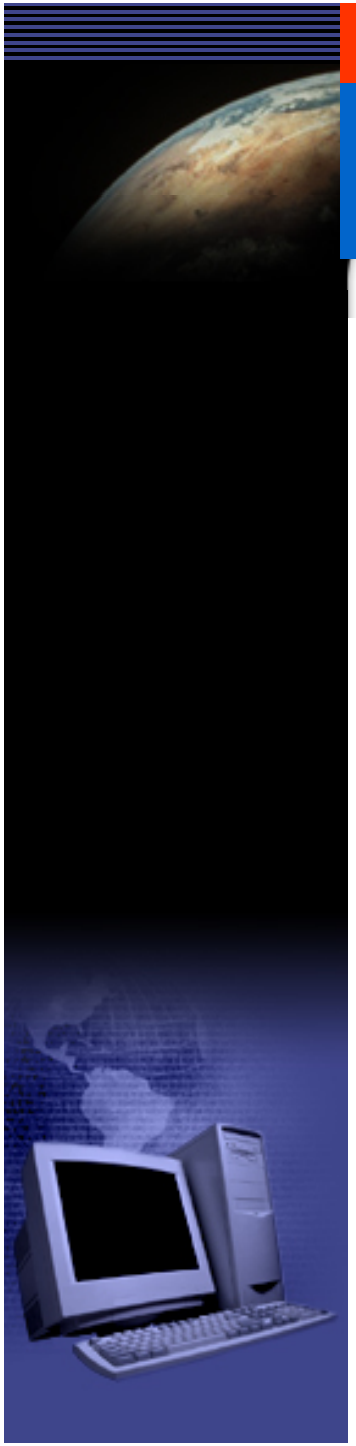
## CHƯƠNG 1. MỞ ĐẦU

# Vị trí của hệ điều hành



# Các thành phần của hệ điều hành

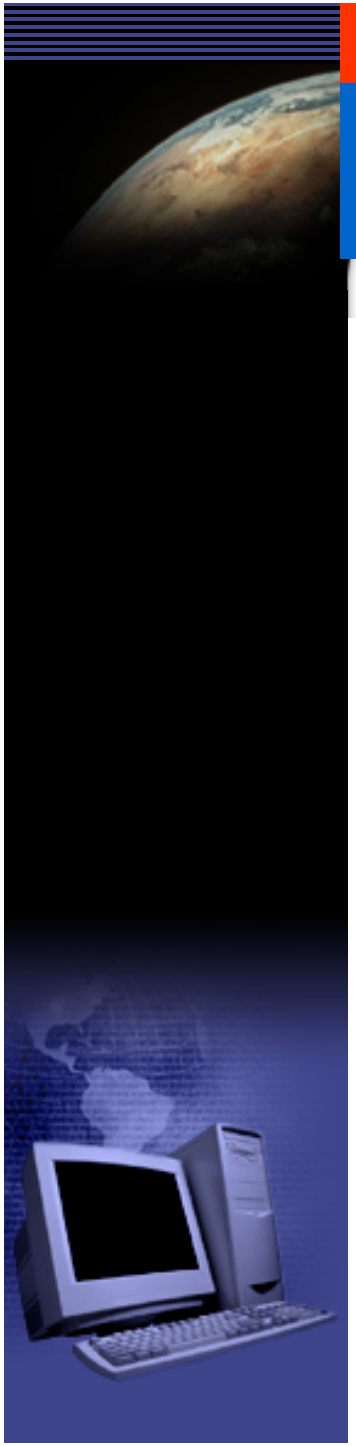
- Quản lý tiến trình
- Quản lý bộ nhớ
- Quản lý nhập xuất
- Quản lý tập tin
- Hệ thống bảo vệ
- Hệ thông dịch lệnh (Shell)
- Quản lý mạng



# Các thành phần của hệ điều hành

## Quản lý tiến trình

- Tạo lập, huỷ bỏ một tiến trình
- Tạm dừng, tái kích hoạt một tiến trình
- Cung cấp các cơ chế trao đổi thông tin giữa các tiến trình
- Cung cấp cơ chế đồng bộ hoá các tiến trình



# Các thành phần của hệ điều hành

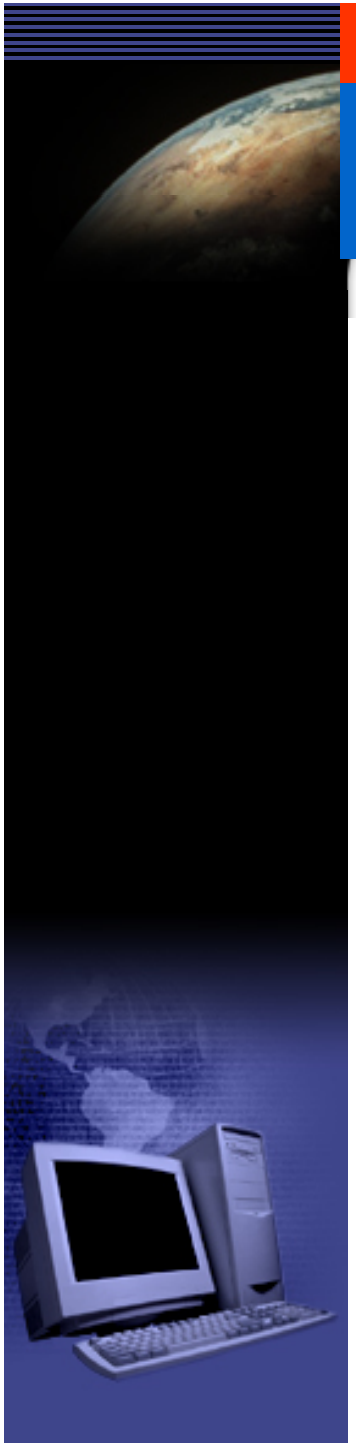
## Quản lý bộ nhớ

- Cấp phát và thu hồi vùng nhớ cho tiến trình khi cần thiết
- Ghi nhận tình trạng bộ nhớ chính: vùng đã cấp phát, vùng còn có thể sử dụng...
- Quyết định tiến trình nào được nạp vào bộ nhớ chính khi có một vùng nhớ trống.

# Các thành phần của hệ điều hành

## Quản lý nhập xuất

- Gửi các lệnh điều khiển đến các thiết bị
- Tiếp nhận các ngắt
- Xử lý lỗi



# Các thành phần của hệ điều hành

## Quản lý tập tin

- Tạo lập, huỷ bỏ một tập tin.
- Tạo lập và huỷ bỏ một thư mục.
- Cung cấp các thao tác xử lý tập tin và thư mục.
- Tạo lập quan hệ tương ứng giữa tập tin và bộ nhớ phụ chứa nó.

# Các thành phần của hệ điều hành

## Hệ thống bảo vệ

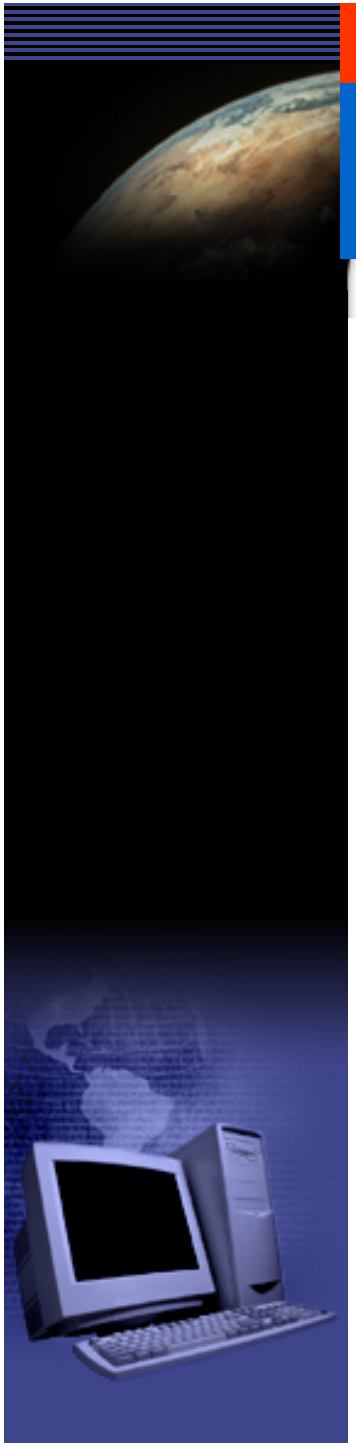
- **Xây dựng cơ chế bảo vệ thích hợp.**

**Trong trường hợp nhiều người cùng sử dụng đồng thời các tiến trình.**

# Các thành phần của hệ điều hành

## Hệ thông dịch lệnh (Shell)

- Đóng vai trò giao diện giữa NSD và HĐH
- Các lệnh được chuyển đến HĐH dưới dạng chỉ thị điều khiển.
- Shell nhận lệnh và thông dịch lệnh để HĐH có xử lý tương ứng



# Các thành phần của hệ điều hành

## Quản lý mạng

- Một hệ thống phân bố nhiều bộ xử lý với các bộ nhớ độc lập.
- Các tiến trình trong hệ thống có thể kết nối với nhau qua mạng truyền thông.
- Việc truy xuất đến tài nguyên mạng thông qua các trình điều khiển giao tiếp mạng.

# Cấu trúc của hệ điều hành

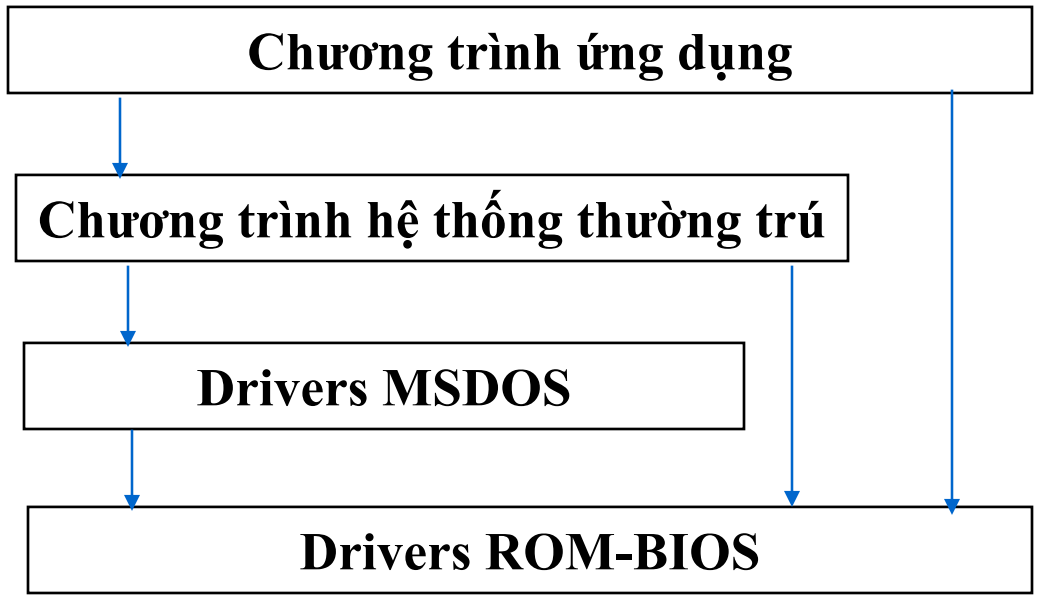
- Hệ thống nguyên khối (Monolithic System)
- Hệ thống phân lớp (Layer System)
- Máy ảo (Virtual Machine)
- Mô hình Client-Server (Client-Server Model)

# Hệ thống nguyên khối

- Cấu trúc HĐH được xem là ko cấu trúc
- HĐH được xây dựng dựa trên tập hợp các thủ tục riêng lẻ.
- Mỗi thủ tục có thể gọi lẫn nhau khi cần
- CT ứng dụng có thể truy xuất đến thủ tục cấp thấp, phần cứng. Do vậy HĐH khó kiểm soát và bảo vệ hệ thống
- Khi xây dựng thủ tục phải định nghĩa rõ tham số đầu vào, tham số đầu ra
- HĐH thiếu tính chủ động trong việc quản lý môi trường. (tính chất tĩnh, chỉ được kích hoạt khi cần)

# Hệ thống nguyên khối

- Ví dụ: Cấu trúc MSDOS



# Hệ thống nguyên khối

- **Hoạt động của bộ xử lý được chia làm 2 chế độ**

**- Chế độ Kernel: chạy thực hiện các thủ tục của HĐH (lời gọi hệ thống)**

**- Chế độ User: chạy thực hiện các CT của NSD**

# Hệ thống nguyên khối

- Khi HĐH khởi động tất cả các lời gọi hệ thống đều được nạp và định vị vào RAM.
- HĐH tạo bảng Dispatch gồm các Slot, mỗi Slot là một con trỏ trỏ đến Đ/C đầu của một CT phục vụ

# Hệ thống phân lớp

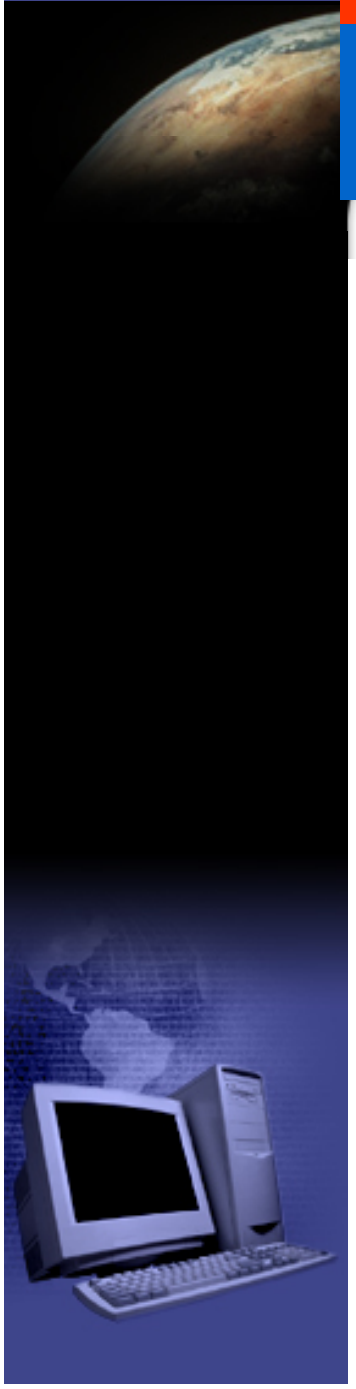
- Hệ thống được xây dựng bởi nhiều lớp.
- Mỗi lớp được xây dựng dựa trên các lớp bên trong
- Lớp trong cùng (lớp 0): phần cứng
- Lớp ngoài cùng (lớp N): giao diện với NSD
- Mỗi lớp là một đối tượng trừu tượng (dữ liệu+thao tác xử lý dữ liệu).
- Mỗi lớp có thể gọi các thủ tục của các lớp bên trong

# Hệ thống phân lớp

❖ **Ví dụ: hệ thống THE (Technische Hogeschool Eidhoven) thiết kế năm 1968**

| |
|---|
| Lớp 5: Chương trình ứng dụng |
| Lớp 4: Quản lý bộ đệm cho thiết bị nhập/xuất |
| Lớp 3: Trình điều khiển thao tác console |
| Lớp 2: Quản lý bộ nhớ |
| Lớp 1: Điều phối CPU |
| Lớp 0: Phần cứng |

## CHƯƠNG 1. MỞ ĐẦU

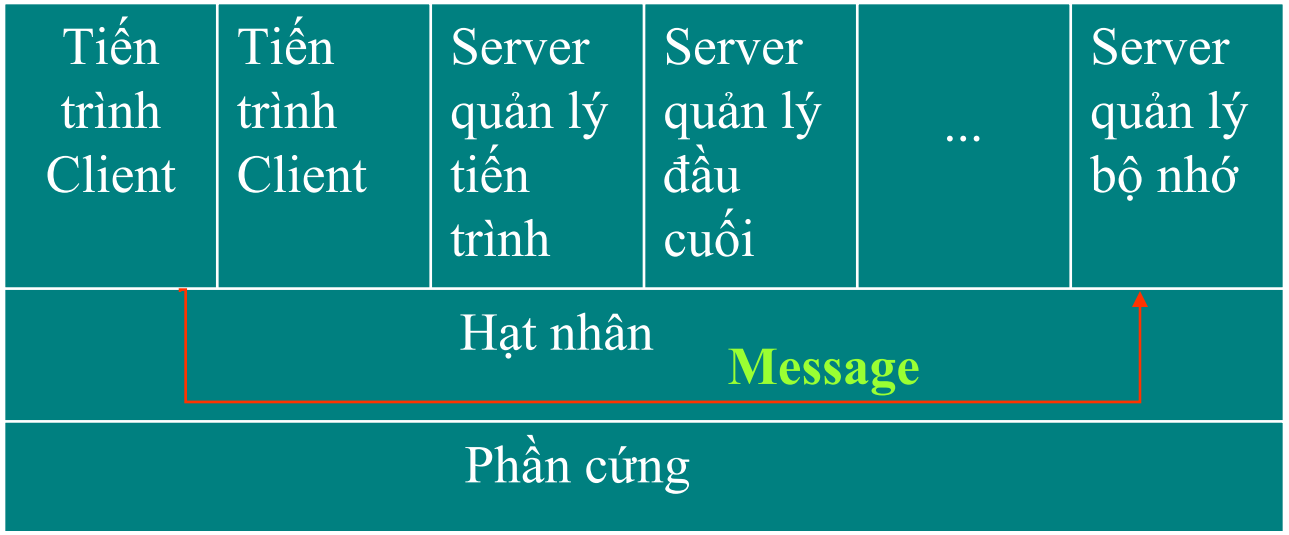


# Máy ảo

# Mô hình Client-Server

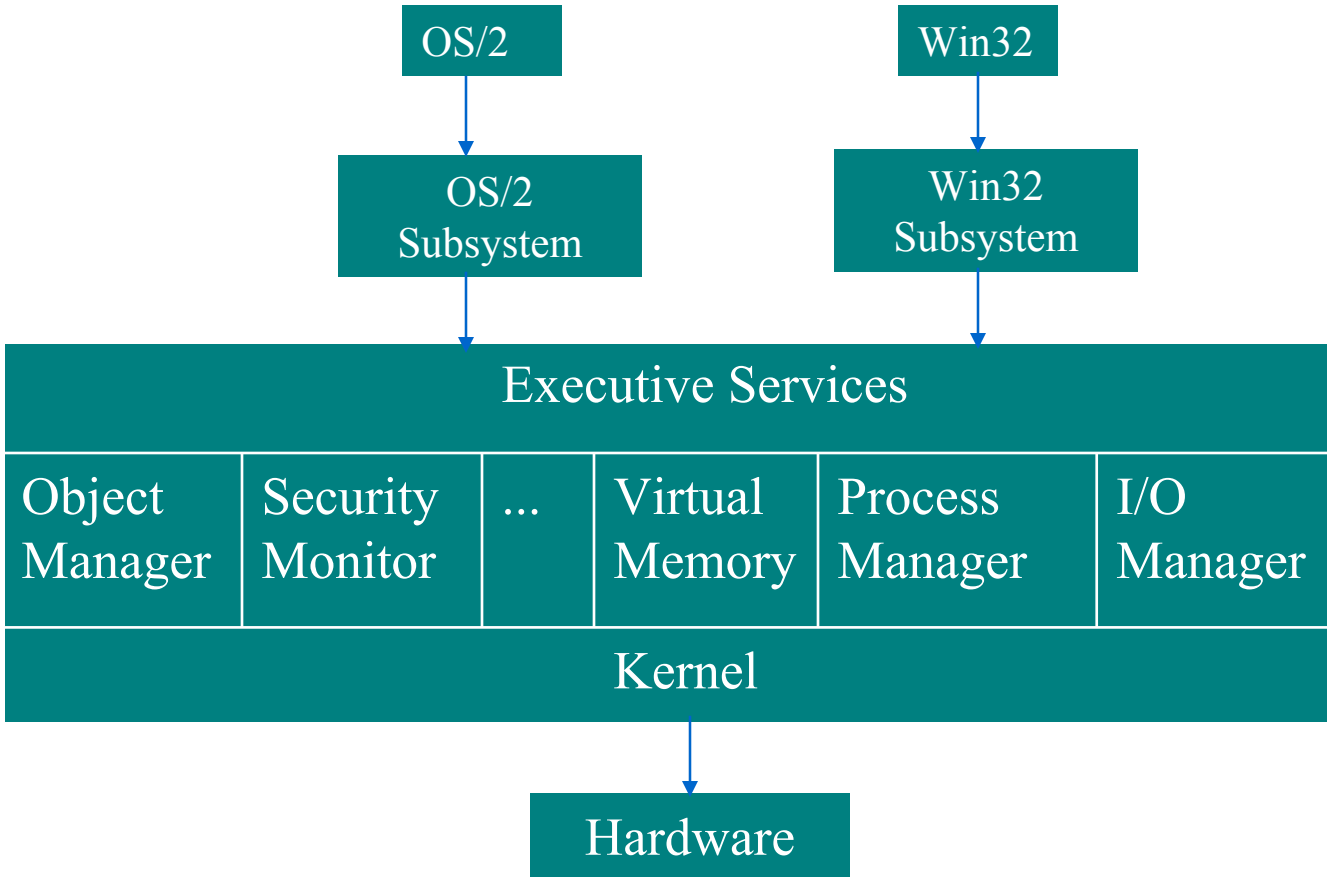
- **HĐH bao gồm nhiều tiến trình đóng vai trò Server với các chức năng chuyên biệt.**
- **Phần hạt nhân HĐH đóng vai trò giao tiếp giữa tiến trình Client và tiến trình Server.**
- **Chỉ có phần hạt nhân cực nhỏ phụ thuộc vào phần cứng.**

# Mô hình Client-Server



# Mô hình Client-Server

## Ví dụ: Cấu trúc Windows NT



## CHƯƠNG 2. TIẾN TRÌNH

# Các vấn đề

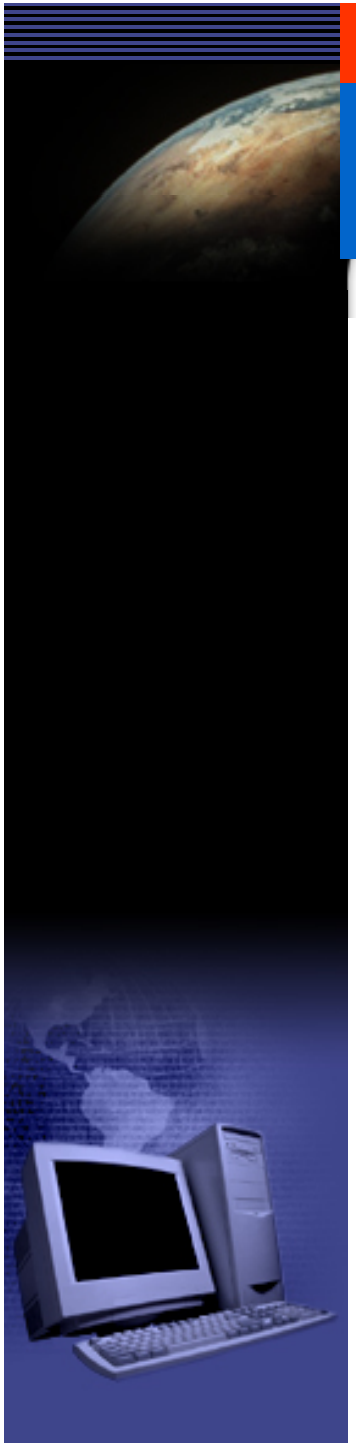
1. Các khái niệm
2. Mô hình trạng thái
3. Thao tác trên tiến trình
4. Điều phối tiến trình
5. Đồng bộ hoá tiến trình

# Các khái niệm

- Tiến trình (Process): chương trình đang thực hiện
- Mỗi tiến trình có một tập tài nguyên và môi trường riêng (con trỏ lệnh, Stack, thanh ghi, không gian địa chỉ)
- Các tiến trình hoàn toàn độc lập với nhau, có thể liên lạc thông qua các cơ chế truyền tin giữa các tiến trình.

# Các khái niệm

- **Tiến trình hệ thống: được sinh ra khi thực hiện các lời gọi hệ thống**
- **Tiến trình của người sử dụng: được sinh ra khi thực thi CT của NSD**



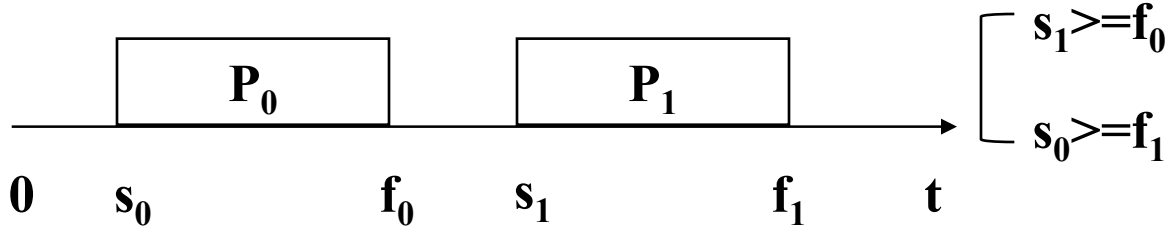
# Các khái niệm

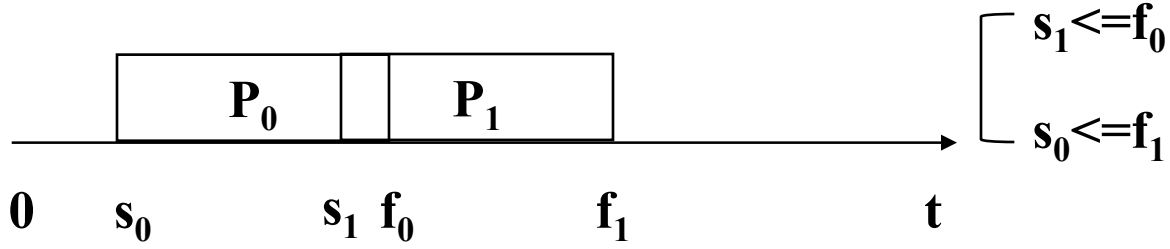
- **Có 2 loại tiến trình:**

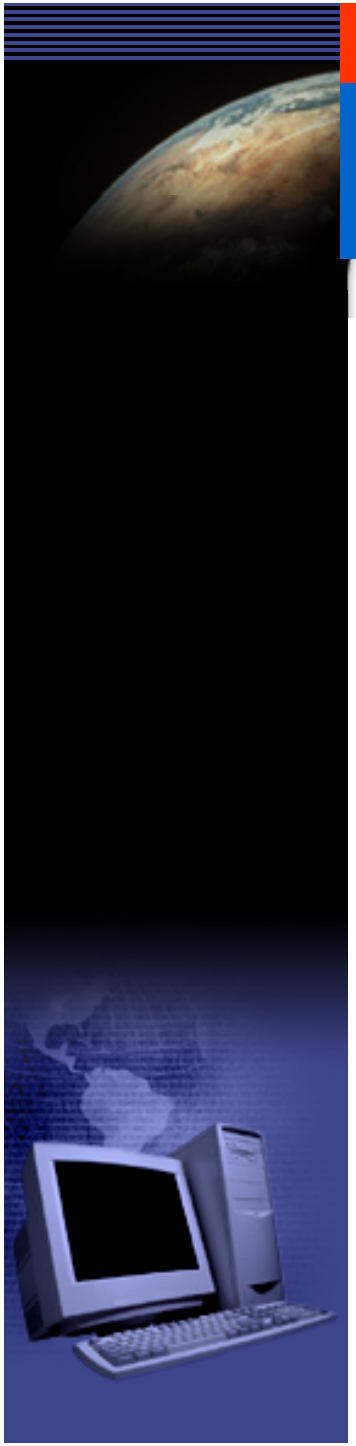
**- Tiến trình kế tiếp: thời điểm bắt đầu của tiến trình này nằm sau thời điểm kết thúc của tiến trình kia**

**- Tiến trình song song: thời điểm bắt đầu của tiến trình này nằm trước thời điểm kết thúc của tiến trình kia**

# Các khái niệm

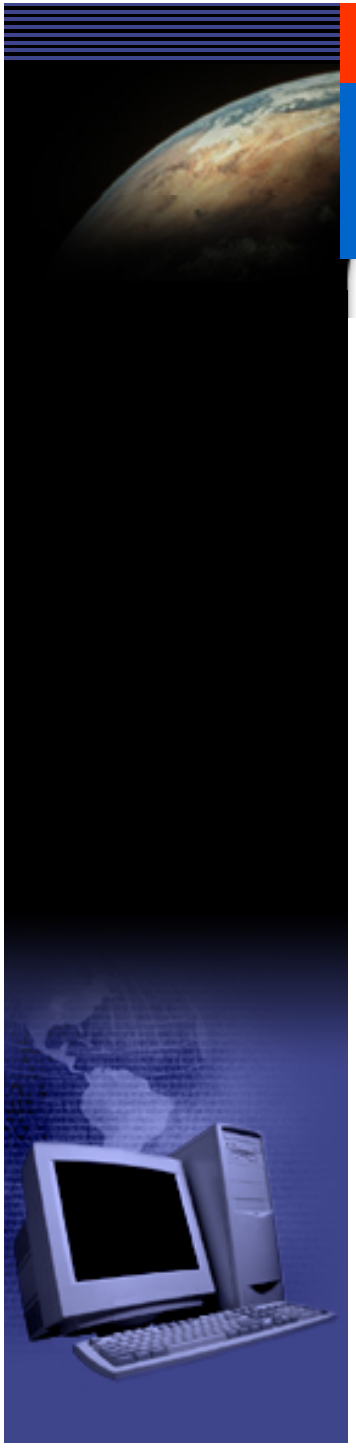




# Các khái niệm

- **HĐH quản lý tiến trình thông qua khối quản lý tiến trình (Process Control Block:PCB)**
- **PCB: vùng nhớ lưu trữ các thông tin mô tả cho tiến trình như:**
  - **Định danh của tiến trình: phân biệt giữa các tiến trình.**
  - **Trạng thái tiến trình: hoạt động hiện hành của tiến trình.**



# Các khái niệm

- **Ngữ cảnh của tiến trình:**

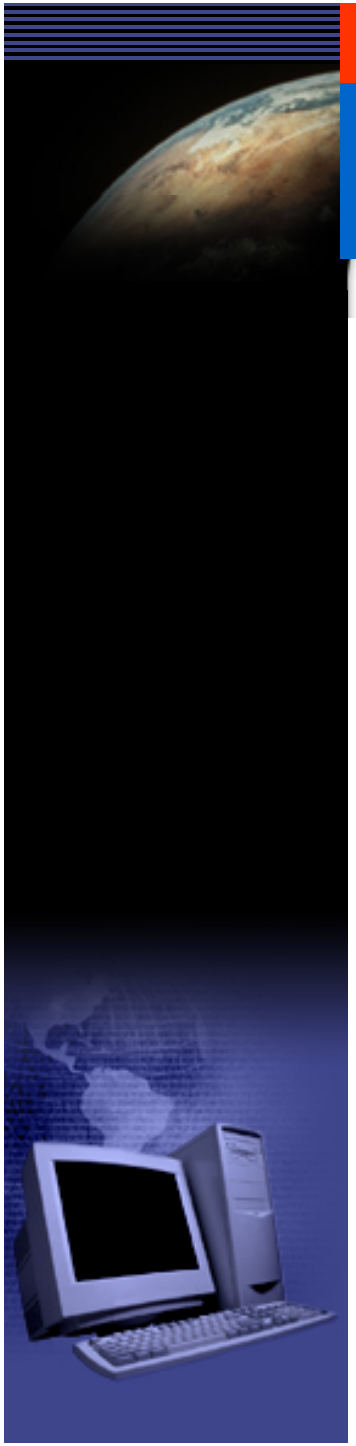
**- Trạng thái CPU: nội dung các thanh ghi (IP). Lưu trữ nội dung thanh ghi khi xảy ra ngắt.**

**- Bộ xử lý: xác định số hiệu CPU mà tiến trình đang sử dụng (máy có cấu hình nhiều CPU).**

**- Bộ nhớ chính: danh sách các vùng nhớ được cấp cho tiến trình.**

**- Tài nguyên sử dụng: danh sách các tài nguyên hệ thống mà tiến trình đang sử dụng.**

**- Tài nguyên tạo lập: danh sách các tài nguyên được tiến trình tạo lập.**



# Các khái niệm

- **Thông tin giao tiếp:**

**- Tiến trình cha: tiến trình tạo lập tiến trình này**

**- Tiến trình con: các tiến trình do tiến trình này tạo ra**

**- Độ ưu tiên: thông tin giúp bộ điều phối lựa chọn tiến trình được cấp CPU**

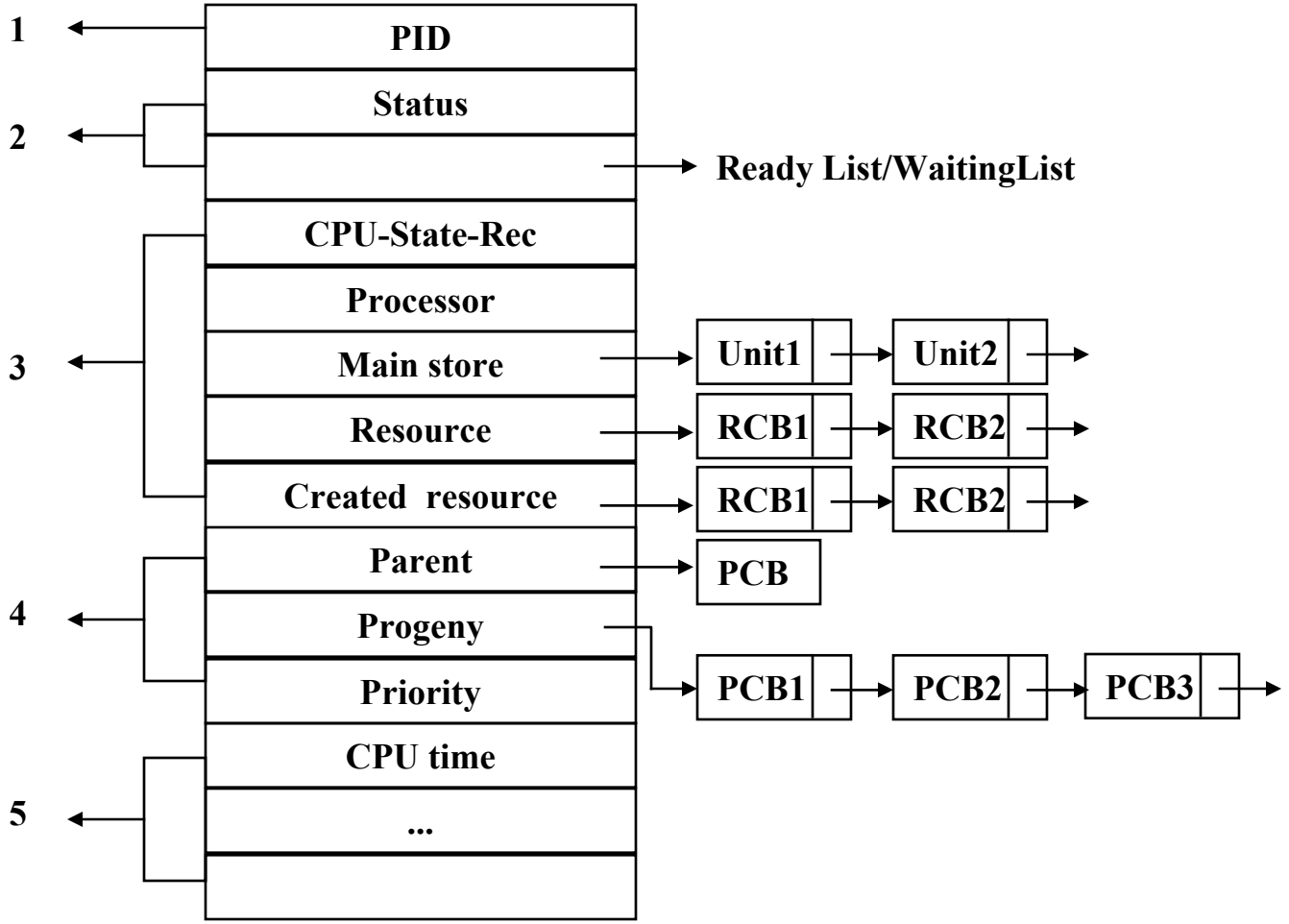
- **Thông tin thống kê về hoạt động của tiến trình:**

**- Thời gian sử dụng CPU**

**- Thời gian chờ**

# Các khái niệm

## Khối quản lý tiến trình



# Các khái niệm

- Tiểu trình (Threads): một đơn vị xử lý cơ bản của hệ thống.
- Một tiểu trình cũng có thể tạo lập các tiến trình con
- Một tiến trình có thể sở hữu nhiều tiểu trình
- Các tiểu trình trong cùng một tiến trình có thể:
  - Chia sẻ một không gian địa chỉ.
  - Truy xuất đến các Stack của nhau

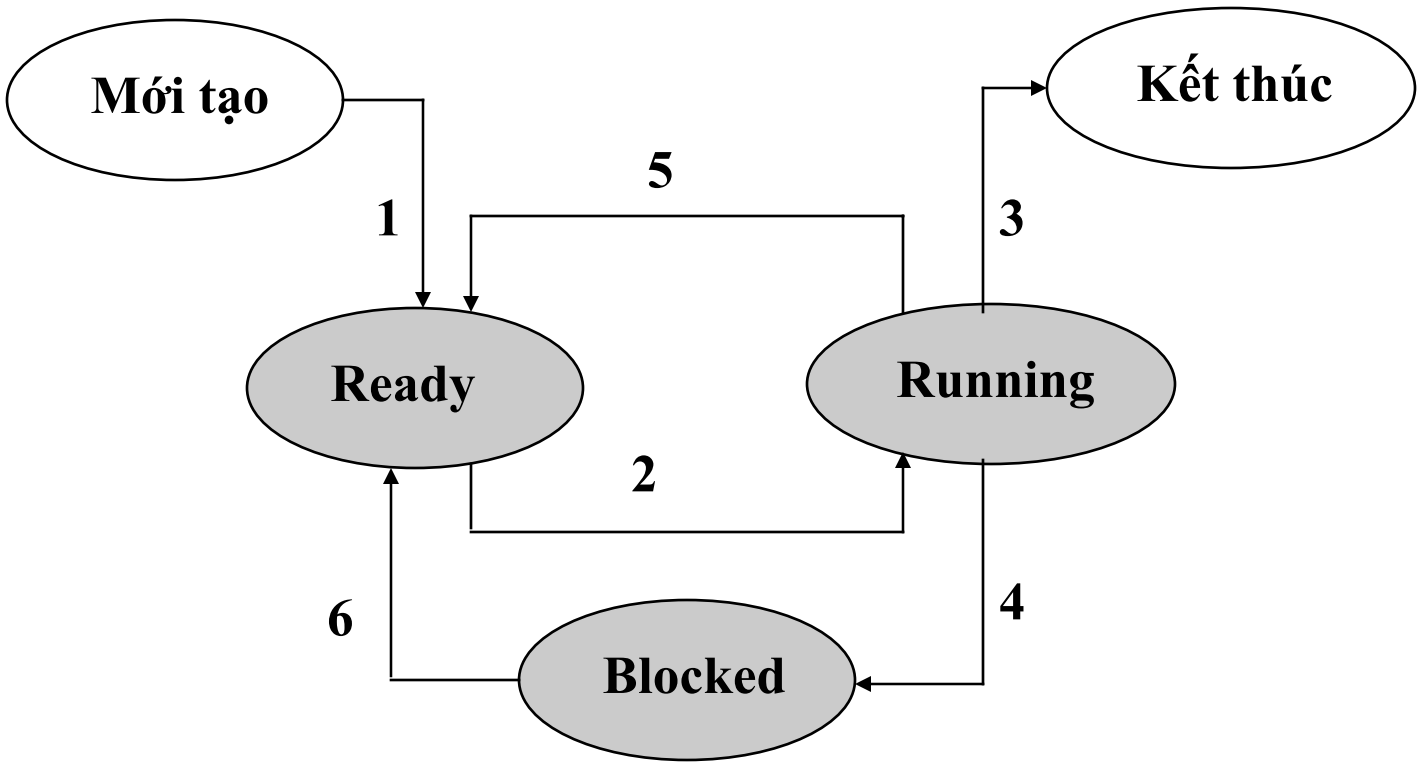
# Mô hình trạng thái

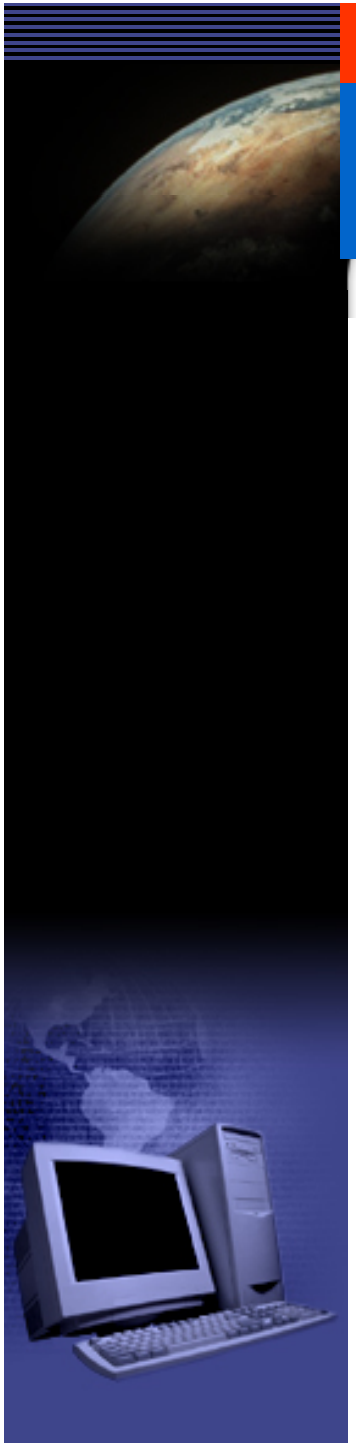
## ❖ Các trạng thái của tiến trình

- **Mới tạo: tiến trình đang được tạo lập.**
- **Running: tiến trình đang được xử lý.**
- **Ready: tiến trình đang sẵn sàng, chờ cấp CPU để xử lý**
- **Blocked: tiến trình bị chặn, không thể tiếp tục.**
- **Kết thúc: tiến trình hoàn tất xử lý.**

# Mô hình trạng thái

❖ Sơ đồ chuyển trạng thái của tiến trình





# Mô hình trạng thái

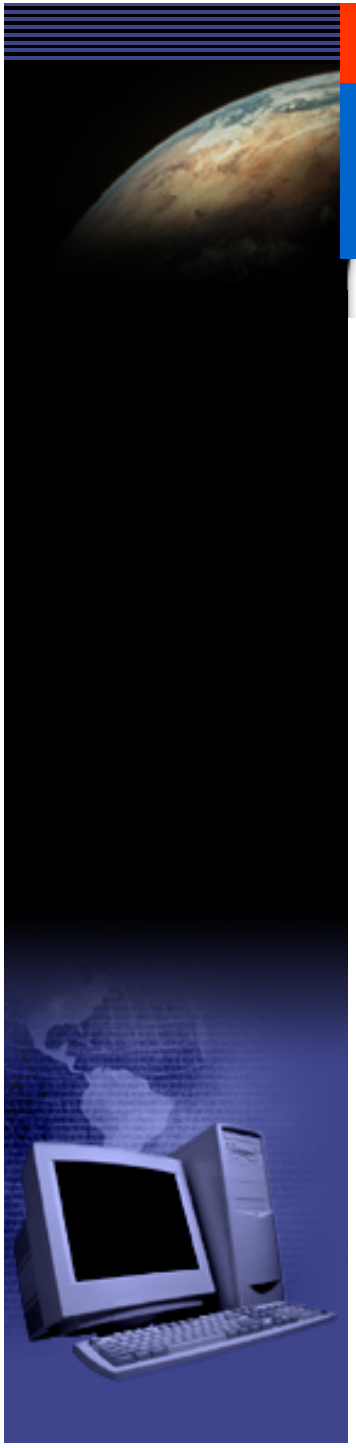
❖ **Sơ đồ chuyển trạng thái của tiến trình**

(1) Tiến trình mới tạo lập được đưa vào hệ thống.

(2) Bộ điều phối cấp phát cho tiến trình một khoảng thời gian sử dụng CPU.

(3) Tiến trình kết thúc, bộ điều phối thu lại CPU.

(4) Tiến trình yêu cầu tài nguyên nhưng chưa được đáp ứng, hoặc phải chờ một sự kiện hay thao tác nhập/xuất.

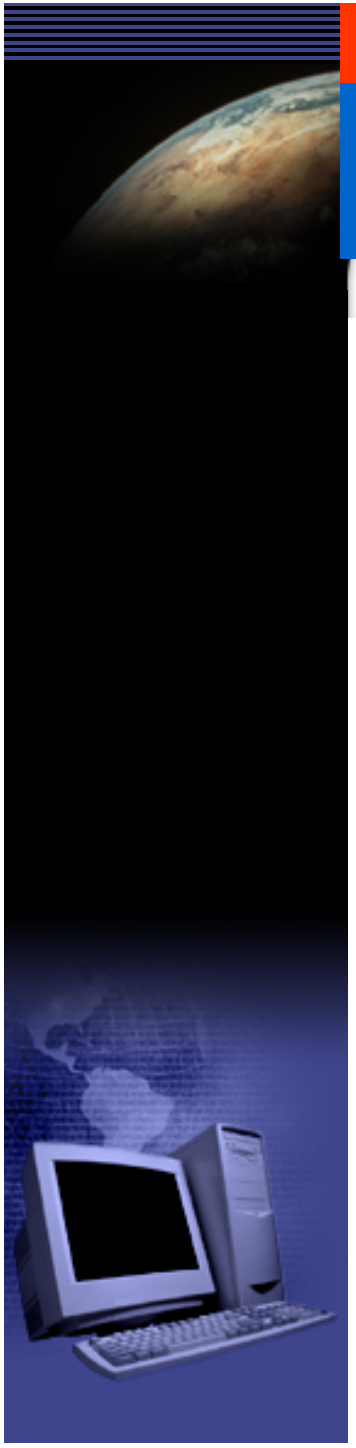


# Mô hình trạng thái

❖ Sơ đồ chuyển trạng thái của tiến trình

(5) Bộ điều phối chọn một tiến trình khác để xử lý.

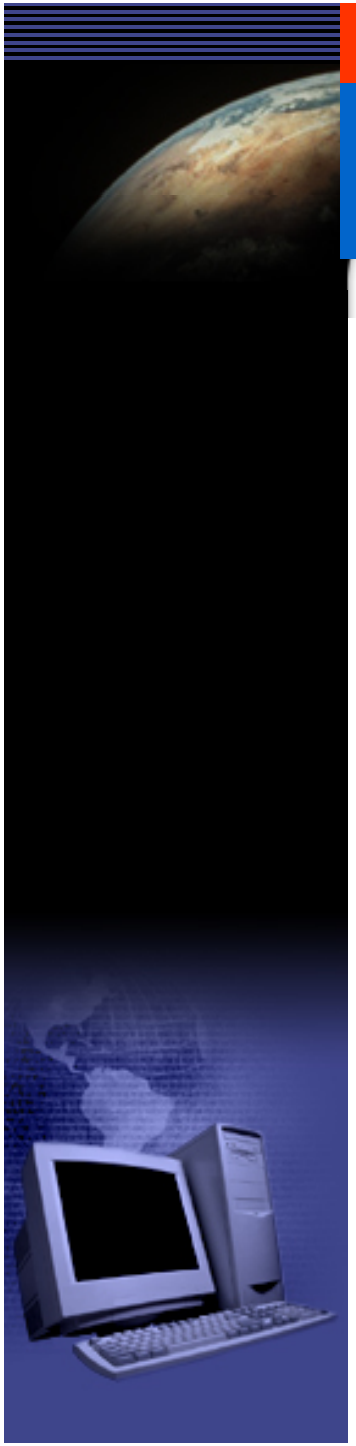
(6) Tài nguyên mà tiến trình yêu cầu đã sẵn sàng để cấp phát, hay sự kiện, thao tác nhập/xuất tiến trình đang đợi hoàn tất.



# Thao tác trên tiến trình

## ❖ Tạo lập tiến trình

- Một tiến trình có thể tạo lập nhiều tiến trình mới
- Tiến trình tạo ra tiến trình mới gọi là tiến trình cha
- Tiến trình mới được tạo ra gọi là tiến trình con
- Tiến trình con đến lượt lại tạo ra một loạt các tiến trình con của nó,... Quá trình này tiếp tục sẽ tạo thành cây tiến trình



# Thao tác trên tiến trình

## ❖ Tạo lập tiến trình

- Khi tạo lập tiến trình,HĐH cần thực hiện:

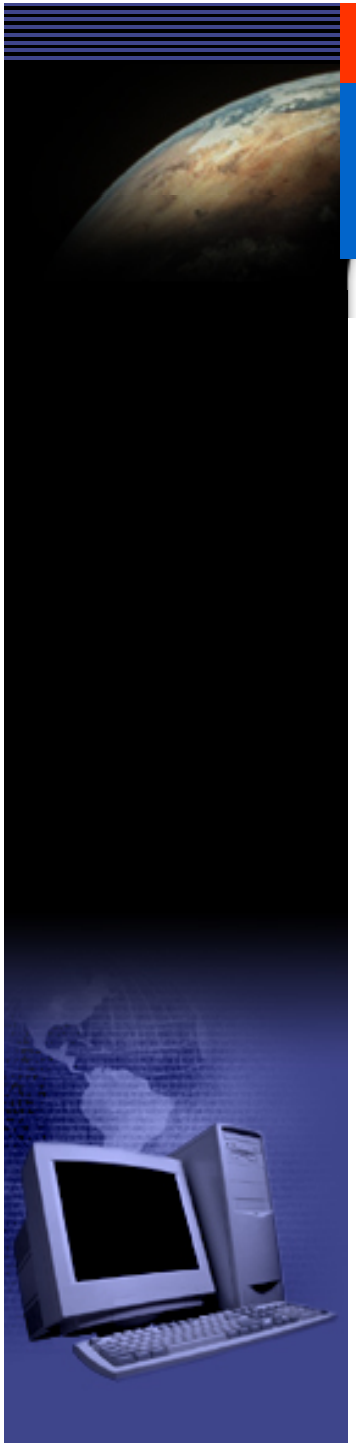
✓ Định danh cho tiến trình (PID)

✓ Đưa tiến trình vào danh sách quản lý của hệ thống

✓ Xác định độ ưu tiên của tiến trình

✓ Tạo khối quản lý tiến trình (PCB)

✓ Cấp phát tài nguyên ban đầu cho tiến trình

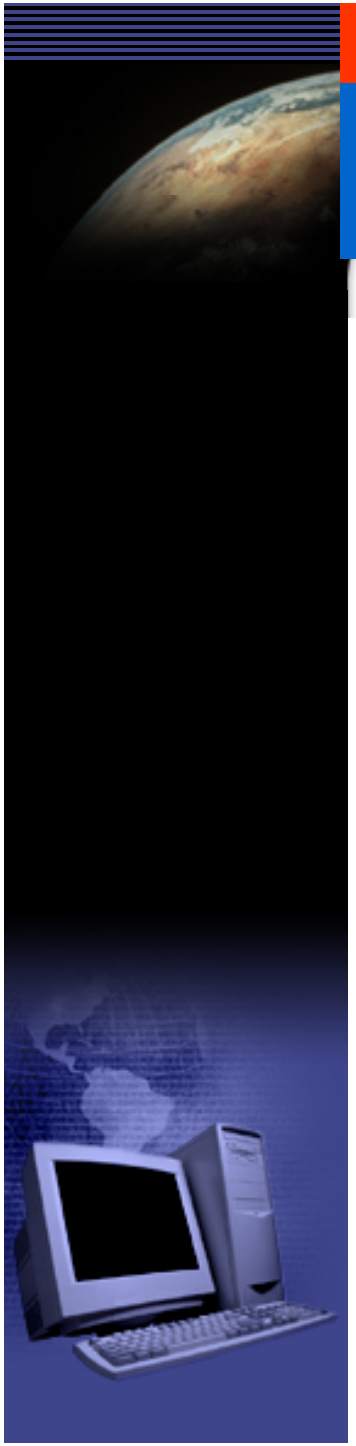


# Thao tác trên tiến trình

## ❖ Kết thúc tiến trình

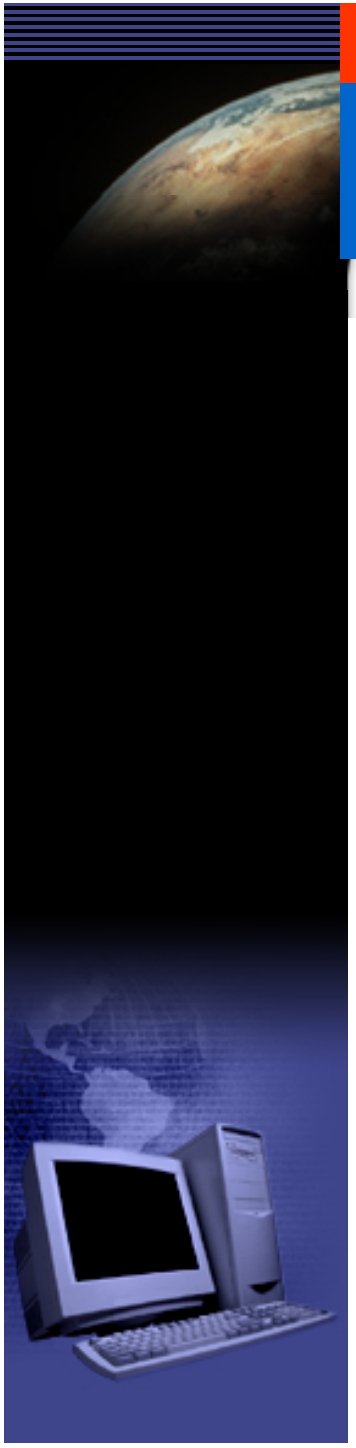
**Khi tiến trình kết thúc, HĐH thực hiện:**

- **Thu hồi các tài nguyên của hệ thống đã cấp phát cho tiến trình**
- **Huỷ tiến trình khỏi tất cả các danh sách quản lý của hệ thống**
- **Huỷ bỏ PCB của tiến trình**



# Điều phối tiến trình

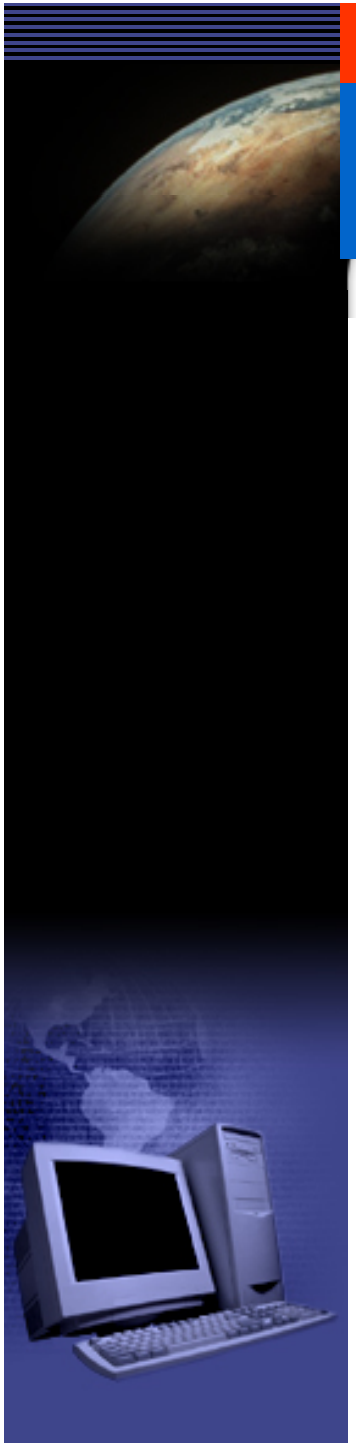
- Mục tiêu điều phối
- Tiêu chuẩn điều phối
- Điều phối không độc quyền, điều phối độc quyền
- Đồng hồ ngắt giờ
- Độ ưu tiên của tiến trình
- Tổ chức điều phối
- Các chiến lược điều phối



# Điều phối tiến trình

## ❖ Mục tiêu điều phối

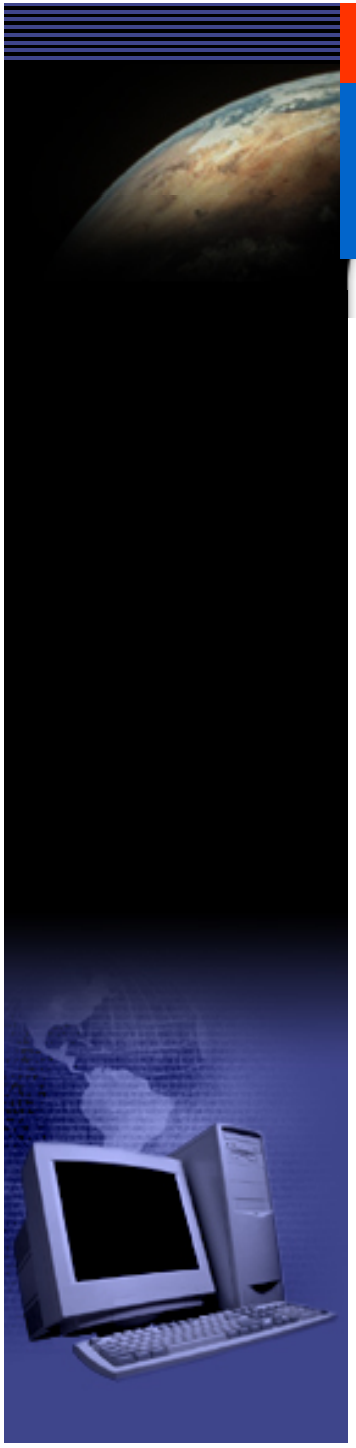
- Sự công bằng giữa các tiến trình
- Tính hiệu quả (tận dụng 100% thời gian sử dụng CPU)
- Cực tiểu hoá thời gian lưu lại trong hệ thống
- Thời gian đáp ứng hợp lý (cực tiểu hoá thời gian hồi đáp cho các tương tác của NSD)
- Thông lượng tối đa (cực đại hoá số công việc được xử lý trong một thời gian cố định)



# Điều phối tiến trình

❖ **Tiêu chuẩn điều phối (đặc điểm của tiến trình)**

- Tính hướng xuất/nhập của tiến trình
- Tính hướng xử lý của tiến trình
- Tiến trình tương tác hay xử lý theo lô
- Độ ưu tiên của tiến trình
- Thời gian đã sử dụng CPU của tiến trình
- Thời gian còn lại tiến trình cần để hoàn tất



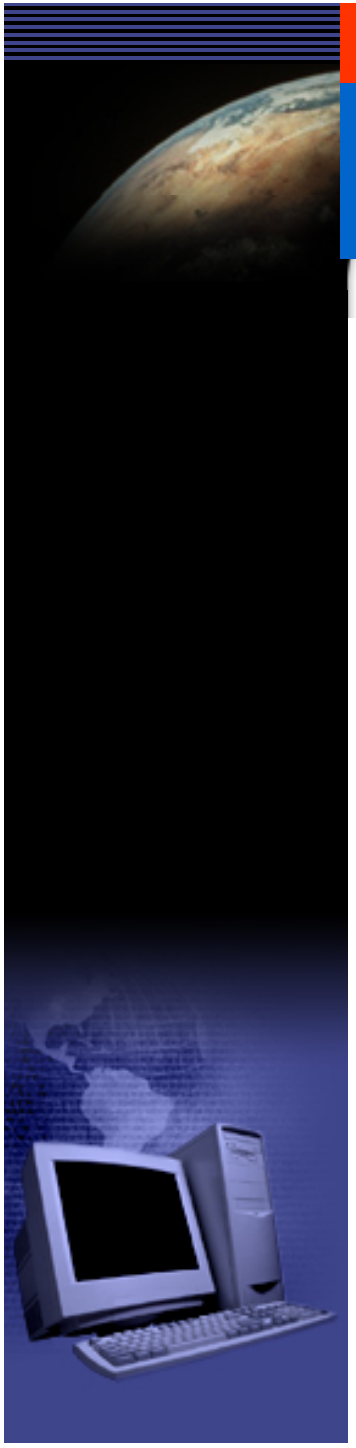
# Điều phối tiến trình

## ❖ Điều phối độc quyền

- Tiến trình khi nhận được CPU thì có độc quyền sử dụng cho đến khi tiến trình hoàn tất hay tự nguyện giải phóng CPU
- Quyết định điều phối CPU xảy ra khi:

  + Tiến trình chuyển từ trạng thái Running sang Blocked

  + Tiến trình kết thúc
- Giải thuật đơn giản, dễ cài đặt nhưng ngăn cản các tiến trình còn lại trong hệ thống có cơ hội để xử lý



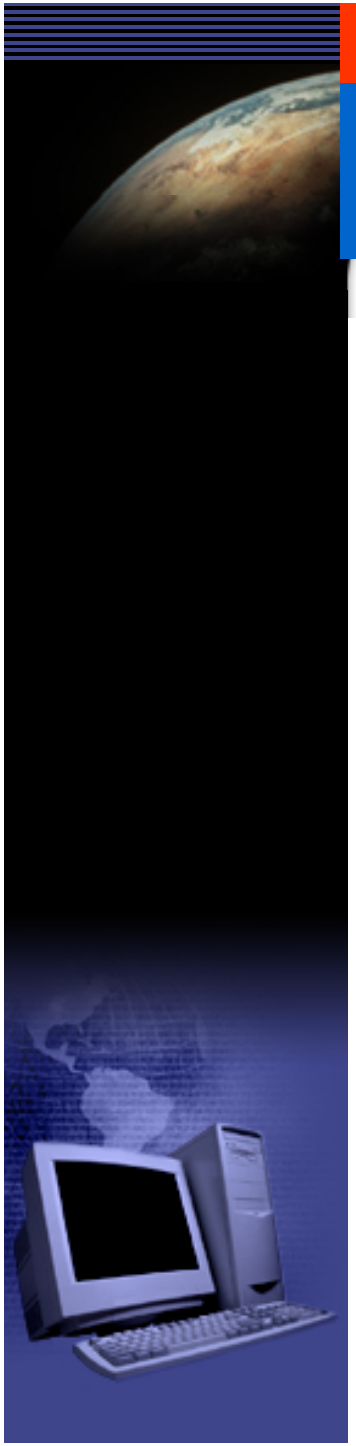
# Điều phối tiến trình

## ❖ Điều phối không độc quyền

- Tiến trình có thể bị tạm dừng hoạt động bất cứ lúc nào mà không được báo trước, để tiến trình khác xử lý. (khi có một tiến trình khác có độ ưu tiên cao hơn về quyền dành sử dụng CPU)
- Quyết định điều phối CPU xảy ra khi:

  + Tiến trình chuyển từ trạng thái Running sang Blocked

  + Tiến trình chuyển từ trạng thái Running sang Ready



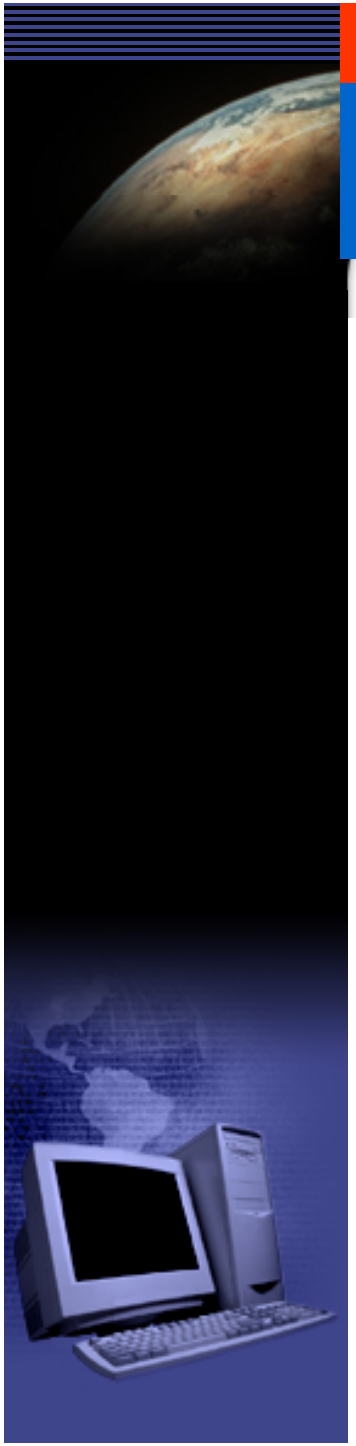
# Điều phối tiến trình

❖ **Điều phối không độc quyền**

**+ Tiến trình chuyển từ trạng thái blocked sang Ready**

**+ Tiến trình kết thúc**

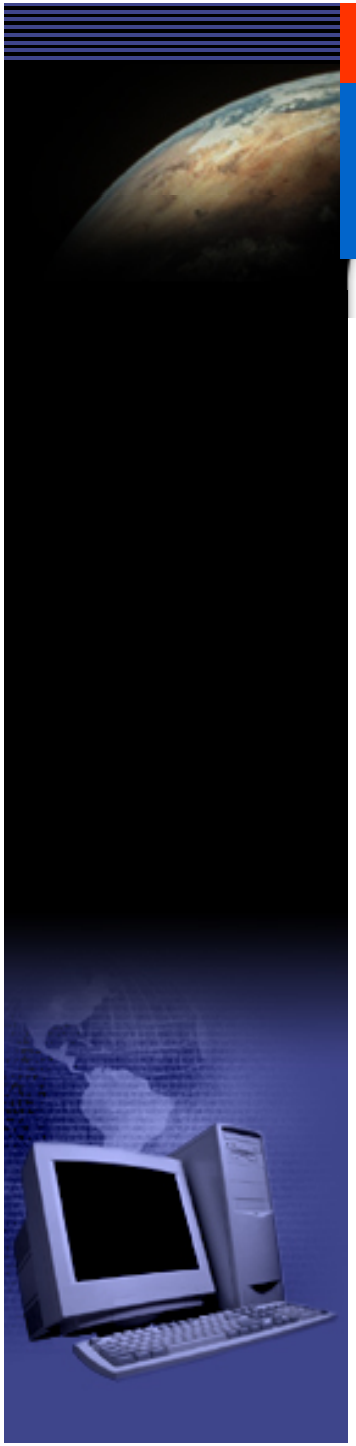
- **Ngăn cản được tình trạng các tiến trình độc chiếm CPU, nhưng việc tam dừng một tiến trình dẫn đến các mâu thuẫn trong truy xuất. Đòi hỏi phương pháp đồng bộ hoá thích hợp**



# Điều phối tiến trình

## ❖ Đồng hồ ngắt thời gian

- Bộ đếm thời gian qui định một thông số thời gian t thích hợp ứng với một lượt cấp CPU cho một tiến trình
- Sau một khoảng thời gian t sẽ xảy ra một ngắt báo hiệu hết thời gian sử dụng CPU của tiến trình hiện hành. HĐH sẽ thu hồi CPU và bộ điều phối sẽ quyết định tiến trình nào sẽ được cấp phát.



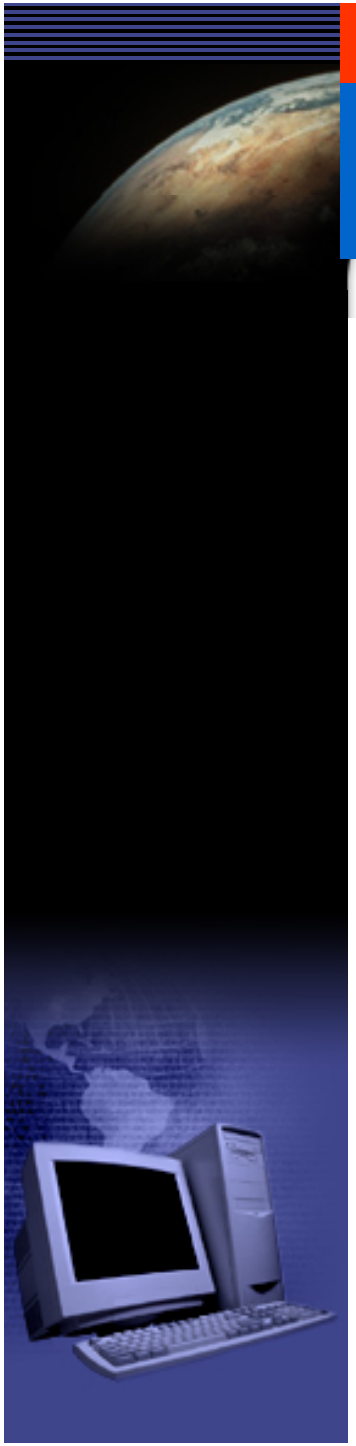
# Điều phối tiến trình

## ❖ Độ ưu tiên của tiến trình

- Độ ưu tiên của tiến trình: giá trị giúp phân định tầm quan trọng của các tiến trình
- Độ ưu tiên tĩnh:

  + Được gán sẵn cho tiến trình khi mới được ta ra

  + Không thay đổi
- Độ ưu tiên động: thay đổi theo thời gian và môi trường xử lý của tiến trình



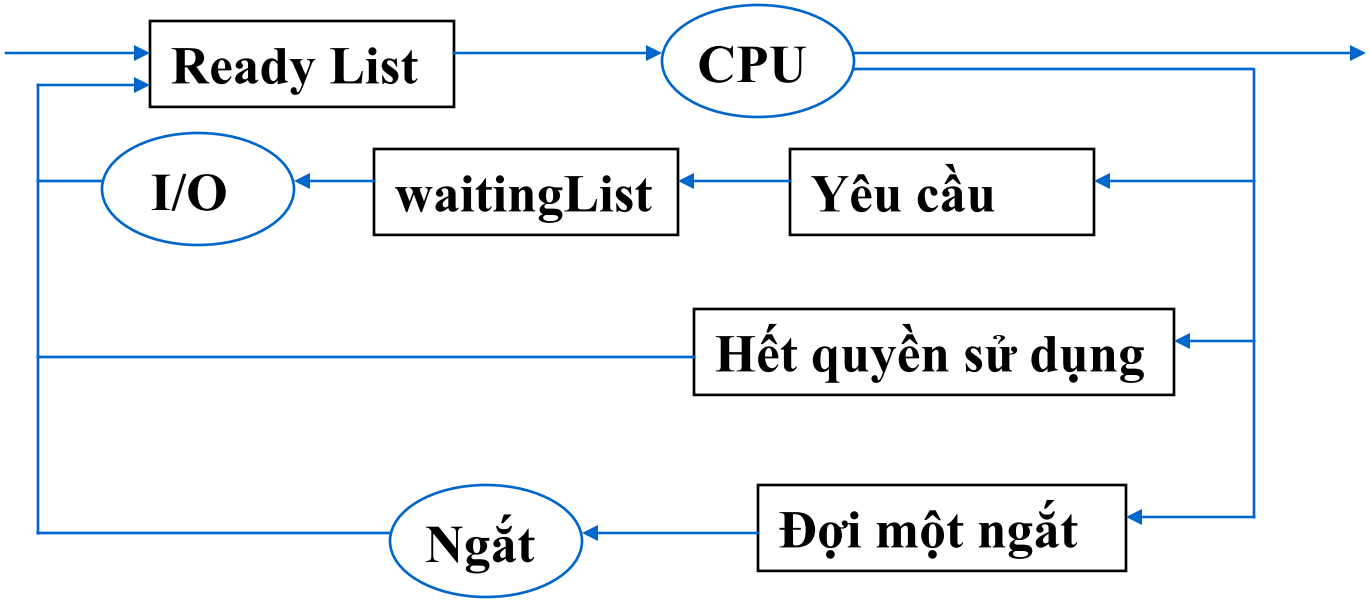
# Điều phối tiến trình

## ❖ Tổ chức điều phối

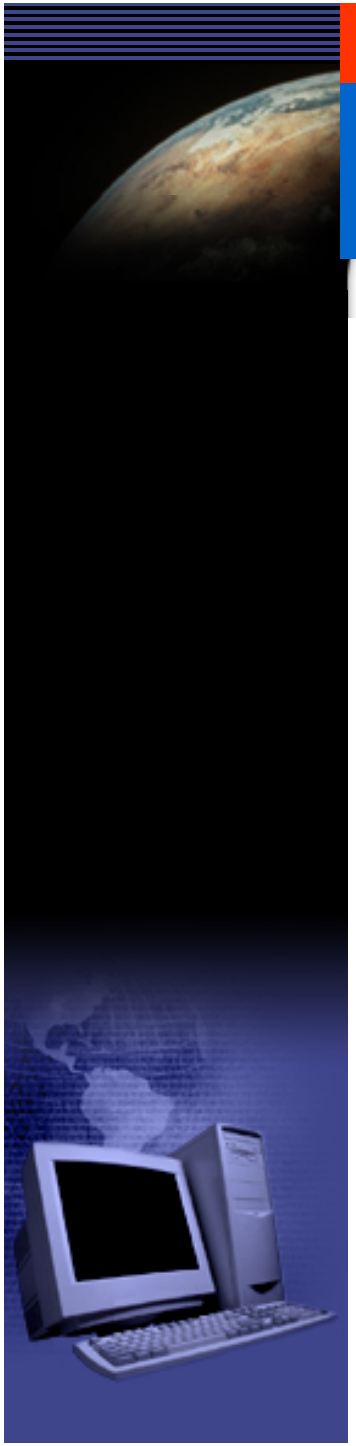
- Danh sách sẵn sàng (Ready List)
- Danh sách chờ đợi (Waiting List)
- Các danh sách chờ đợi riêng cho từng tài nguyên (thiết bị ngoại vi)

# Điều phối tiến trình

## ❖ Tổ chức điều phối




**Sơ đồ chuyển đổi giữa các danh sách điều phối**



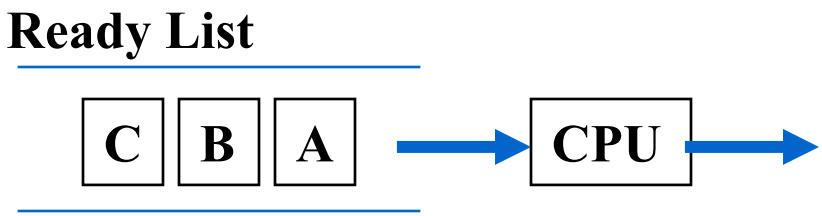
# Điều phối tiến trình

## ❖ Chiến lược điều phối

- Thuật toán FIFO
- Thuật toán Round Robin (xoay vòng)
- Thuật toán SJF (Shortest-Job-First)
- Thuật toán sử dụng độ ưu tiên

# Điều phối tiến trình

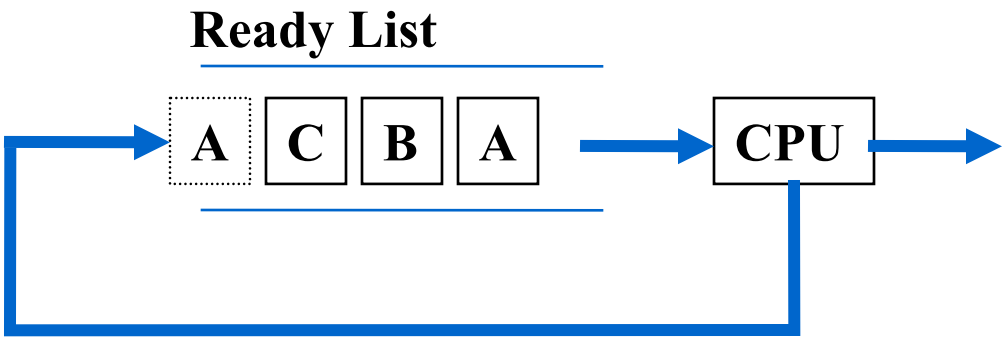
## ❖ Chiến lược điều phối



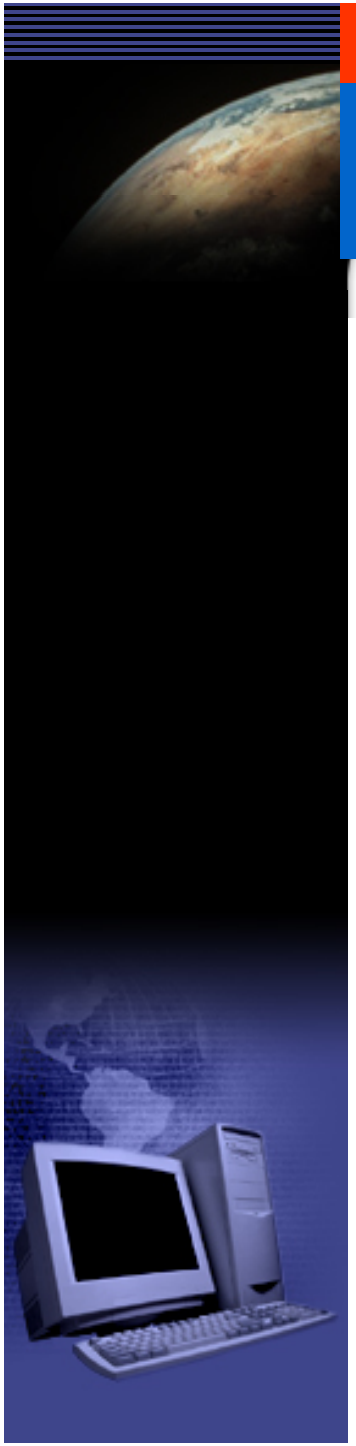

**Điều phối FIFO**

# Điều phối tiến trình

## ❖ Chiến lược điều phối



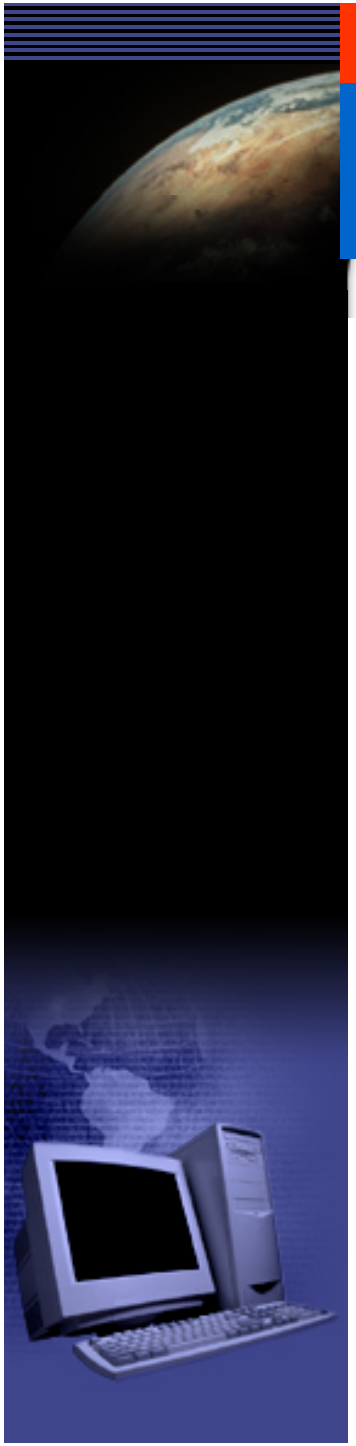

**Điều phối Round Robin**



# Điều phối tiến trình

## ❖ Tổ chức điều phối

- Danh sách sẵn sàng (Ready List)
- Danh sách chờ (Waiting List)
- Các danh sách chờ riêng cho từng tài nguyên (thiết bị ngoại vi)



# Đồng bộ hoá tiến trình

## ❖ Nhu cầu đồng bộ hoá

- Yêu cầu truy xuất độc quyền
- Yêu cầu phối hợp

# Đồng bộ hoá tiến trình

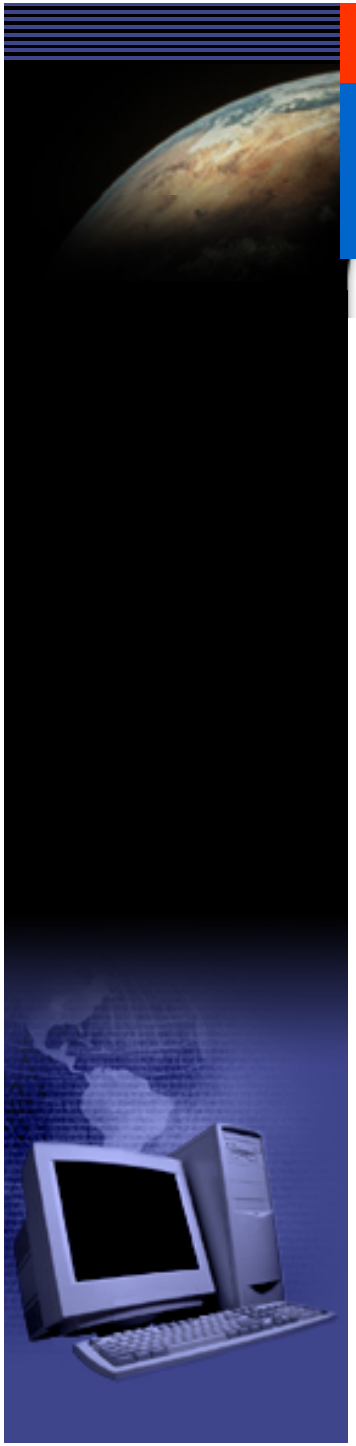
## ❖ Miền găng (Critical Section)

- **Vấn đề tranh đoạt điều khiển**

```
if (taikhoan-tienrut)>=0
    taikhoan=taikhoan-tienrut;
else
    error (<<khong the rut tien!>>);
```

- **Khái niệm miền găng:**

  **Đoạn chương trình có khả năng xảy ra các mâu thuẫn truy xuất trên tài nguyên chung**

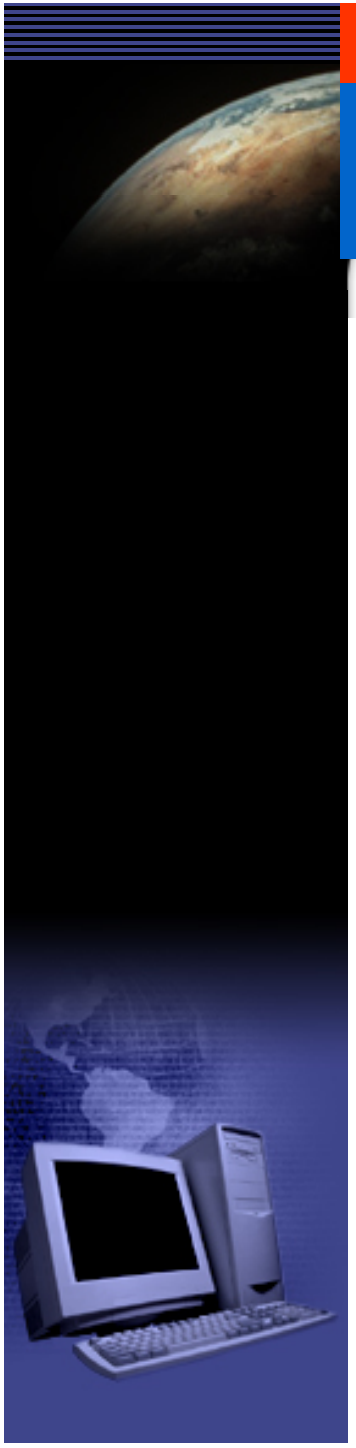


# Đồng bộ hoá tiến trình

## ❖ Miền găng (Critical Section)

- **Điều kiện giải quyết tốt bài toán miền găng:**

- **Không có 2 tiến trình cùng ở trong miền găng**
- **Không phụ thuộc vào tốc độ của tiến trình**
- **Một tiến trình tạm dừng bên ngoài miền găng không được ngăn cản các tiến trình khác vào miền găng**
- **Không có tiến trình nào phải chờ vô hạn để được vào miền găng.**



# Đồng bộ hoá tiến trình

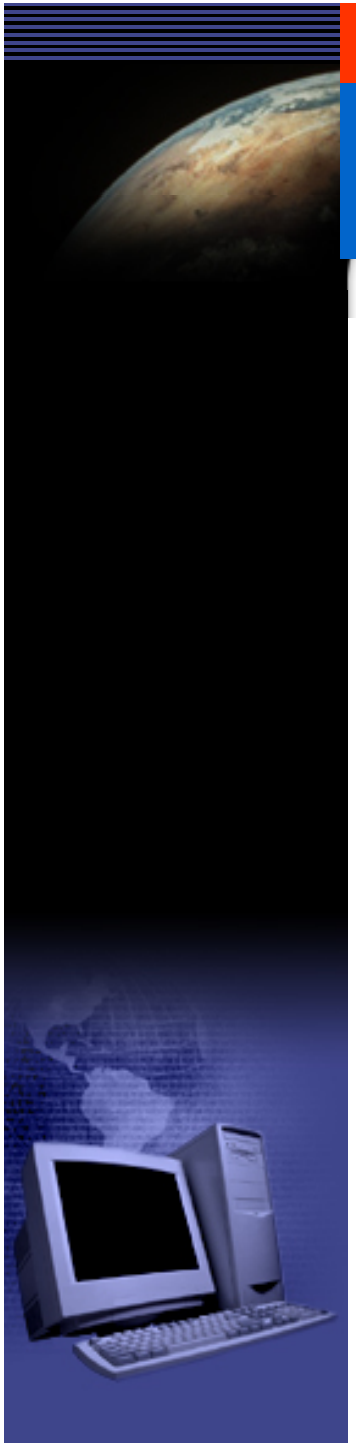
## ❖ Giải pháp

### ➢ Sử dụng biến khoá

- Dùng biến lock chung cho các tiến trình

- Nếu lock==1 thì khoá, không cho tiến trình vào miền găng. Chờ cho đến khi lock==0

- Nếu lock==0 thì cho tiến trình vào miền găng, đặt lock==1 để khoá không cho các tiến trình khác vào miền găng



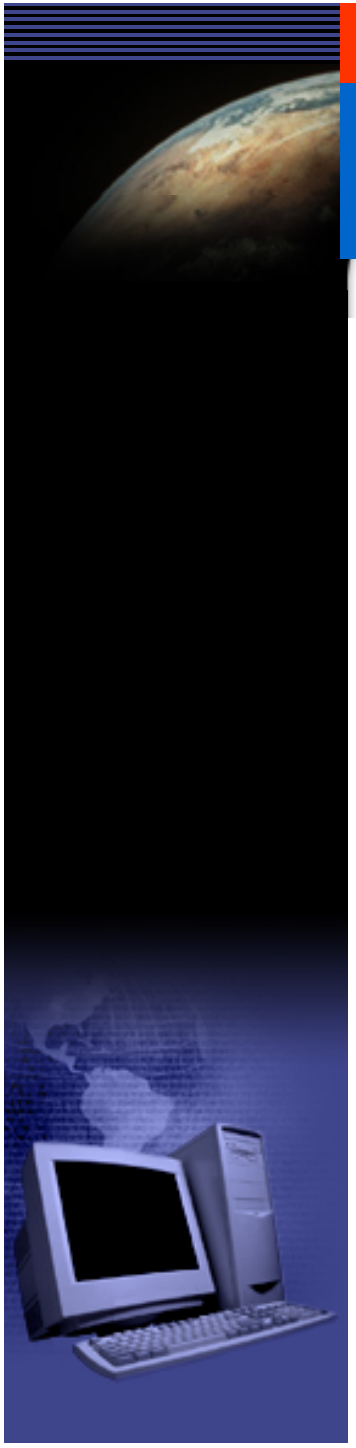
# Đồng bộ hoá tiến trình

## ❖ Giải pháp

### ➢ Sử dụng biến khoá

**- Giải thuật sử dụng biến khoá để đồng bộ**

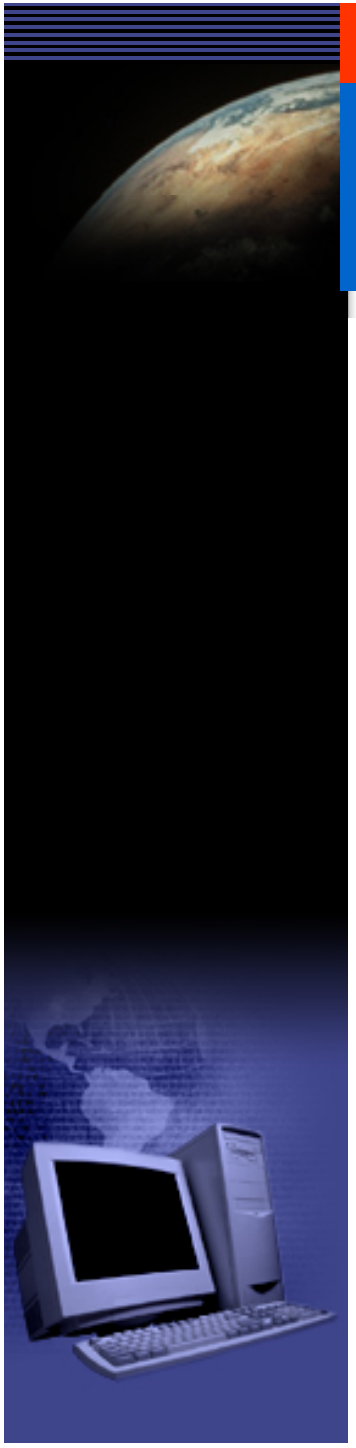
```
while (1) {
    while (lock==1);// wait
    lock=1;
    critical_section();
    lock=0;
    Noncritical_section();
}
```



# Đồng bộ hoá tiến trình

- Giải thuật sử dụng biến khoá để đồng bộ

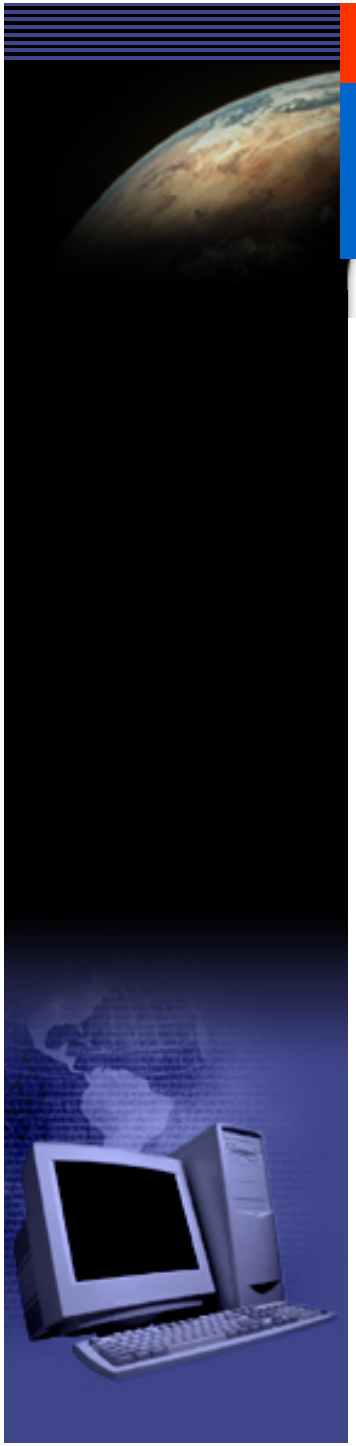
```
while (1) {
    while (lock==1);// wait
    lock=1;
    critical_section();
    lock=0;
    non_critical_section();
    }
```



# Đồng bộ hoá tiến trình

- Ví dụ: Áp dụng giải thuật sử dụng biến khoá để đồng bộ

```
while (1) {
    t=t*2;
    while (lock==1);// wait
    lock=1;
    for  (s=0,i=0;i<=t;i++)  s+=i;
    printf("s=%i",s);
    lock=0;
    break;
    }
```



# Đồng bộ hoá tiến trình

## ❖ Giải pháp

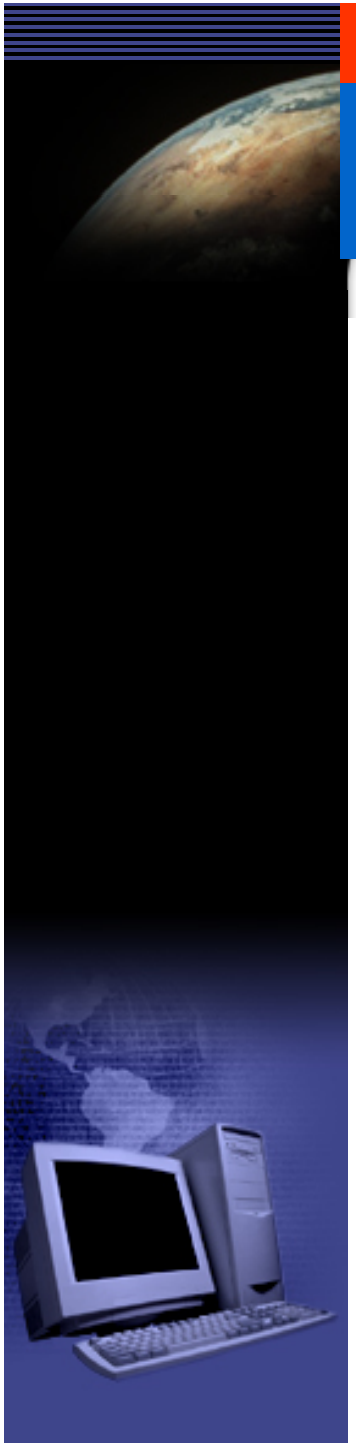
### ➢Kiểm tra luân phiên

- Các tiến trình muốn đi vào miền găng thì được gắn nhãn 0|1

- Sử dụng biến turn để chỉ thứ tự luân phiên.

- Nếu turn==0: tiến trình có nhãn 0 được vào miền găng

- Nếu turn==1: tiến trình có nhãn 1 được vào miền găng



# Đồng bộ hoá tiến trình

## ❖ Giải pháp

### ➢Kiểm tra luân phiên

**- Giải thuật của tiến trình có nhãn 0**

```
while (1) {
    while (turn != 0);// wait
    critical_section();
    turn=1;
    non_critical_section();
}
```

# Đồng bộ hoá tiến trình

## ❖ Giải pháp

### ➢Kiểm tra luân phiên

**- Giải thuật của tiến trình có nhãn 1**

```
while (1) {
    while (turn != 1);// wait
    critical_section();
    turn=0;
    non_critical_section();
}
```

# Đồng bộ hoá tiến trình

## ❖ Giải pháp

## ➢ Giải pháp Peterson

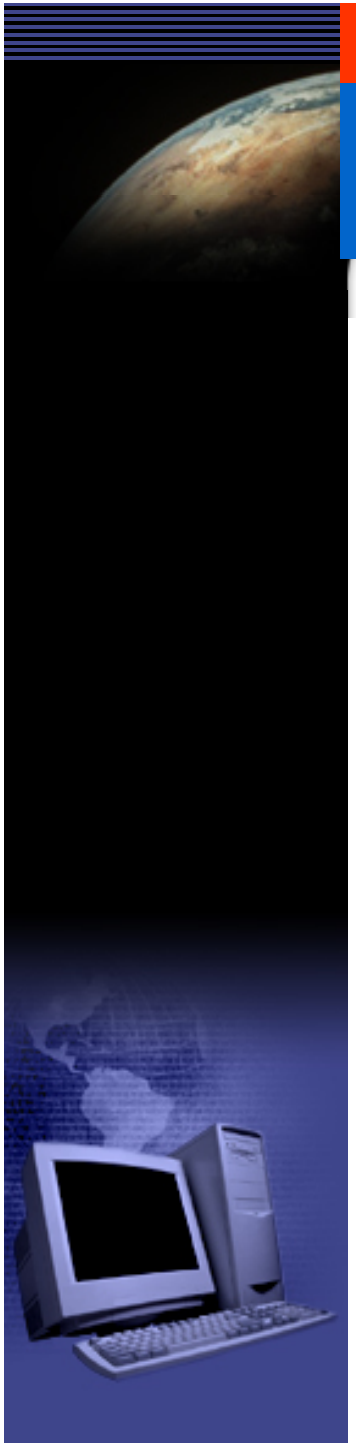
```
#define N  2  // Chỉ 2 tiến trình
int turn=0, interested[N]={0,0};
void enter_region(int process) // Vào ĐG
{  int other=1-process;//other là tiến trình đối của process
   interested[prcess]=1;
   turn=process;
   while ((turn==process)&&interested[other]==1);//chờ
}
```

# Đồng bộ hoá tiến trình

## ❖ Giải pháp

### ➢ Giải pháp Peterson

```
void leave_region(int process) // Ra khỏi ĐG
{
  interested[prcess]=0;
}
```



# Đồng bộ hoá tiến trình

## ❖ Giải pháp

### ➢ Giải pháp Sleep and Wakeup

**- Sử dụng 2 thủ tục: sleep và wakeup**

**- Khi tiến trình chưa đủ điều kiện để vào miền găng, nó goi sleep để tự khoá đến khi một tiến trình khác gọi wakeup để đánh thức nó.**

**- Tiến trình khi ra khỏi miền găng sẽ gọi wakeup để đánh thức tiến trình khác.**

**- int busy;// 1: nếu miền găng đang bận, 0:không bận**

**- int blocked;//đếm số lượng tiến trình đang bị khoá**

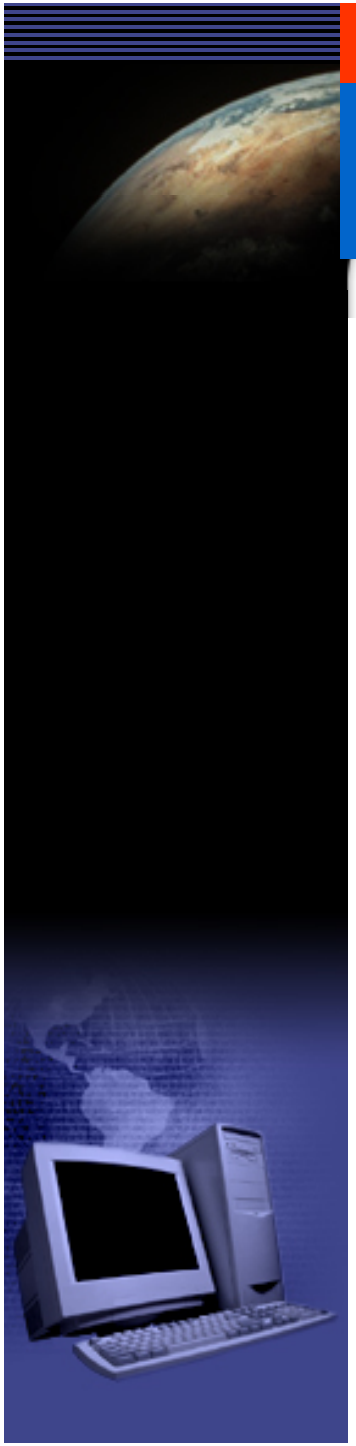
# Đồng bộ hoá tiến trình

## ❖ Giải pháp

### ➢ Giải pháp Sleep and Wakeup

**Giải thuật:**

```
while (1) {
    if (busy) {
        blocked=blocked+1;
        sleep();
    }
    else busy=1;
    critical_section();
    busy=0;
    if (blocked) {
        wakeup(process);
        blocked=blocked-1;
    }
    noncritical_section();
}
```



# Xác định trạng thái an toàn

❖ **Thuật toán:**

**Sử dụng các cấu trúc dữ liệu sau:**

**int allocation[numprocs,numresources];**

**//allocation[p,r] số lượng tài nguyên r thực sự cấp phát cho p**

**int max[numprocs,numresources];**

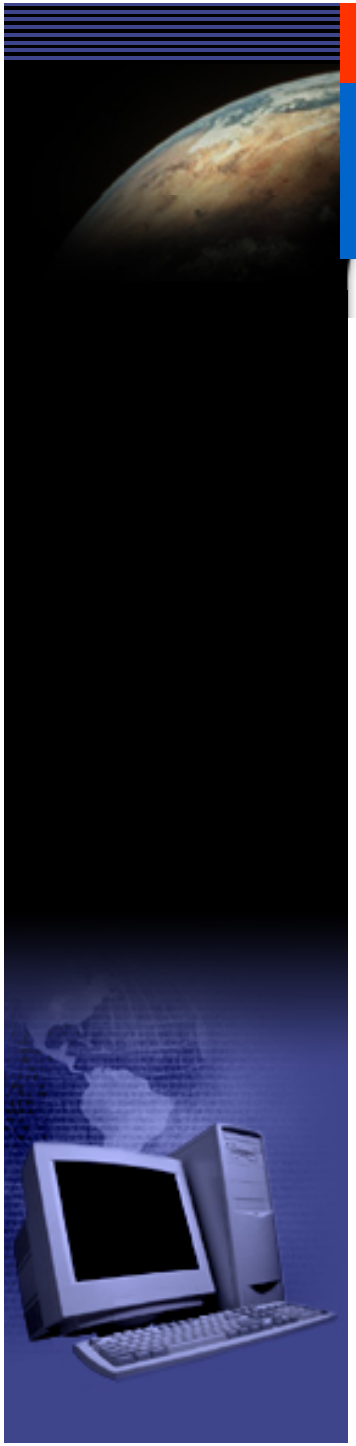
**// max[p,r] nhu cầu tối đa của tiến trình p về tài nguyên r**

**int need[numprocs,numresources];**

**//need[p,r]=max[p,r]-allocation[p,r]**

**int available[numresources]**

**//available[r] số lượng tài nguyên r còn tự do**



# Xác định trạng thái an toàn

❖ **Thuật toán:**

**int word[numresouces]=available;**

**int finish[numproces]=false;**

1. **Tìm i sao cho**

   **a. finish[i]==false;**

   **b. need[i,j]<=word[j]; với mọi tài nguyên j**

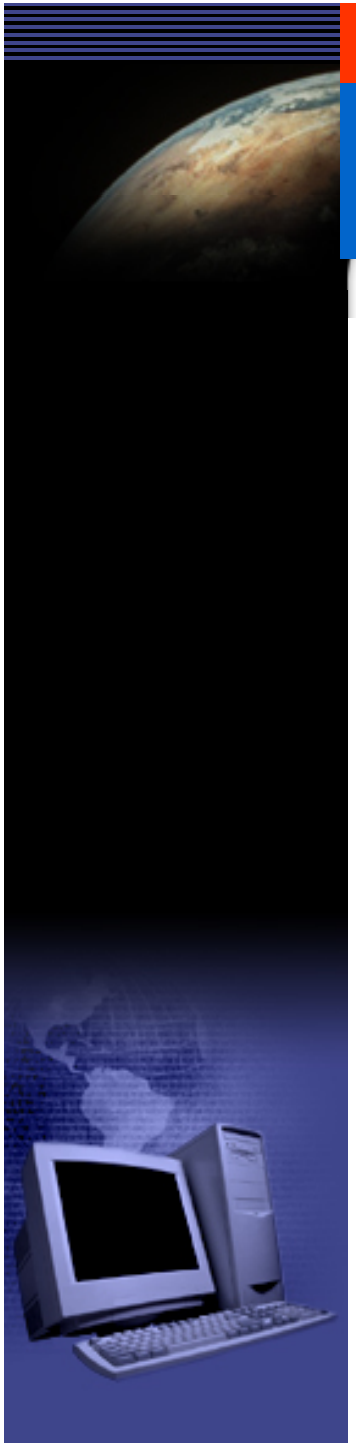
   **nếu không có i như thế, đến bước 3**

2. **Word[j]=word[j]+allocation[i,j];**

   **finish[i]=true;**

   **đến bước 1;**

3. **Nếu finish[i]==true với mọi i thì hệ thống ở trạng thái an toàn. Ngược lại hệ thống bị tắc nghẽn**



# Xác định trạng thái an toàn

❖ **ví dụ: giả sử tình trạng hiện hành của hệ thống được mô tả ở bảng dưới. Nếu tiến trình P2 yêu cầu cấp 4 R1, 1 R3. Hãy cho biết yêu cầu này có thể đáp ứng mà không xảy ra tình trạng tắt nghẽn**

| | Max | | | Allocation | | | Available | | |
|---|---|---|---|---|---|---|---|---|---|
| | R1 | R2 | R3 | R1 | R2 | R3 | R1 | R2 | R3 |
| P1 | 3 | 2 | 2 | 1 | 0 | 0 | 4 | 1 | 2 |
| P2 | 6 | 1 | 3 | 2 | 1 | 1 | | | |
| P3 | 3 | 1 | 4 | 2 | 1 | 1 | | | |
| P4 | 4 | 2 | 2 | 0 | 0 | 2 | | | |

# Xác định trạng thái an toàn

❖ **ví dụ:**

**Available[1]=4, Available[3]=2 đủ để thoả mãn yêu cầu của P2, ta có**

| | Need | | | Allocation | | | Available | | |
|---|---|---|---|---|---|---|---|---|---|
| | R1 | R2 | R3 | R1 | R2 | R3 | R1 | R2 | R3 |
| P1 | 2 | 2 | 2 | 1 | 0 | 0 | 0 | 1 | 1 |
| P2 | 0 | 0 | 1 | 6 | 1 | 2 | | | |
| P3 | 1 | 0 | 3 | 2 | 1 | 1 | | | |
| P4 | 4 | 2 | 0 | 0 | 0 | 2 | | | |

# Xác định trạng thái an toàn

❖ **ví dụ:**

| | Need | | | Allocation | | | Available | | |
|---|---|---|---|---|---|---|---|---|---|
| | R1 | R2 | R3 | R1 | R2 | R3 | R1 | R2 | R3 |
| P1 | 2 | 2 | 2 | 1 | 0 | 0 | 6 | 2 | 3 |
| P2 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| P3 | 1 | 0 | 3 | 2 | 1 | 1 | | | |
| P4 | 4 | 2 | 0 | 0 | 0 | 2 | | | |

# Xác định trạng thái an toàn

❖ **ví dụ:**

| | Need | | | Allocation | | | Available | | |
|---|---|---|---|---|---|---|---|---|---|
| | R1 | R2 | R3 | R1 | R2 | R3 | R1 | R2 | R3 |
| P1 | 0 | 0 | 0 | 0 | 0 | 0 | 7 | 2 | 3 |
| P2 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| P3 | 1 | 0 | 3 | 2 | 1 | 1 | | | |
| P4 | 4 | 2 | 0 | 0 | 0 | 2 | | | |

# Xác định trạng thái an toàn

❖ **Ví dụ:**

| | Need | | | Allocation | | | Available | | |
|---|---|---|---|---|---|---|---|---|---|
| | R1 | R2 | R3 | R1 | R2 | R3 | R1 | R2 | R3 |
| P1 | 0 | 0 | 0 | 0 | 0 | 0 | 9 | 3 | 4 |
| P2 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| P3 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| P4 | 4 | 2 | 0 | 0 | 0 | 2 | | | |

# Xác định trạng thái an toàn

❖ **Ví dụ:**

| | Need | | | Allocation | | | Available | | |
|---|---|---|---|---|---|---|---|---|---|
| | R1 | R2 | R3 | R1 | R2 | R3 | R1 | R2 | R3 |
| P1 | 0 | 0 | 0 | 0 | 0 | 0 | 9 | 3 | 6 |
| P2 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| P3 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| P4 | 0 | 0 | 0 | 0 | 0 | 0 | | | |

**Trạng thái kết quả là an toàn, có thể cấp phát.**

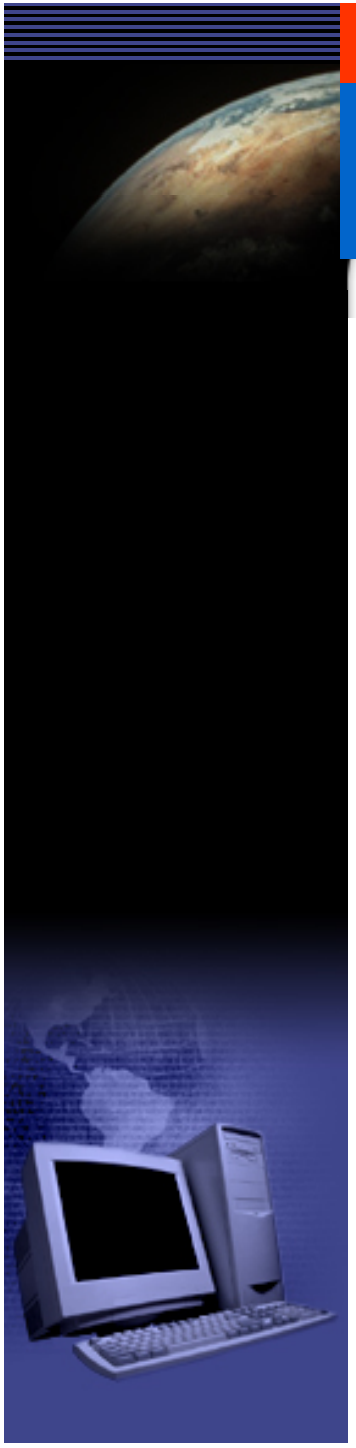
# Xác định trạng thái an toàn

❖ **Bài tập:**

| | Max | | | Allocation | | | Available | | |
|---|---|---|---|---|---|---|---|---|---|
| | R1 | R2 | R3 | R1 | R2 | R3 | R1 | R2 | R3 |
| P1 | 3 | 2 | 2 | 1 | 0 | 0 | 4 | 1 | 2 |
| P2 | 6 | 1 | 3 | 2 | 1 | 1 | | | |
| P3 | 3 | 1 | 4 | 2 | 1 | 1 | | | |
| P4 | 4 | 2 | 2 | 0 | 0 | 2 | | | |

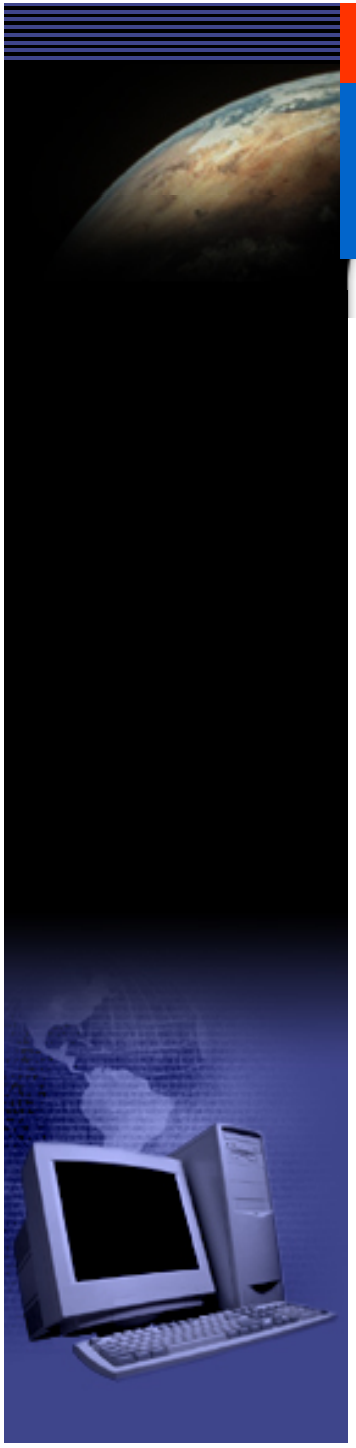
**Tiến trình P2 yêu cầu 4 R1, 1 R3. Hãy cho biết yêu cầu này có thể đáp ứng mà đảm bảo không xảy ra tình trạng tắt nghẽn hay không?**

## CHƯƠNG 4. QUẢN LÝ BỘ NHỚ



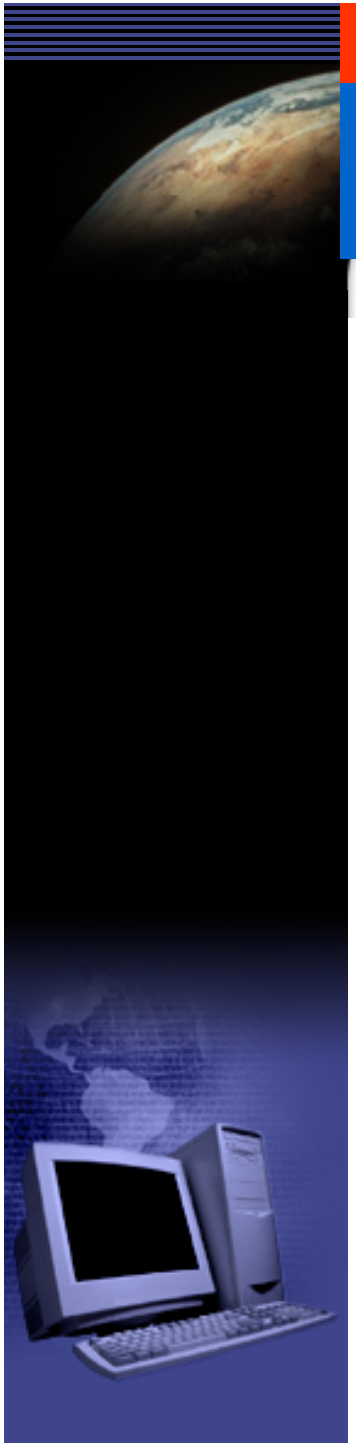
# Các vấn đề

1. Khái niệm
2. Không gian địa chỉ và không gian vật lý
3. Cấp phát liên tục
4. Cấp phát không liên tục
5. Bộ nhớ ảo



# Khái niệm

- **Bộ nhớ là thiết bị lưu trữ duy nhất thông qua đó CPU có thể trao đổi thông tin với môi trường ngoài.**
- **Bộ nhớ chính được tổ chức như một mảng một chiều các từ nhớ (word), mỗi từ nhớ có một địa chỉ.**
- **Việc trao đổi với môi trường ngoài thông qua thao tác đọc, ghi dữ liệu vào một địa chỉ cụ thể trong bộ nhớ**



# Khái niệm

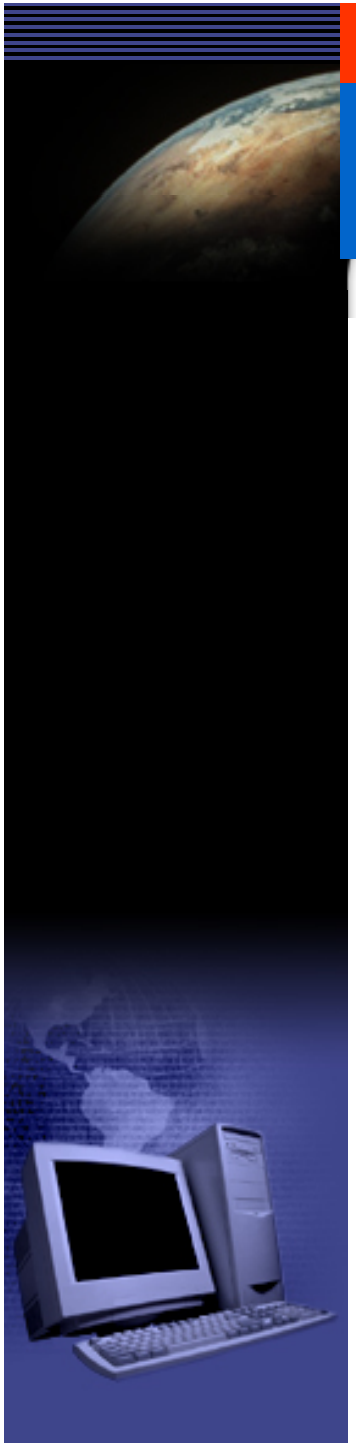
- **Hệ điều hành thực hiện:**
  - **Sự tương ứng giữa địa chỉ logic và địa chỉ vật lý**
  - **Quản lý bộ nhớ vật lý**
  - **Chia sẻ thông tin**
  - **Bảo vệ**

# Không gian địa chỉ và không gian vật lý

- **Địa chỉ logic (địa chỉ ảo): các địa chỉ do bộ xử lý tạo ra.**
- **Địa chỉ vật lý: địa chỉ thực tế mà trình quản lý bộ nhớ nhìn thấy và thao tác.**
- **Không gian địa chỉ: tập hợp tất cả các địa chỉ ảo phát sinh bởi một chương trình.**

# Không gian địa chỉ và không gian vật lý

- **Không gian vật lý: tập hợp tất cả các địa chỉ vật lý tương ứng với các địa chỉ ảo.**
- **MMU (Memory Management Unit): một cơ chế phần cứng chuyển đổi địa chỉ ảo thành địa chỉ vật lý.**
- **Chương trình của NSD chỉ thao tác trên địa chỉ ảo.**

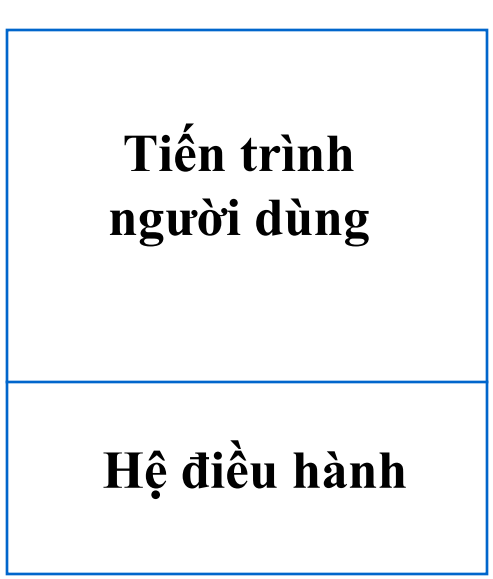


# Cấp phát liên tục

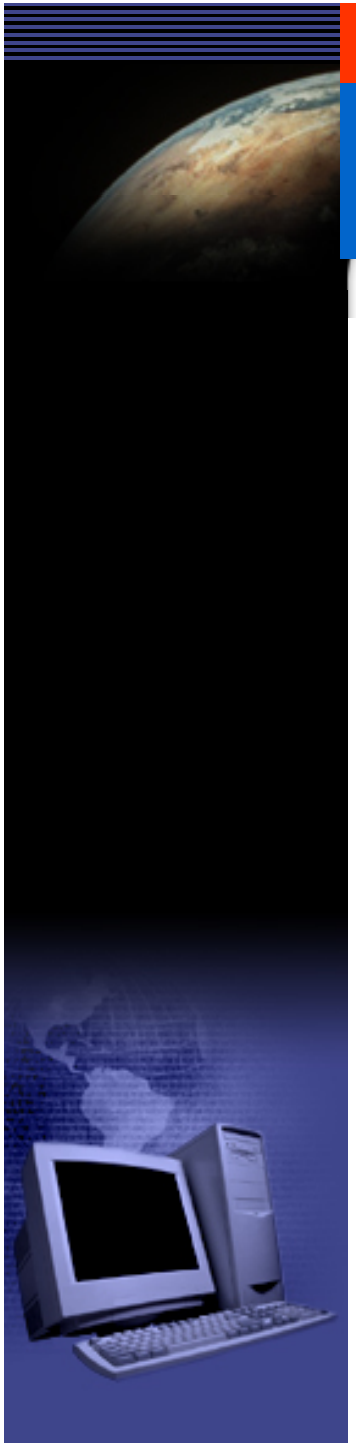
- Các hệ đơn chương
- Các hệ thống đa chương với phân vùng cố định
- Các hệ thống đa chương với phân vùng động
- Các hệ thống đa chương với kỹ thuật “Swapping”

# Cấp phát liên tục

- Các hệ đơn chương




Tổ chức bộ nhớ trong hệ thống đơn chương



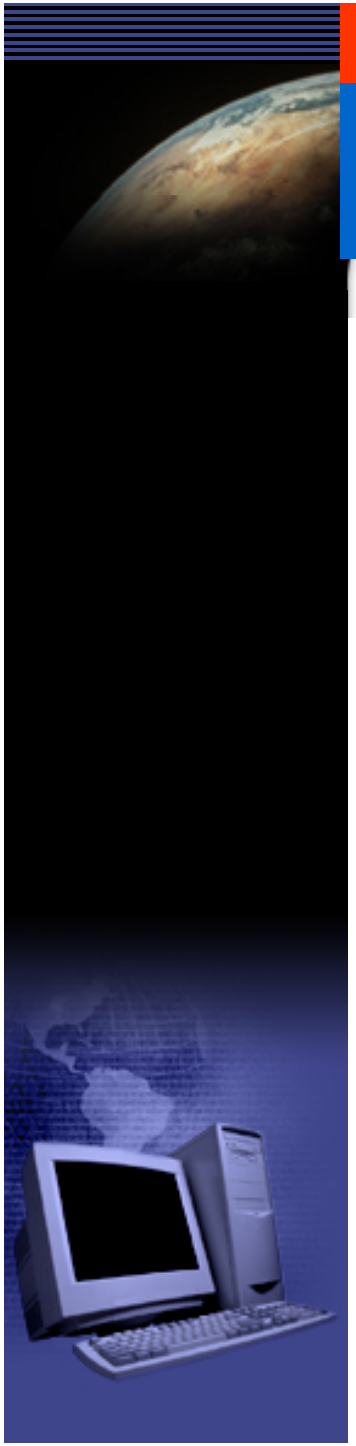
# Cấp phát liên tục

❖ **Các hệ thống đơn chương**

- **Sử dụng thanh ghi giới hạn: địa chỉ cao nhất của vùng nhớ được cấp cho HĐH**
- **Tất cả các địa chỉ được tiến trình NSD truy xuất đến sẽ được so sánh với nội dung thanh ghi giới hạn.**

  **+ Nếu lớn hơn: hợp lý.**

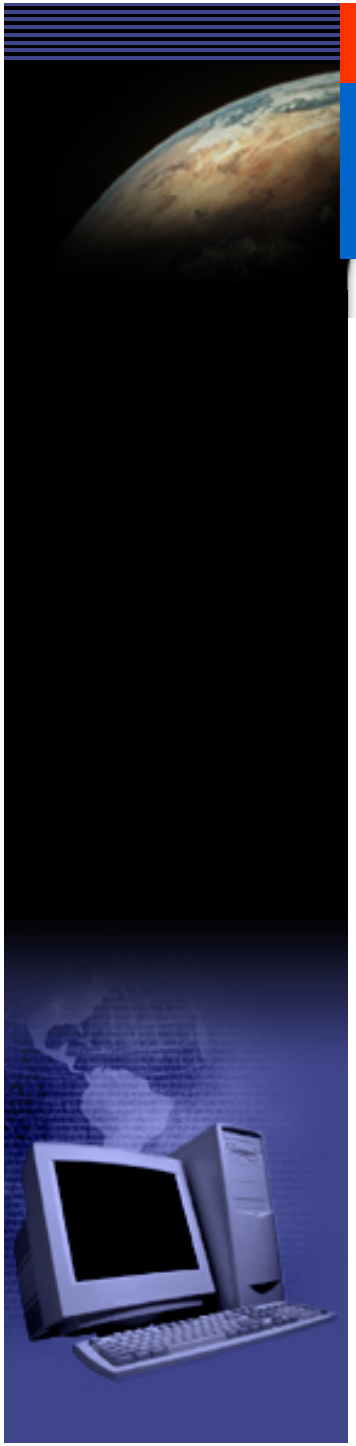
  **+ Ngược lại : một ngắt sẽ được phát sinh báo sự truy xuất bất hợp lý.**
- **Tại một thời điểm chỉ có một chương trình được xử lý.**



# Cấp phát liên tục

- **Các hệ thống đơn chương**

**Ví dụ: Trong HĐH MSDOS, một lúc chỉ thực thi được một lệnh. Khi NSD gõ lệnh lập tức lệnh đó được thực hiện và sau khi hoàn tất, con trỏ xuất hiện sau dấu nhắc đợi lệnh chờ NSD gõ lệnh tiếp theo.**

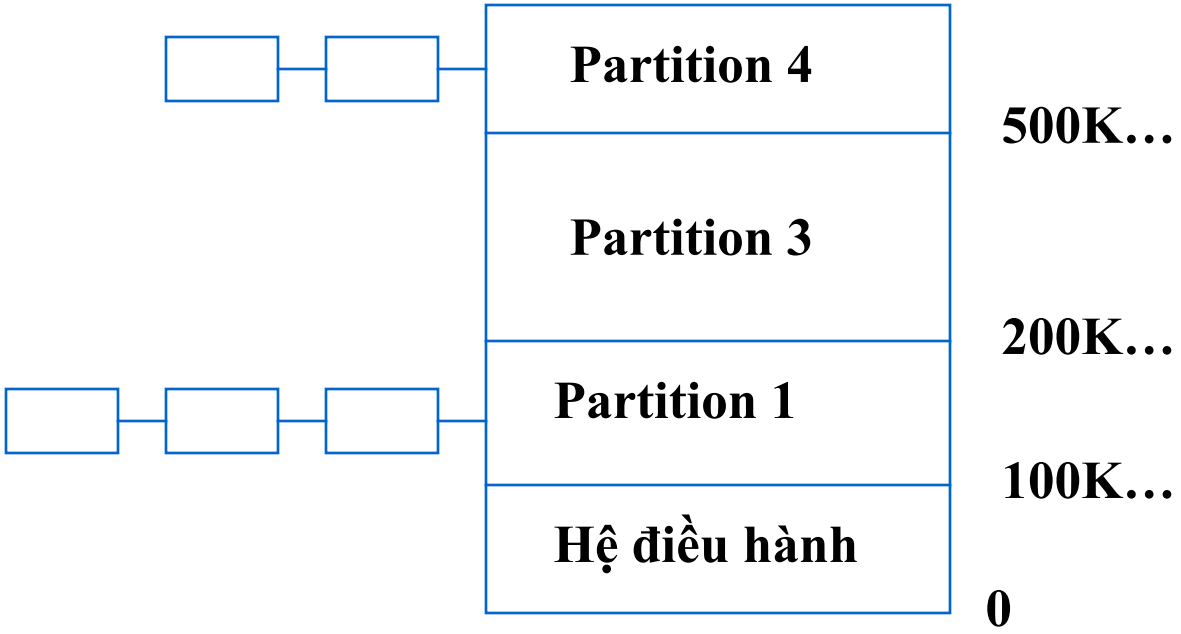


# Cấp phát liên tục

- Các hệ thống đa chương với phân vùng cố định
  - Bộ nhớ được chia thành các phân vùng (kích thước khác hay bằng nhau)
  - Các tiến trình có nhu cầu bộ nhớ sẽ được lưu trữ vào hàng đợi.
  - Sử dụng nhiều hàng đợi
  - Sử dụng một hàng đợi

# Cấp phát liên tục

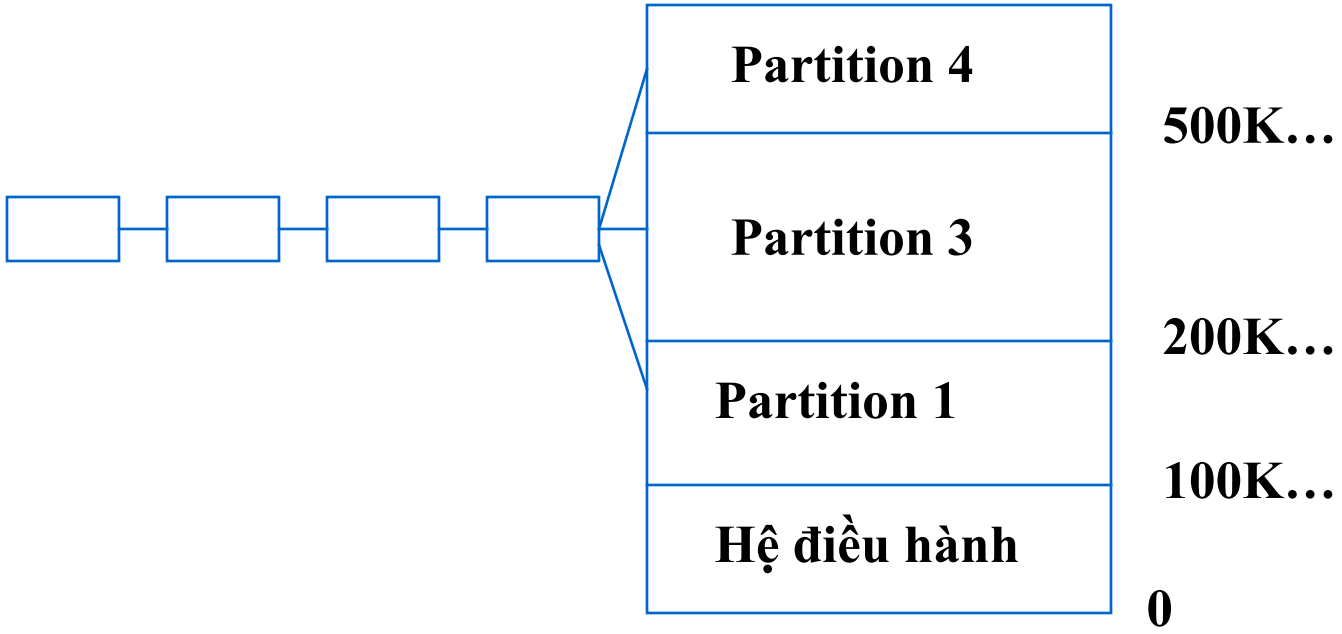
- Các hệ thống đa chương với phân vùng cố định



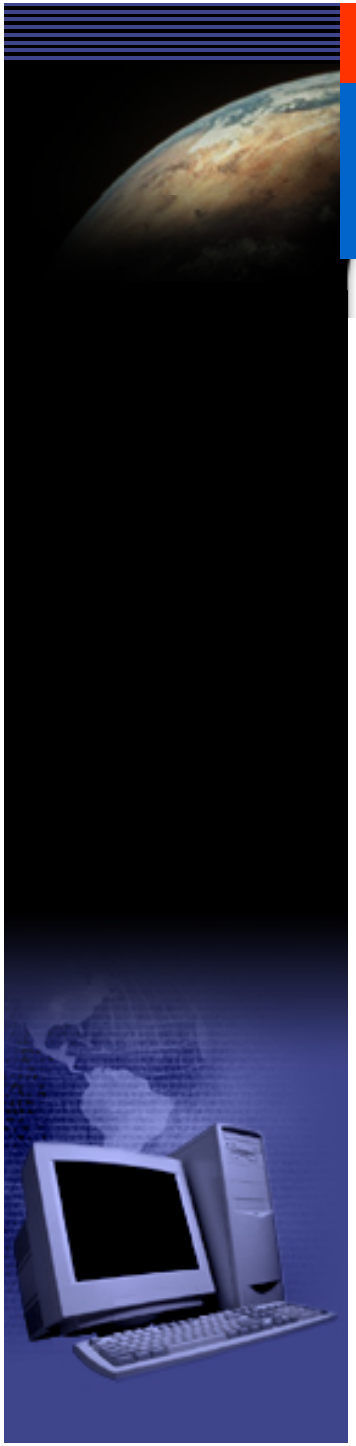

Phân vùng cố định nhiều hàng đợi

# Cấp phát liên tục

- Các hệ thống đa chương với phân vùng cố định

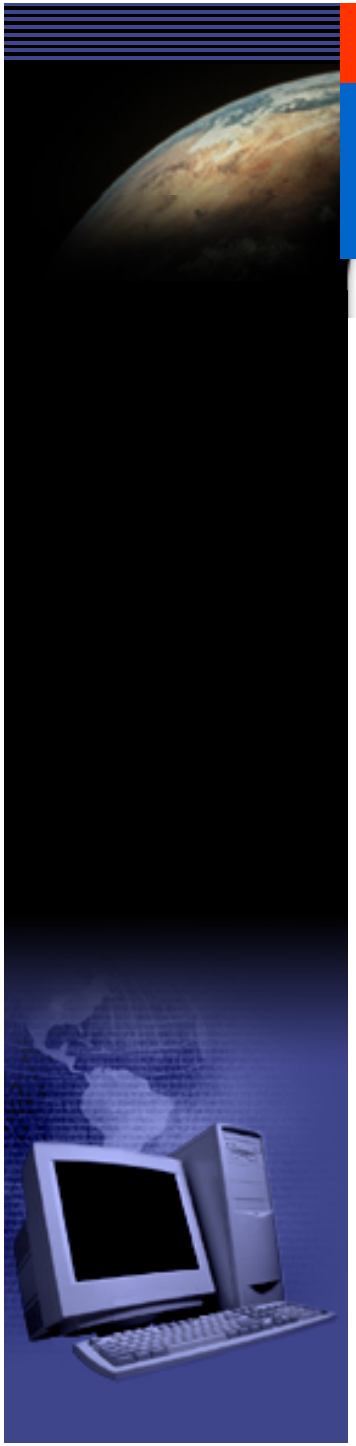



Phân vùng cố định một hàng đợi



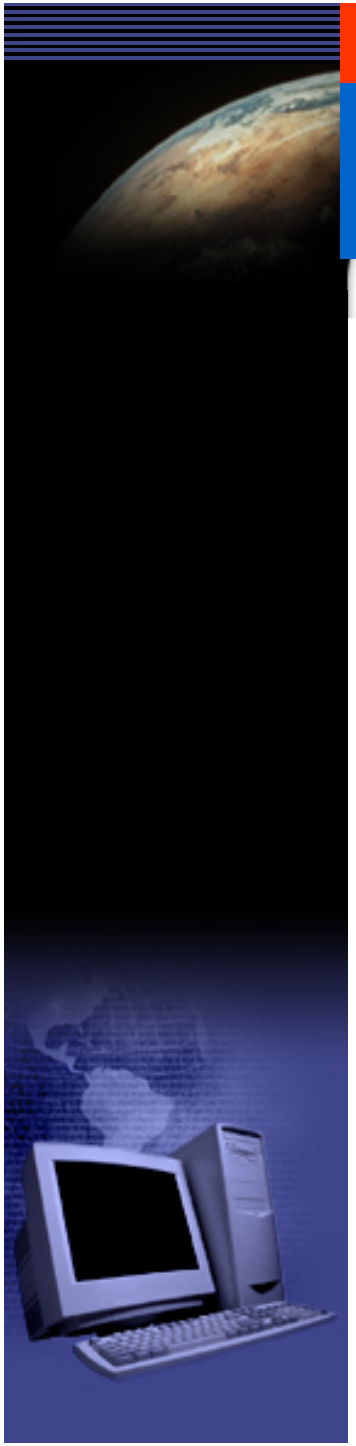
# Cấp phát liên tục

- Các hệ thống đa chương với phân vùng cố định
  - Phân vùng cố định nhiều hàng đợi
    - Mỗi phân vùng có một hàng đợi
    - Mỗi tiến trình mới được tạo lập sẽ được đưa vào hàng đợi của phân vùng có kích thước nhỏ nhất đủ để thoả mãn nhu cầu chứa nó.
    - Các hàng đợi của một số phân vùng trống, đầy. Các tiến trình phải chờ được cấp phát bộ nhớ.



# Cấp phát liên tục

- Các hệ thống đa chương với phân vùng cố định
- Phân vùng cố định một hàng đợi
  - Tất cả các tiến trình được đặt trong một hàng đợi.
  - Khi có một phân vùng tự do, tiến trình đầu tiên trong hàng đợi có kích thước phù hợp sẽ được đặt vào phân vùng này cho xử lý.
  - Kích thước của tiến trình không đúng bằng kích thước của phân vùng tự do ⇒ phân mảnh nội vi
  - Mức độ đa chương bị giới hạn bởi số lượng phân vùng

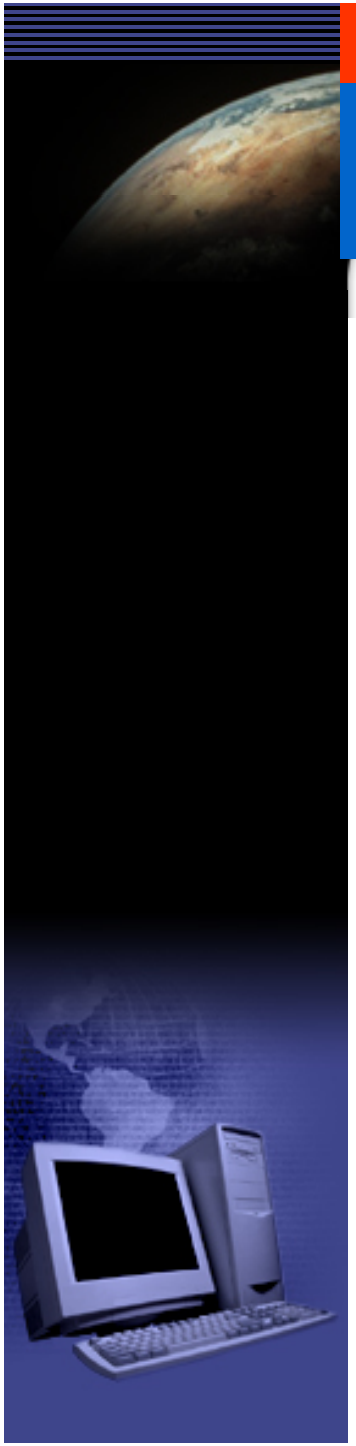


# Cấp phát liên tục

- Các hệ thống đa chương với phân vùng cố định
  - Phân vùng cố định một hàng đợi
    - Giải quyết 2 vấn đề của đa chương: sự tái định vị, sự bảo vệ

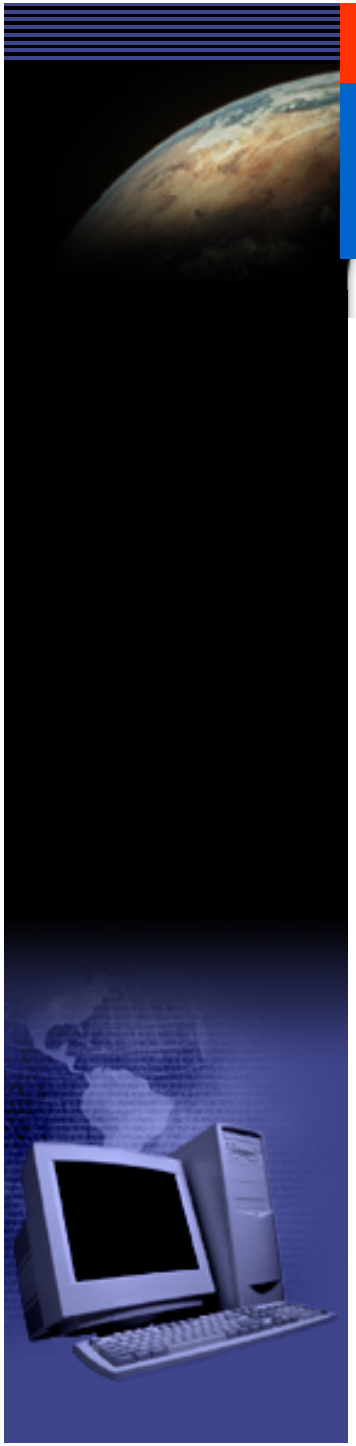
Ví dụ: giả sử chương trình truy xuất đến địa chỉ 100 (địa chỉ tương đối), ct được nạp vào phân vùng 1 địa chỉ bắt đầu 100k, thì địa chỉ truy xuất là (100k+100)

- Tái định vị vào thời điểm nạp chương trình



# Cấp phát liên tục

- Các hệ thống đa chương với phân vùng cố định
- Phân vùng cố định một hàng đợi
- Sử dụng các thanh ghi đặc biệt: phần cứng
  - Thanh ghi nền (Base Register)
  - Thanh ghi giới hạn (Limit Register)
- Khi một tiến trình được tạo lập, nạp vào thanh ghi nền địa chỉ bắt đầu của phân vùng được nạp, nạp vào thanh ghi giới hạn kích thước của tiến trình.

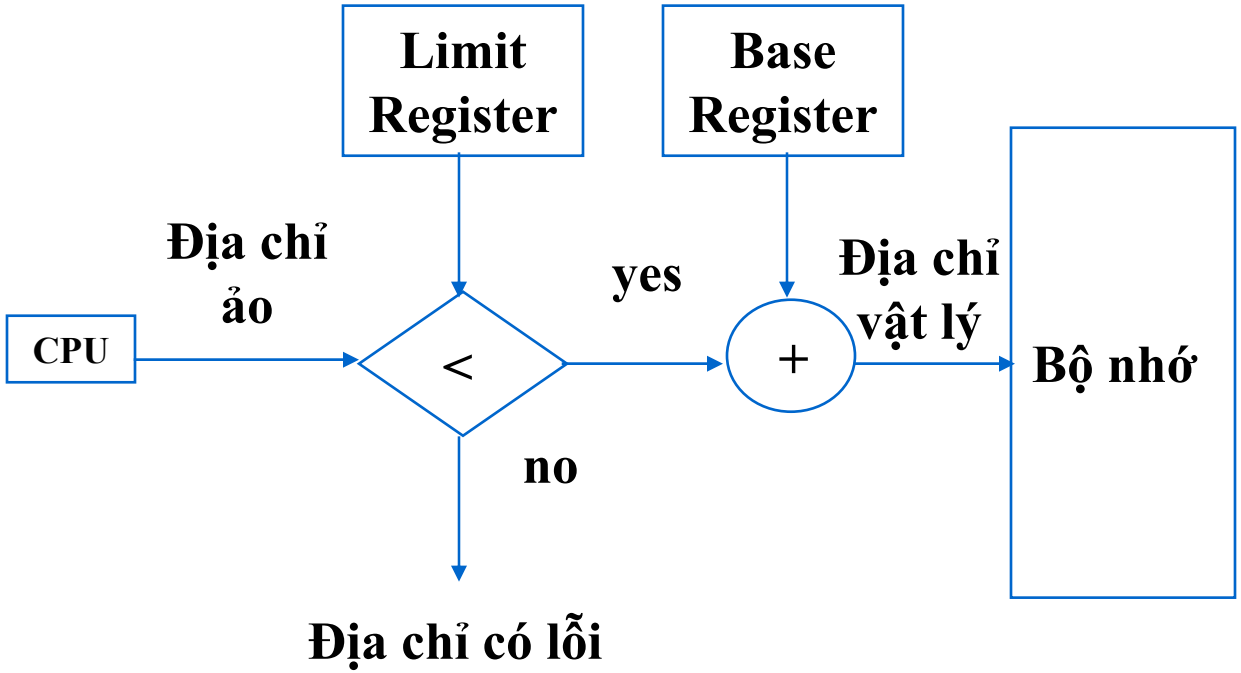


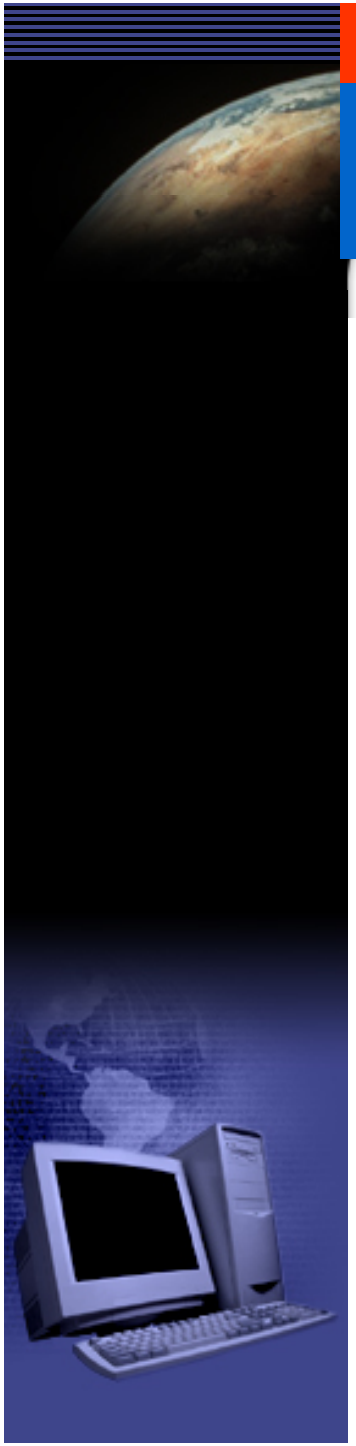
# Cấp phát liên tục

- Các hệ thống đa chương với phân vùng cố định
- Phân vùng cố định một hàng đợi
  - Địa chỉ ảo được đối chiếu với thanh ghi giới hạn để bảo đảm tiến trình không truy xuất ngoài phạm vi phân vùng cấp cho nó.
  - Địa chỉ vật lý=địa chỉ ảo+địa chỉ trong thanh ghi nền.
  - Sử dụng thanh ghi nền là có thể di chuyển các chương trình trong bộ nhớ sau khi chúng bắt đầu xử lý. Chỉ cần nạp lại thanh ghi nền.

# Cấp phát liên tục

- Các hệ thống đa chương với phân vùng cố định
- Phân vùng cố định một hàng đợi





# Cấp phát liên tục

❖ **Các hệ thống đa chương với phân vùng động**

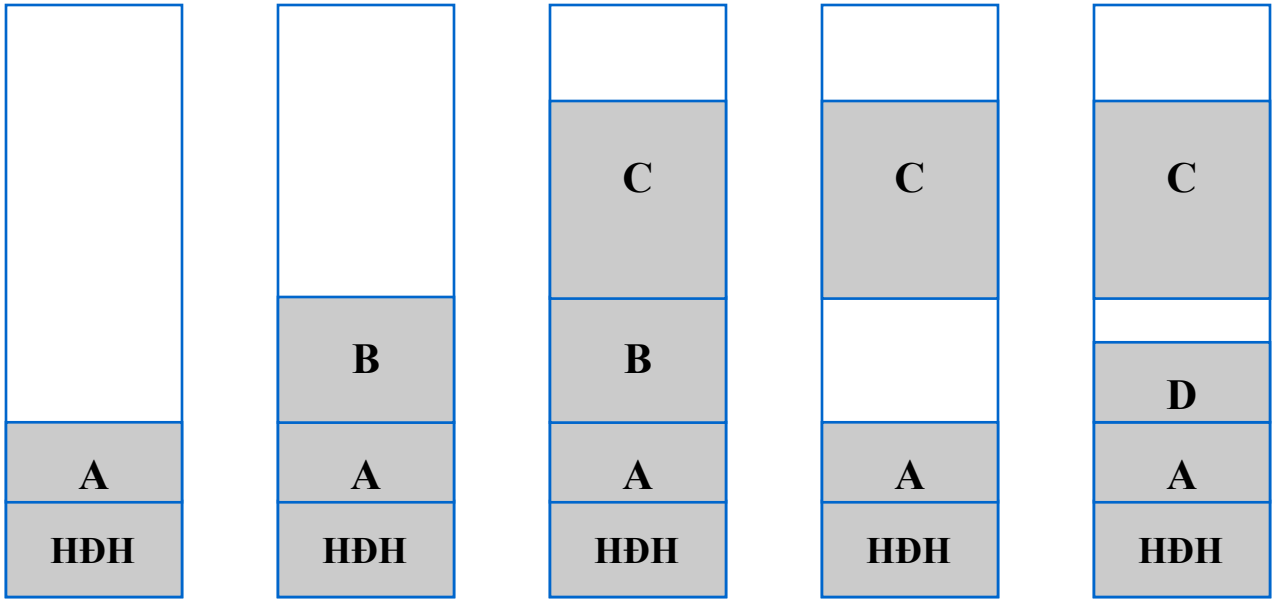
- **Xảy ra hiện tượng phân mảnh ngoại vi**
- **Kỹ thuật "dồn bộ nhớ": kết hợp các mảnh bộ nhớ nhỏ rời rạc thành một vùng nhớ lớn liên tục**

⇒ **Các tiến trình có thể bị di chuyển.**

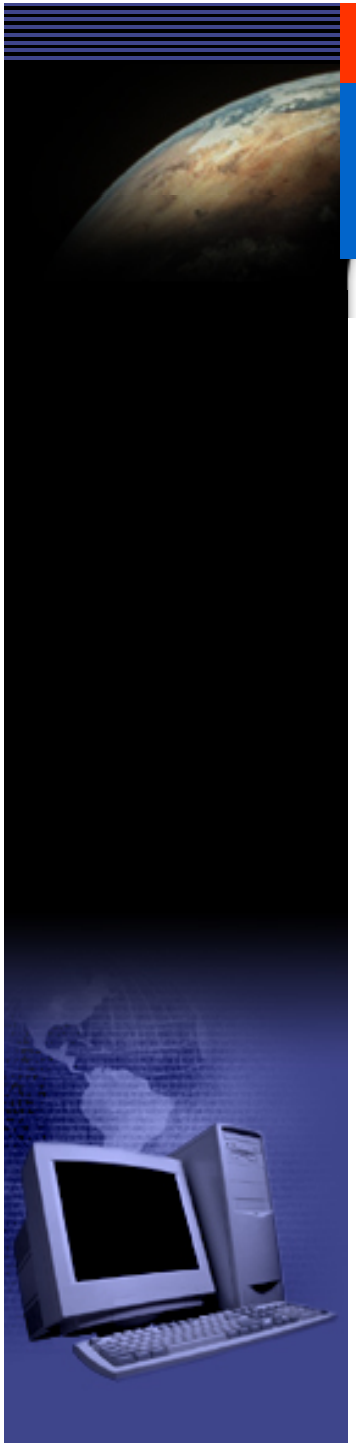
⇒ **Kích thước tiến trình tăng trưởng trong quá trình xử lý mà không còn vùng nhớ trống gần kề (dời chỗ tiến trình, cấp phát dư).**

# Cấp phát liên tục

- Các hệ thống đa chương với phân vùng động




Cấp phát các phân vùng động

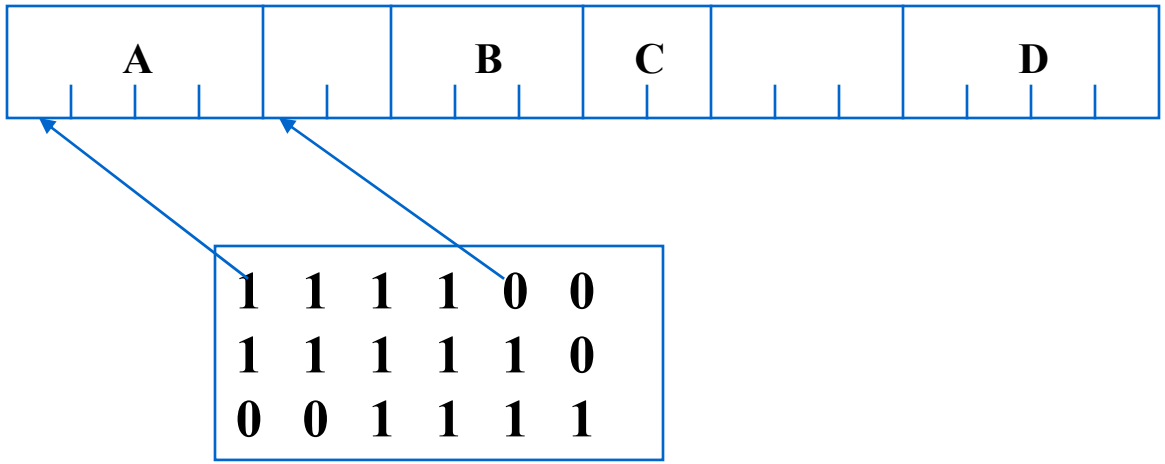


# Cấp phát liên tục

- Các hệ thống đa chương với phân vùng động
  - Giải pháp cấp phát động
    - Quản lý bằng một bảng các bit
    - Quản lý bằng danh sách

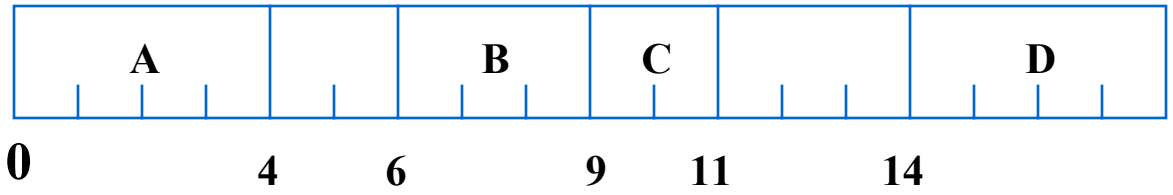
# Cấp phát liên tục

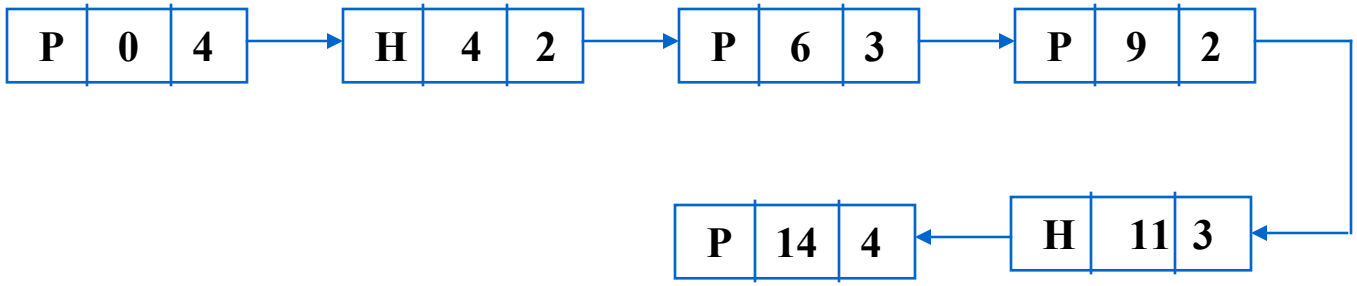
- Các hệ thống đa chương với phân vùng động
- Quản lý bằng một bảng các bit



# Cấp phát liên tục

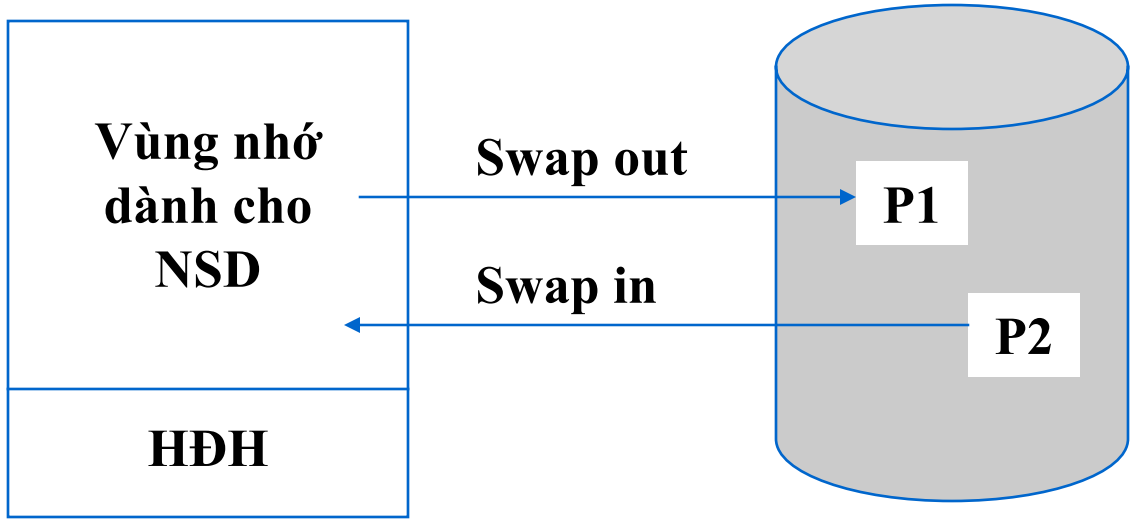
- Các hệ thống đa chương với phân vùng động
- Quản lý bằng danh sách

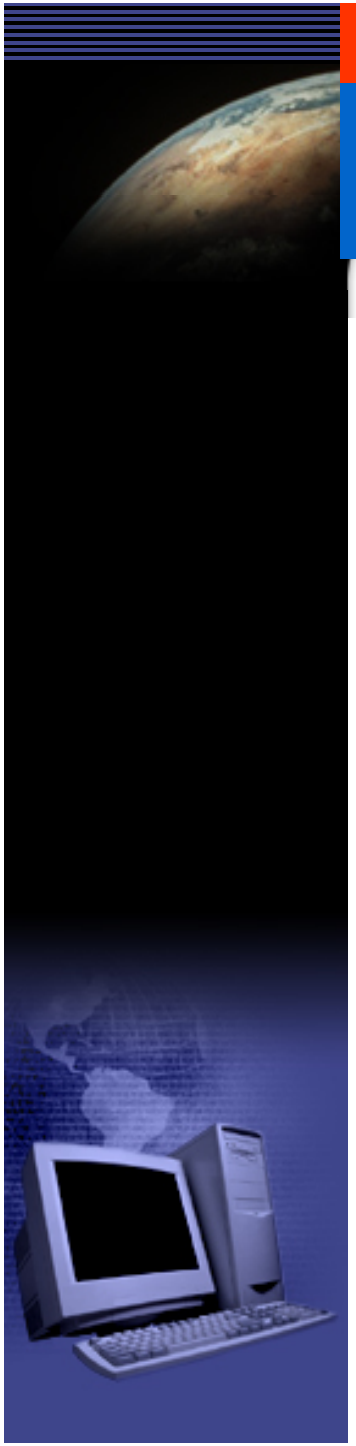


# Cấp phát liên tục

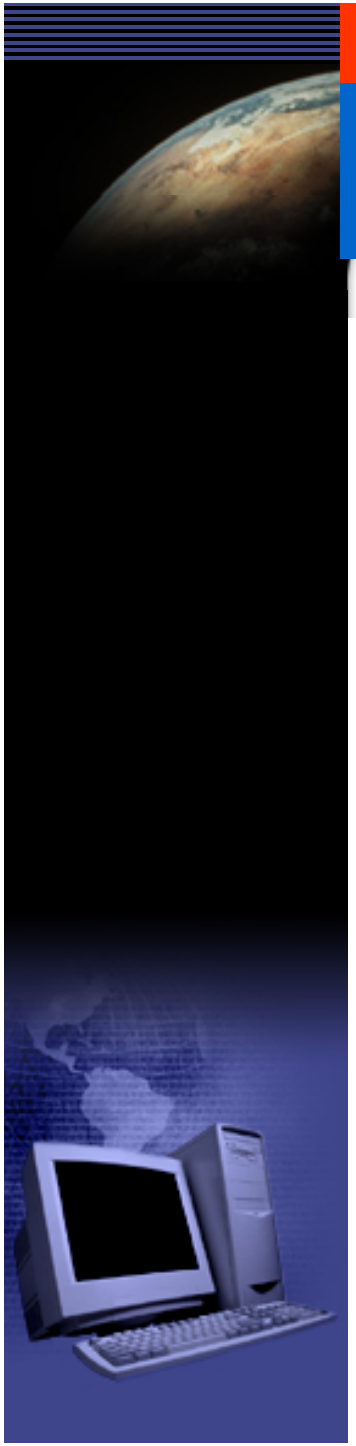
- Các hệ thống đa chương với kỹ thuật "Swapping"





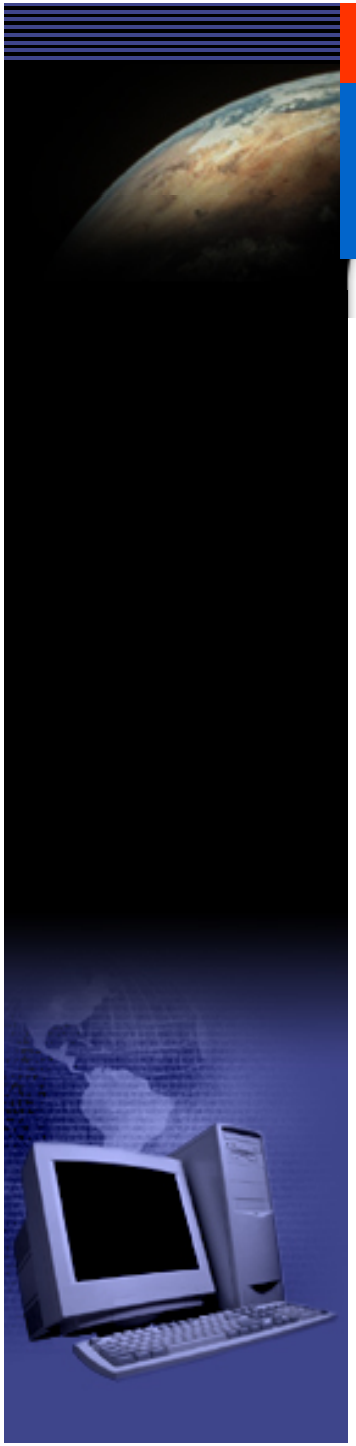
# Cấp phát liên tục

- Các hệ thống đa chương với kỹ thuật "Swapping"
  - Chuyển một tiến trình đang ở trạng thái chờ nằm sang bộ nhớ phụ. (swap out)
  - Khi đến lượt nó sẽ được mang trở lại bộ nhớ chính để tiếp tục xử lý. (swap in)
  - Xảy ra hiện tượng phân mảng ngoại vi.



# Cấp phát không liên tục

- Phân trang
- Phân đoạn
- Phân đoạn kết hợp phân trang



# Cấp phát không liên tục

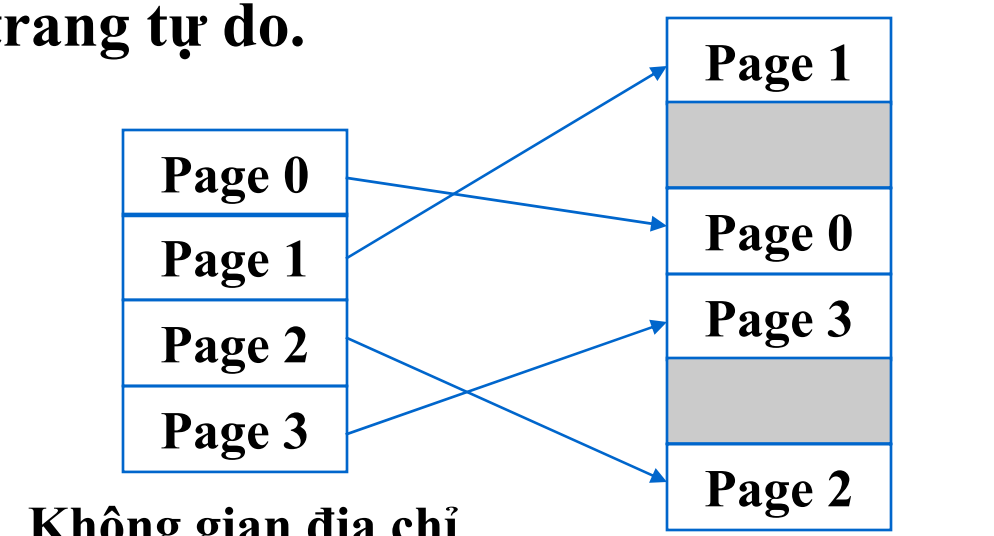
- **Phân trang**
  - **Ý tưởng**
  - **Cơ chế MMU**
  - **Chuyển đổi địa chỉ**
  - **Cài đặt bảng trang**
  - **Tổ chức bảng trang**

# Cấp phát không liên tục

- **Phân trang**
  - **Ý tưởng**
    - **Bộ nhớ vật lý: chia thành các khối (khung trang) có kích thước bằng nhau.**
    - **Không gian địa chỉ: chia thành các khối (trang) có kích thước trùng bằng khung trang.**
    - **Khi cần nạp một tiến trình để xử lý, các trang của tiến trình sẽ được nạp vào các khung trang còn trống.**

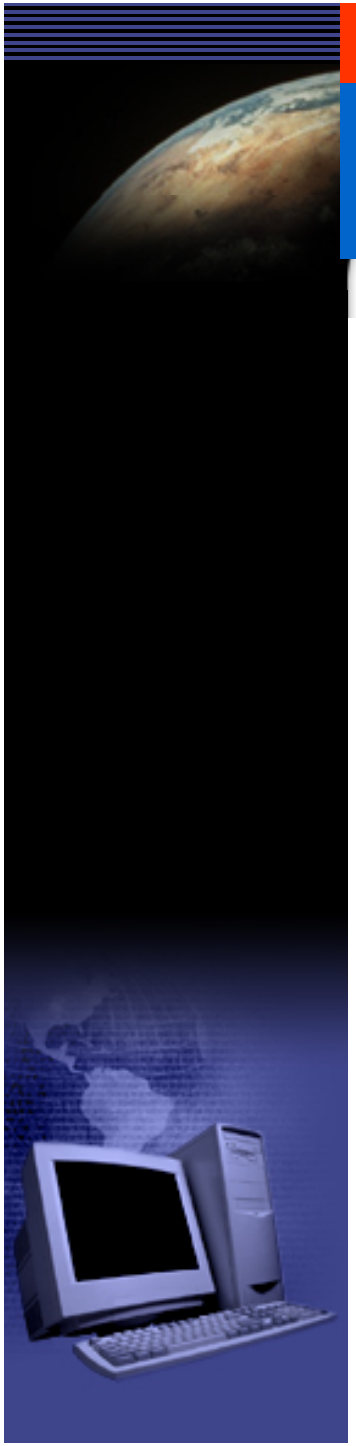
# Cấp phát không liên tục

- **Phân trang**
  - **Ý tưởng**
    - **Tiến trình có kích thước N trang, sẽ yêu cầu N khung trang tự do.**



# Cấp phát không liên tục

- **Phân trang**
  - **Cơ chế MMU(Memory Management Unit)**
    - **Cơ chế phần cứng hỗ trợ chuyển đổi địa chỉ trong cơ chế phân trang (bảng trang).**
    - **Mỗi phân tử trong bảng trang: địa chỉ bắt đầu lưu trữ trang tương ứng trong bộ nhớ vật lý;số hiệu khung trang tương ứng.**



# Cấp phát không liên tục

- **Phân trang**
  - **Chuyển đổi địa chỉ**

**- Địa chỉ phát sinh bởi CPU gồm 2 phần: p,d**

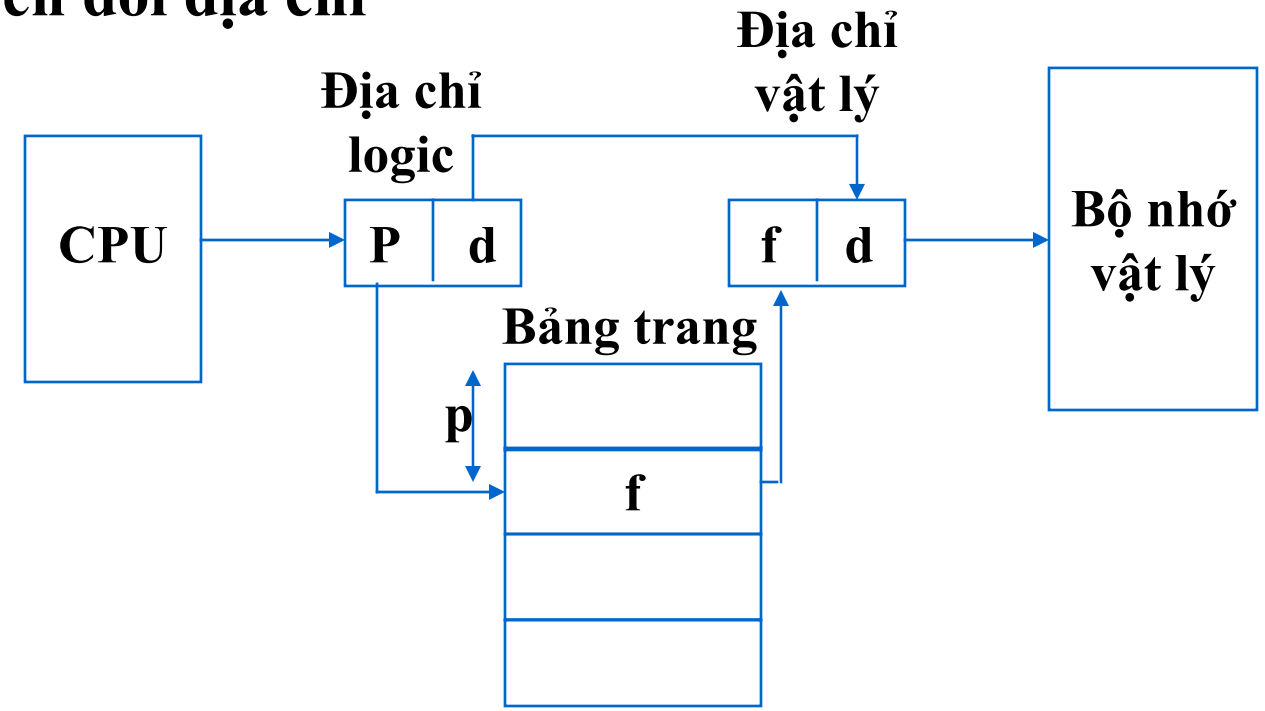
**+ p: số hiệu trang**

**+ d: địa chỉ tương đối**

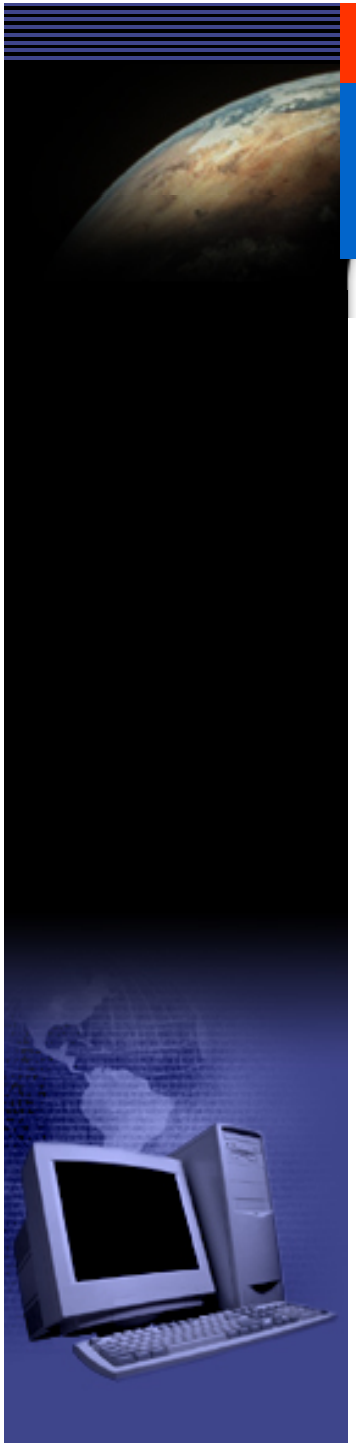
**- Địa chỉ vật lý=địa chỉ bắt đầu của trang + d.**

# Cấp phát không liên tục

- Phân trang
  - Chuyển đổi địa chỉ




Cơ chế phần cứng hỗ trợ phân trang

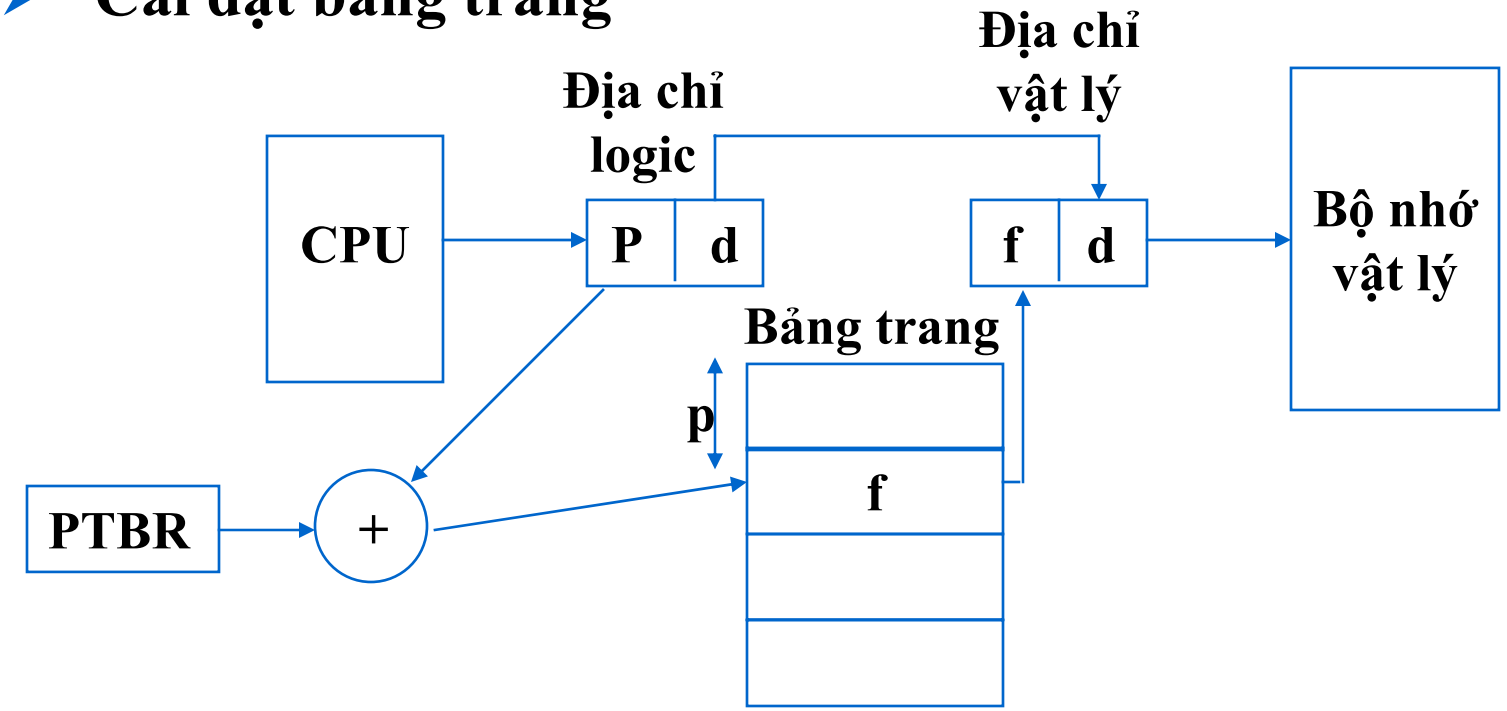


# Cấp phát không liên tục

- **Phân trang**
  - **Cài đặt bảng trang**
    - **Sử dụng tập các thanh ghi: bảng trang có kích thước nhỏ.**
    - **Lưu trữ trong bộ nhớ, sử dụng thanh ghi nền(PTBR) để lưu địa chỉ bắt đầu bảng trang. (Page Table Basic Register)**
    - **Sử dụng bộ nhớ kết hợp (TLB), mỗi thanh ghi trong bộ nhớ gồm: (Translation Lookaside Buffers)**
      - **từ khoá: số hiệu trang**
      - **giá trị: số hiệu khung trang**

# Cấp phát không liên tục

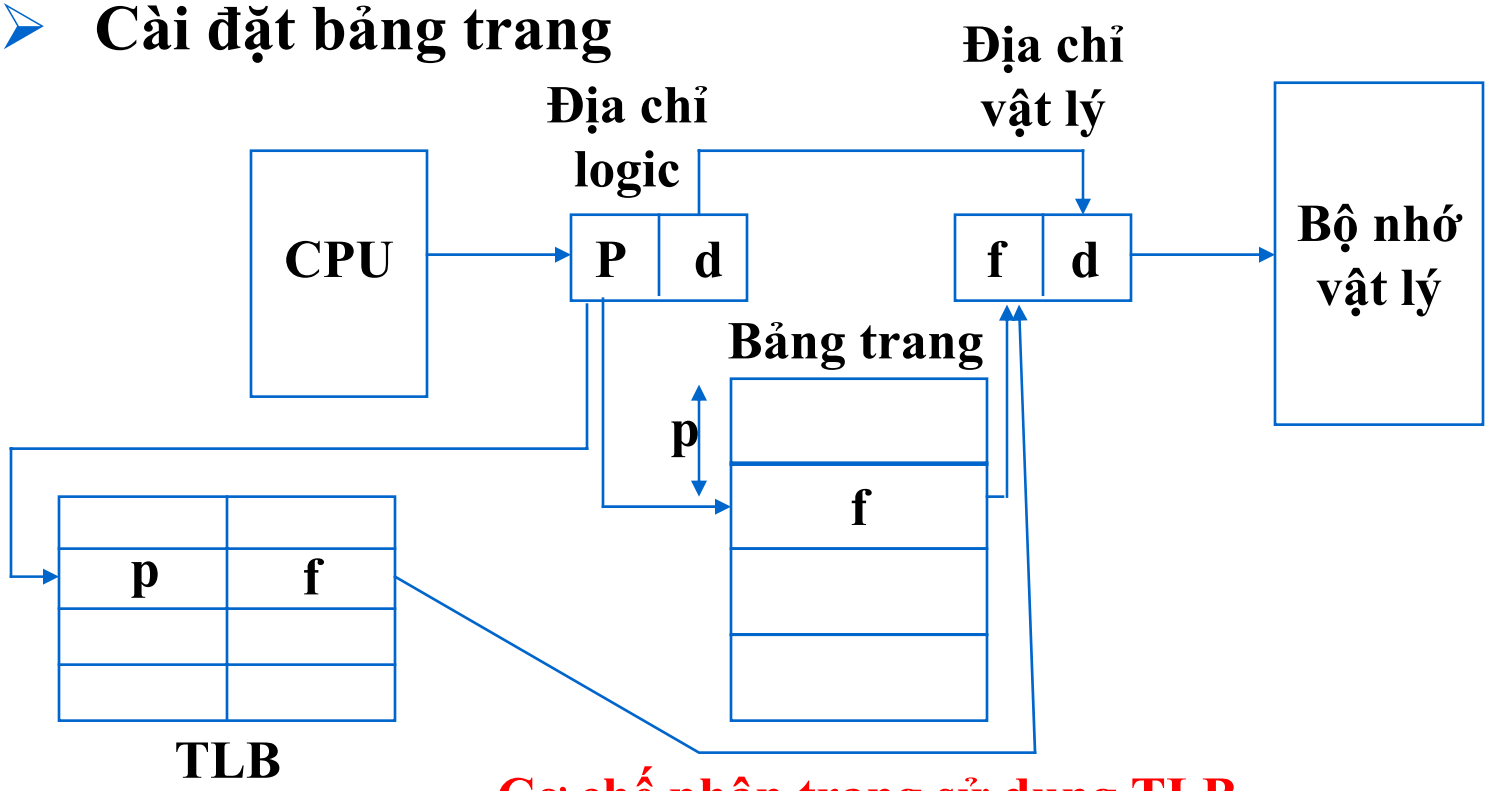
- Phân trang
  - Cài đặt bảng trang



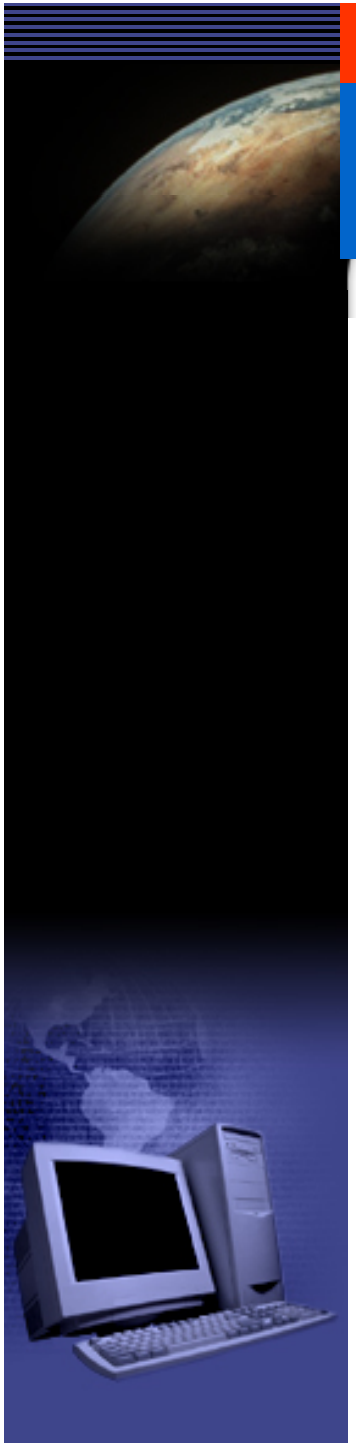

Cơ chế phân trang sử dụng PTBR

# Cấp phát không liên tục

- Phân trang
  - Cài đặt bảng trang




Cơ chế phân trang sử dụng TLB



# Cấp phát không liên tục

- **Phân trang**
- **Tổ chức bảng trang**

- **Mỗi HĐH có một cách tổ chức bảng trang. Đa số các HĐH cấp cho mỗi tiến trình một bảngtrang**
- **Nếu không gian địa chỉ có dung lượng quá lớn. Bảng trang đòi hỏi một vùng nhớ quá lớn. Có 2 giải pháp:**
  - **Phân trang đa cấp.**

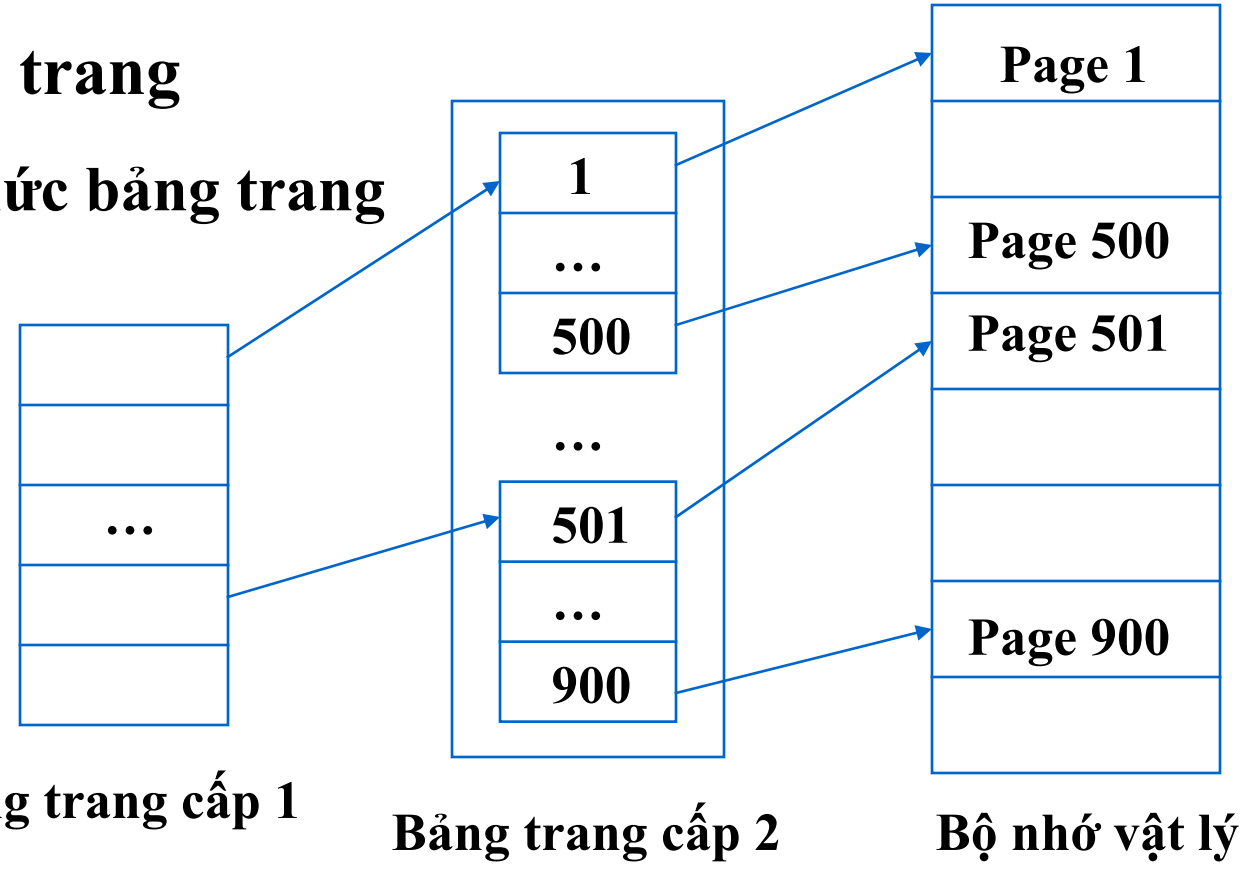
# Cấp phát không liên tục

- **Phân trang**
- **Tổ chức bảng trang**
- **Phân trang đa cấp**

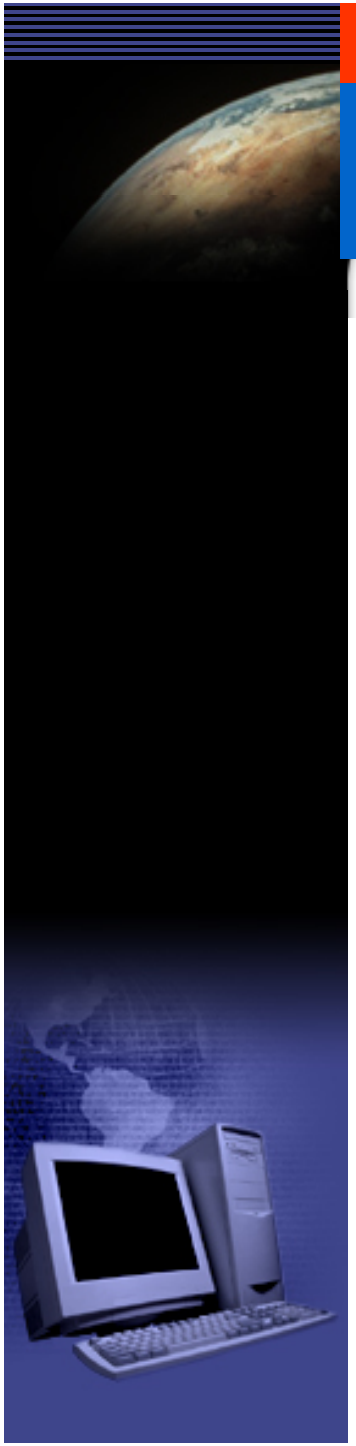
  **Phân chia bảng trang thành các phần nhỏ, bản thân bảng trang cũng sẽ được phân trang**

# Cấp phát không liên tục

- Phân trang
  - Tổ chức bảng trang

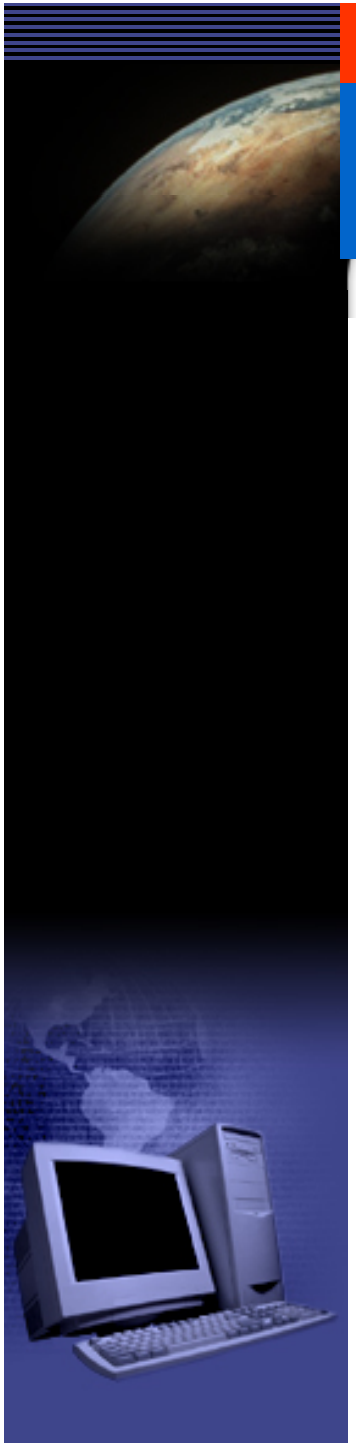



Bảng trang nhị cấp



# Cấp phát không liên tục

- **Phân đoạn**
  - **Ý tưởng**
  - **Cơ chế MMU**
  - **Chuyển đổi địa chỉ**
  - **Cài đặt bảng phân đoạn**
  - **Chia sẻ phân đoạn**



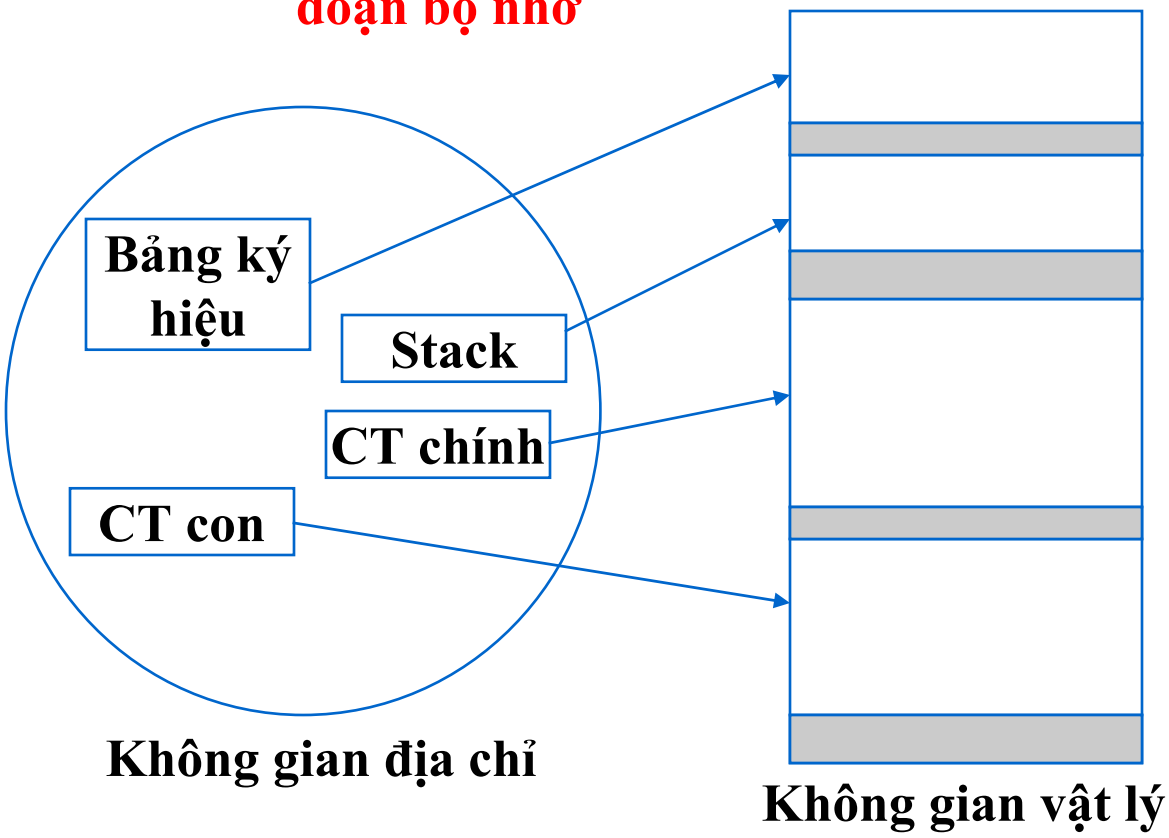
# Cấp phát không liên tục

- **Phân đoạn**
  - **Ý tưởng**
    - **Không gian địa chỉ: tập các phân đoạn(segments) có kích thước khác nhau, có liên hệ logic với nhau**
    - **Mỗi phân đoạn: <số hiệu, độ dài>**
    - **Mỗi địa chỉ logic: <số hiệu phân đoạn, offset>**

# Cấp phát không liên tục

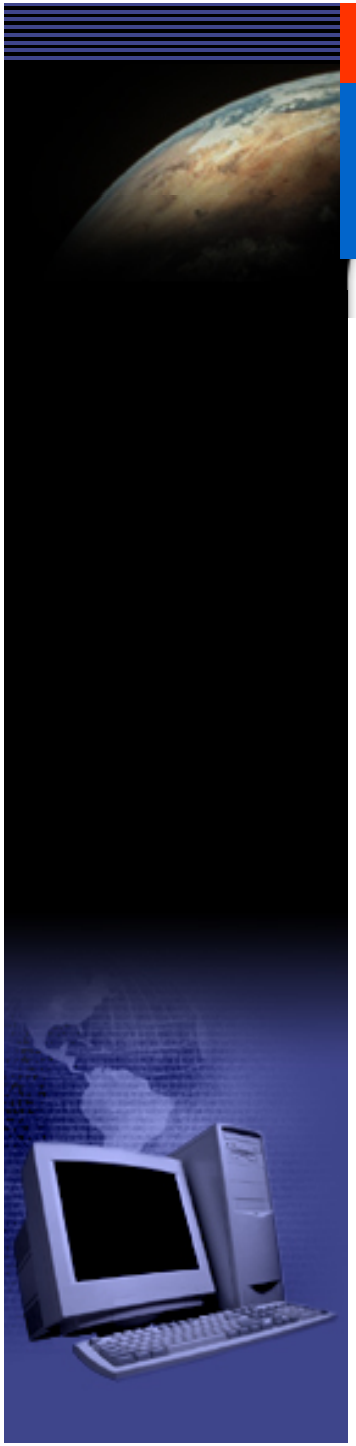
- Phân đoạn
  - Ý tưởng

Mô hình phân đoạn bộ nhớ



# Cấp phát không liên tục

- **Phân đoạn**
- **Cơ chế MMU**
- **Sử dụng bảng phân đoạn:**
  - **Thanh ghi nền: địa chỉ vật lý nơi bắt đầu của phân đoạn**
  - **Thanh ghi giới hạn: chiều dài của phân đoạn**

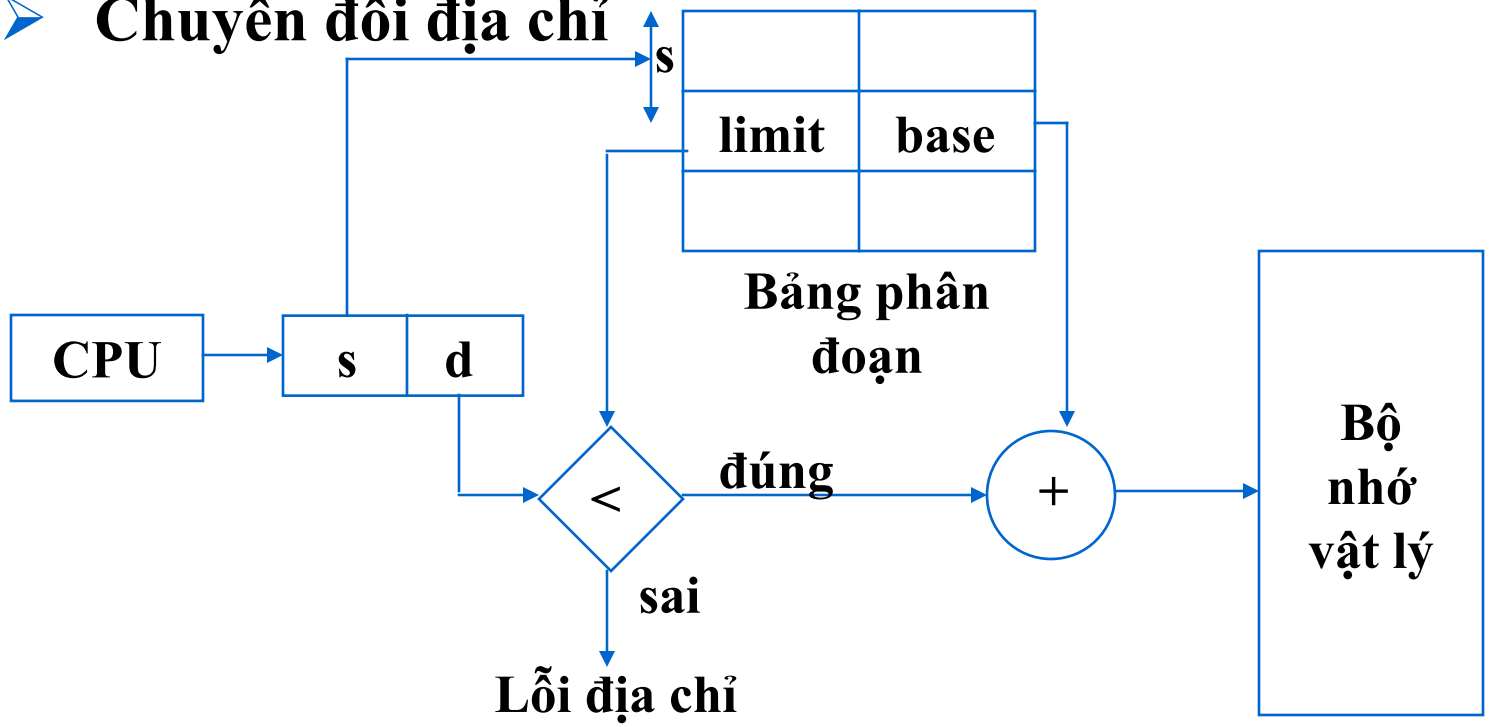


# Cấp phát không liên tục

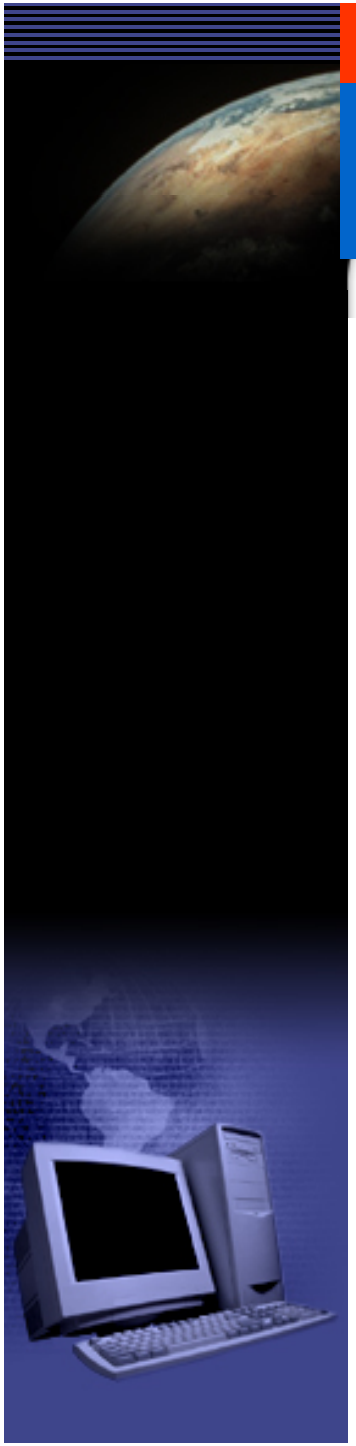
- **Phân đoạn**
  - **Chuyển đổi địa chỉ**
    - **Mỗi địa chỉ logic: <s,d>**
      - **s: số hiệu phân đoạn**
      - **d: địa chỉ tương đối offset, có giá trị từ 0 đến độ dài phân đoạn.**
    - **Địa chỉ vật lý=d+ giá trị chứa trong thanh ghi nền**

# Cấp phát không liên tục

- Phân đoạn
  - Chuyển đổi địa chỉ




Cơ chế phần cứng hỗ trợ kỹ thuật phân đoạn



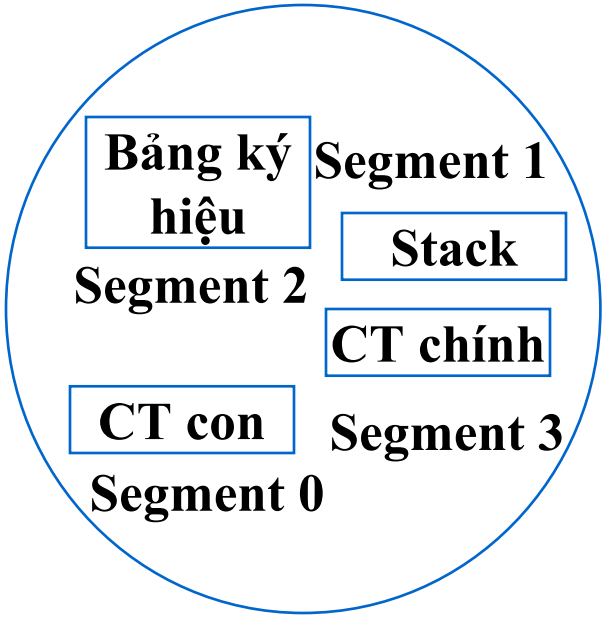
# Cấp phát không liên tục

- **Phân đoạn**
  - **Cài đặt bảng phân đoạn**
    - **Sử dụng tập các thanh ghi: bảng phân đoạn có kích thước nhỏ.**
    - **Lưu trữ trong bộ nhớ: bảng phân đoạn có kích thước lớn**
      - **Thanh ghi nền bảng phân đoạn (STBR) để lưu địa chỉ bắt đầu bảng phân đoạn (Segment Table Basic Register)**
      - **Thanh ghi đặc tả kích thước bảng phân đoạn (STLR)**

# Cấp phát không liên tục

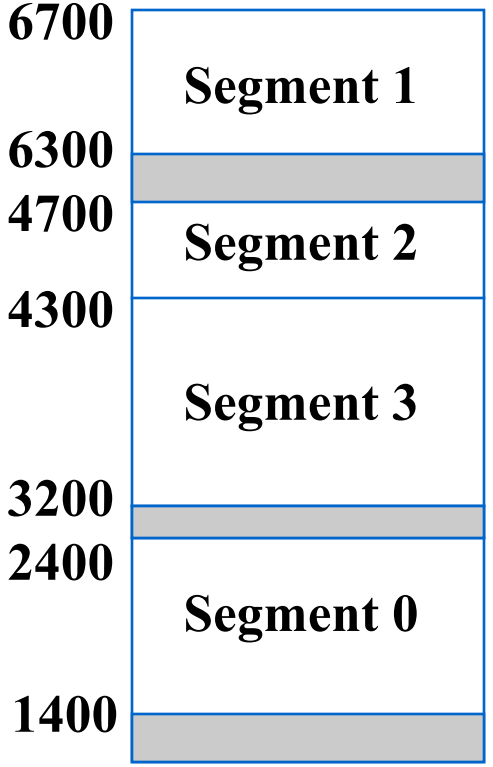
## Phân đoạn

### Cài đặt bảng phân đoạn




| limit | base |
|---|---|
| 1000 | 1400 |
| 400 | 6300 |
| 400 | 4300 |
| 1100 | 3200 |

Hệ thống phân đoạn



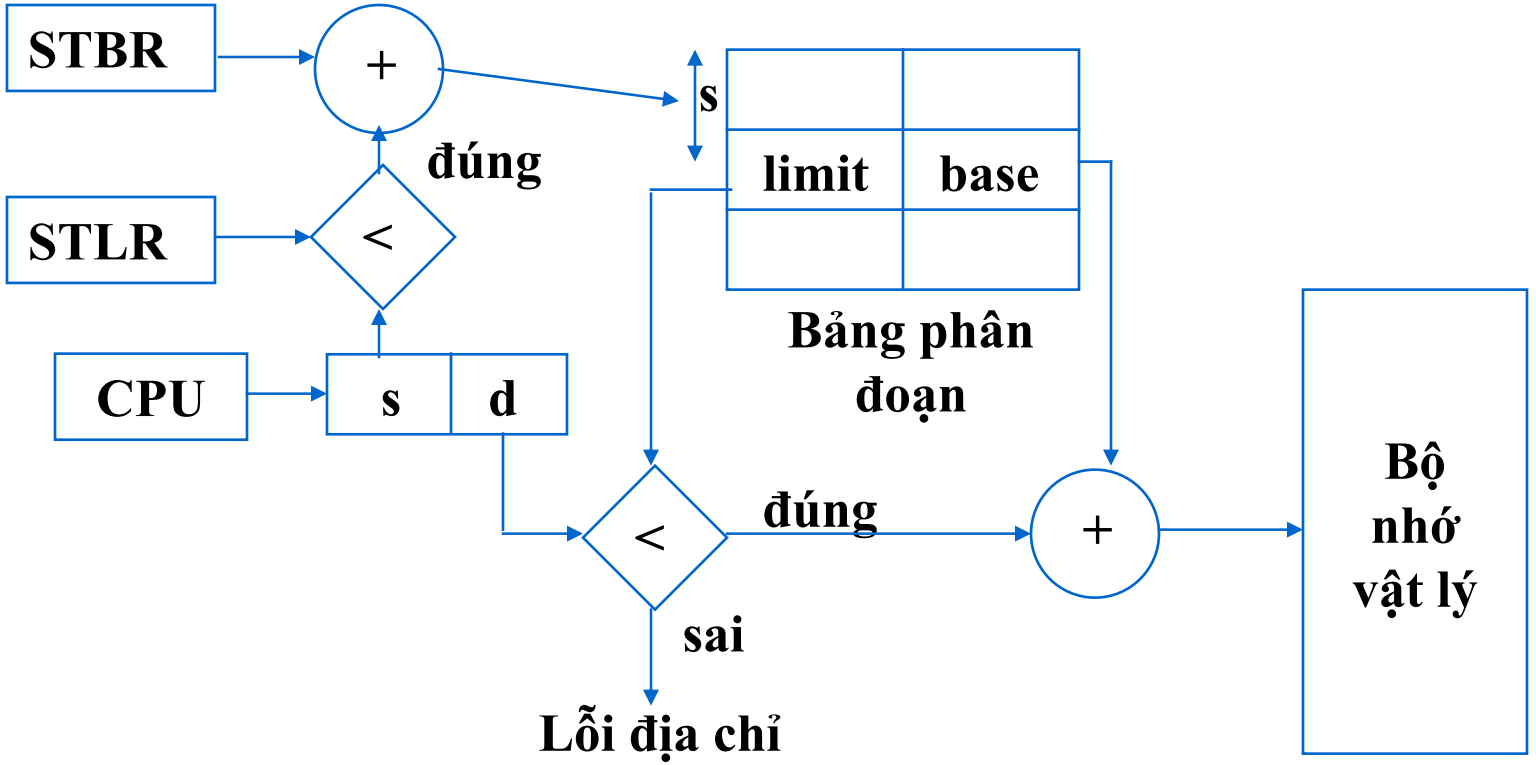

Không gian địa chỉ

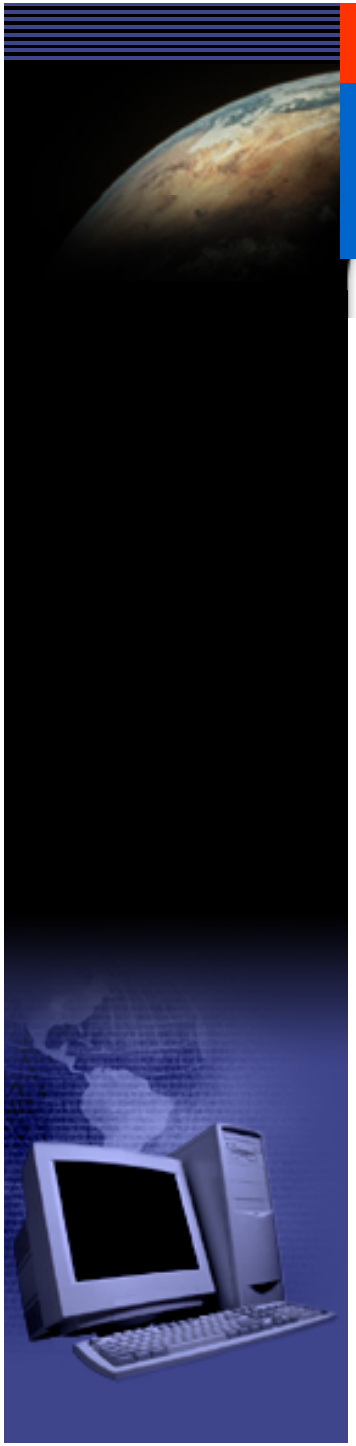
Không gian vật lý

# Cấp phát không liên tục

## Phân đoạn

### Cài đặt bảng phân đoạn





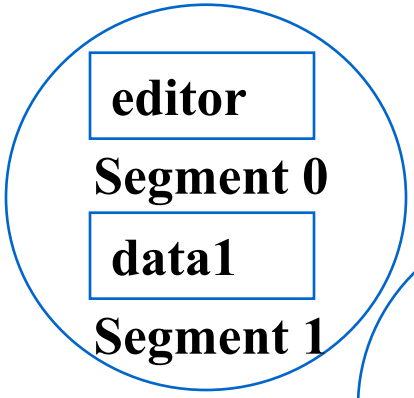
# Cấp phát không liên tục

- **Phân đoạn**
  - **Chia sẻ phân đoạn**
    - **Khả năng chia xẻ ở mức phân đoạn: chia xẻ các chương trình con.**
    - **Mỗi tiến trình có một bảng phân đoạn riêng.**
    - **Một phân đoạn được chia xẻ khi các phần tử trong bảng phân đoạn của hai tiến trình khác nhau cùng truy xuất đến một địa chỉ vật lý giống nhau**

# Cấp phát không liên tục

- Phân đoạn
- Chia sẻ phân đoạn

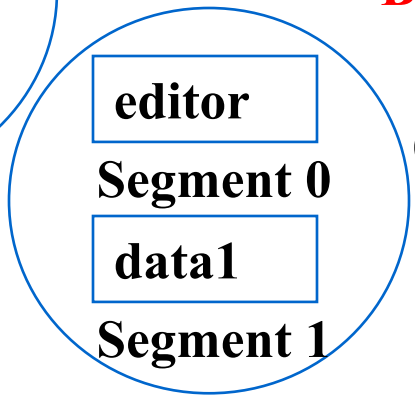
Không gian địa chỉ p1




Bảng phân đoạn p1

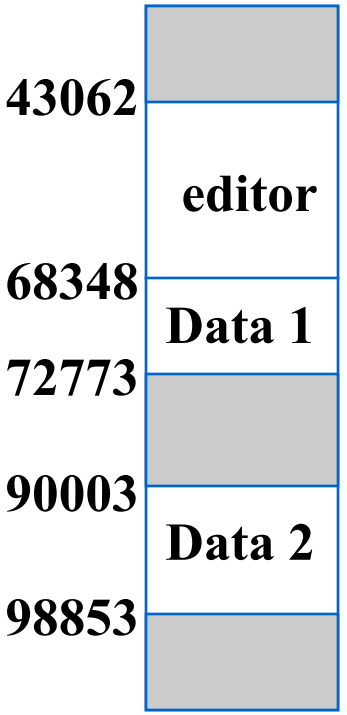
| | limit | base |
|---|---|---|
| 0 | 25286 | 43062 |
| 1 | 4425 | 68348 |

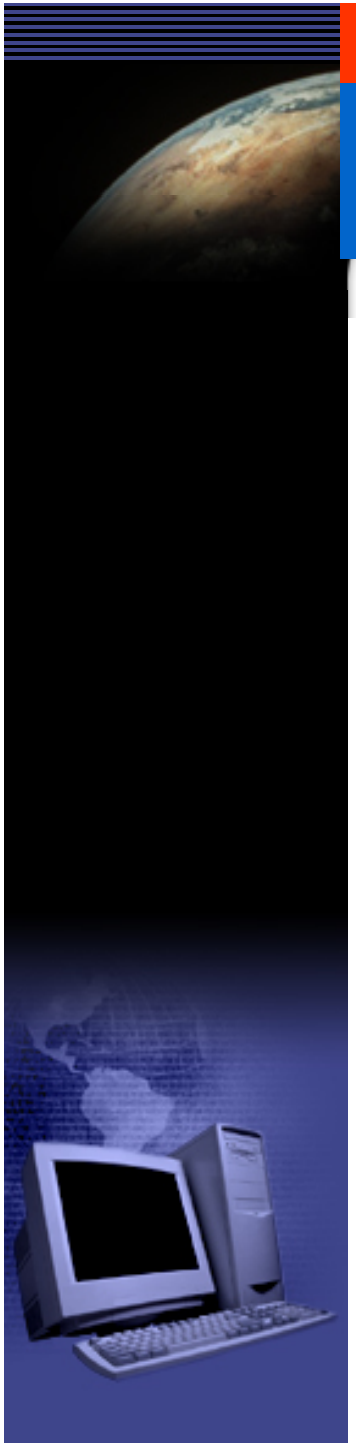
Không gian địa chỉ p2




Bảng phân đoạn p2

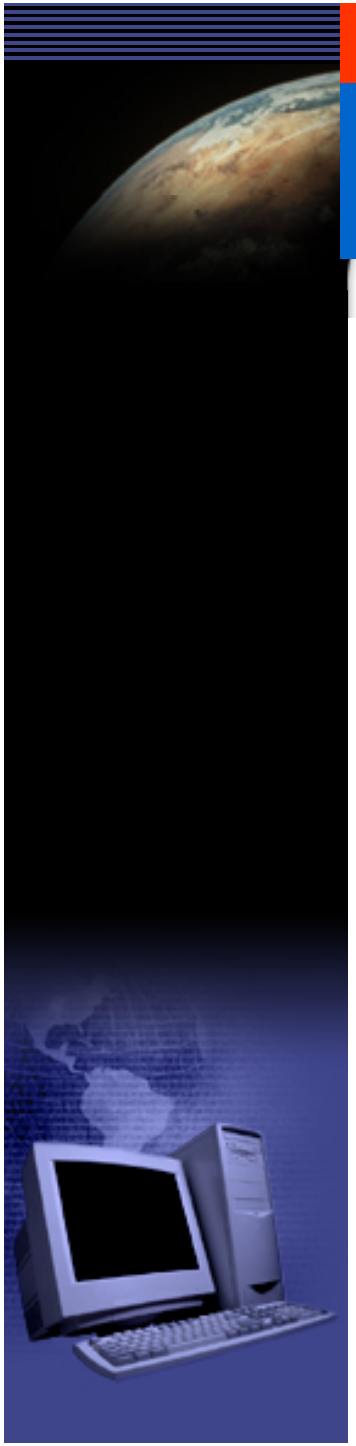
| | limit | base |
|---|---|---|
| 0 | 25286 | 43062 |
| 1 | 8850 | 90003 |





# Cấp phát không liên tục

- **Phân đoạn kết hợp phân trang**
  - **Ý tưởng**
  - **Cơ chế MMU**
  - **Chuyển đổi địa chỉ**

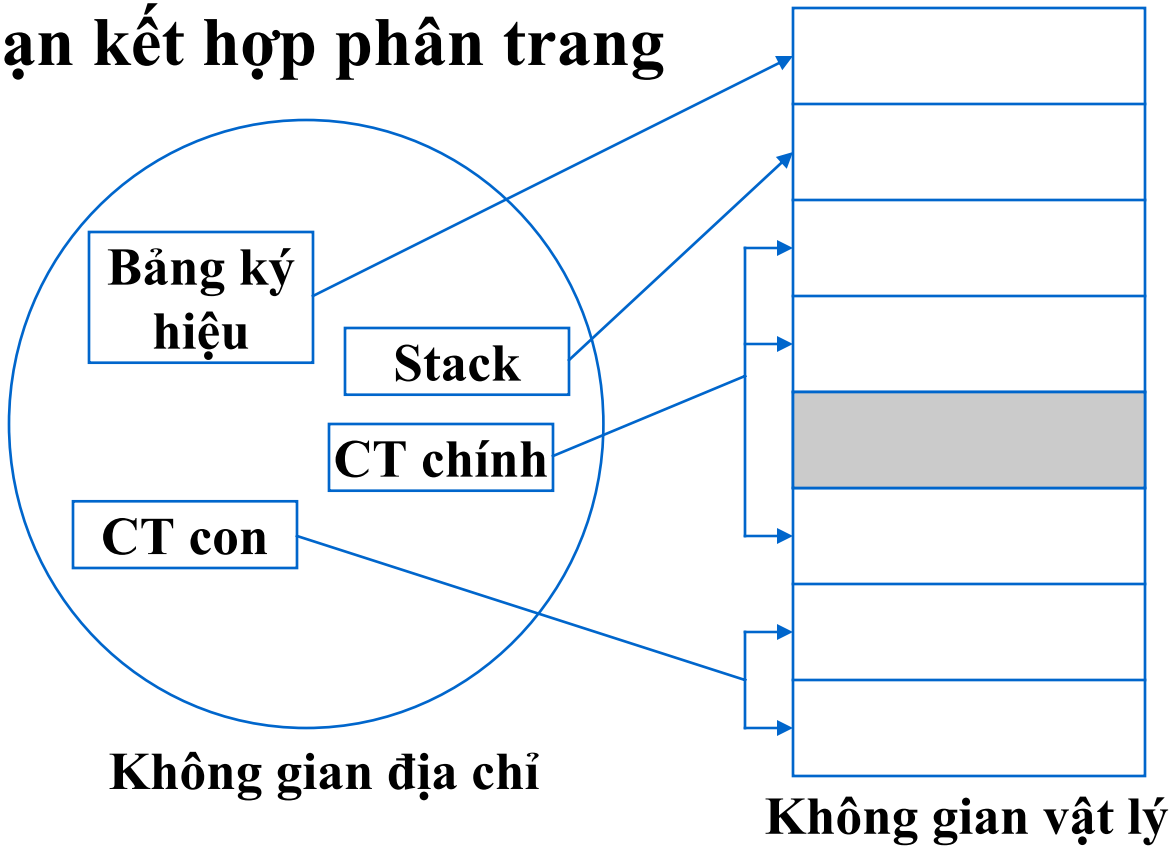


# Cấp phát không liên tục

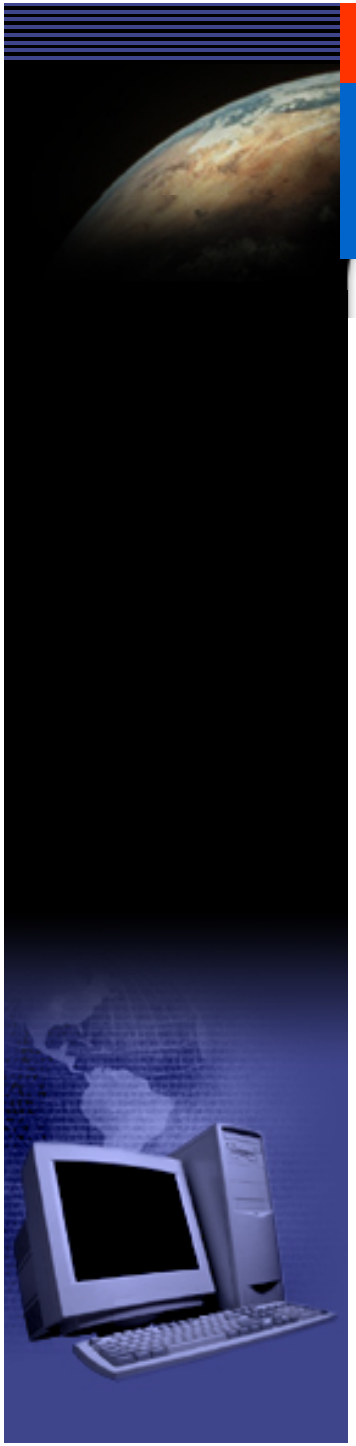
- **Phân đoạn kết hợp phân trang**
  - **Ý tưởng**
    - **Không gian địa chỉ: tập hợp các phân đoạn.**
    - **Mỗi phân đoạn: chia thành nhiều**
    - **Tiến trình được đưa vào hệ thống, HĐH sẽ cấp phát cho tiến trình các trang cần thiết để chứa đủ các phân đoạn của tiến trình**

# Cấp phát không liên tục

- Phân đoạn kết hợp phân trang
  - Ý tưởng




Mô hình phân đoạn kết hợp phân trang

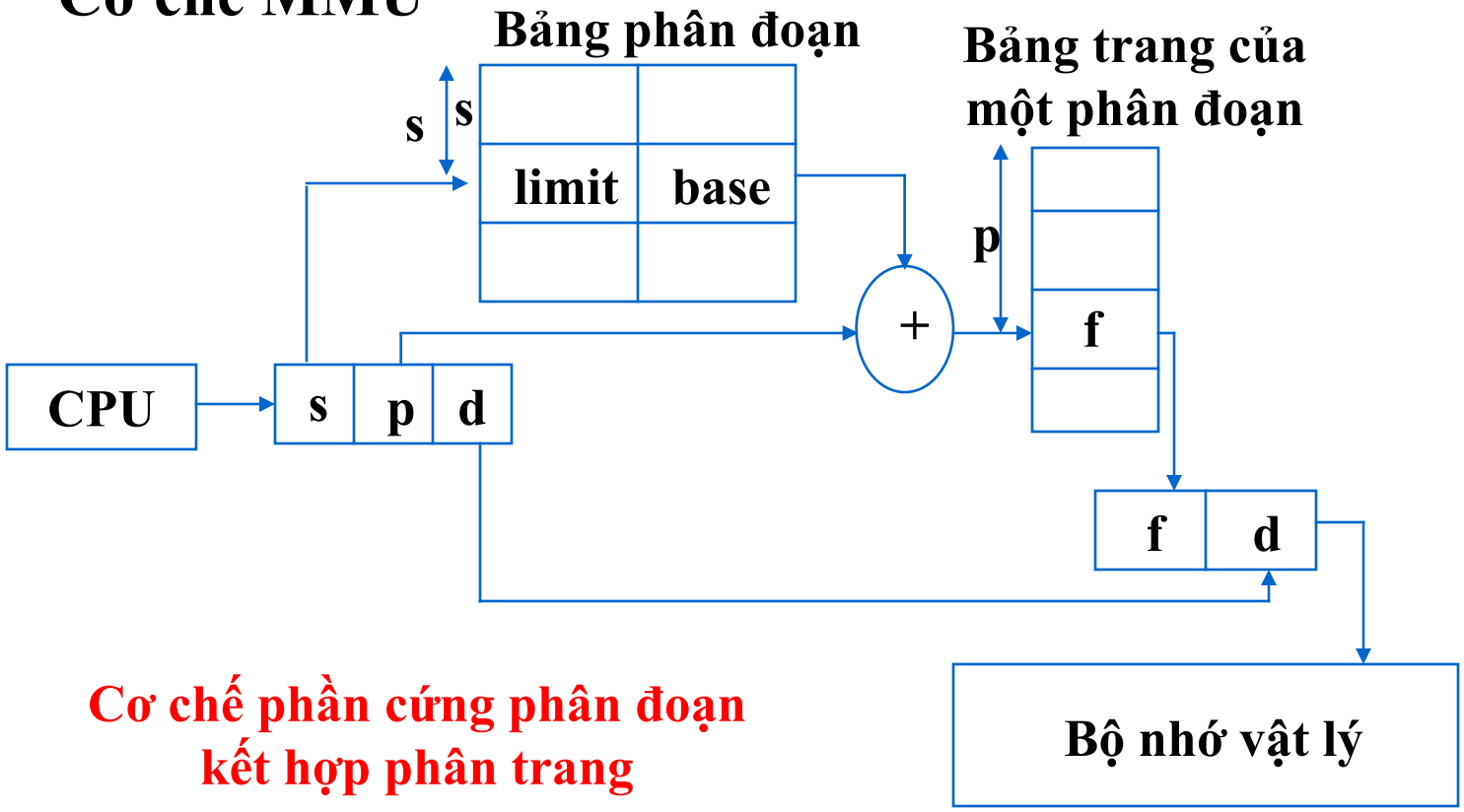


# Cấp phát không liên tục

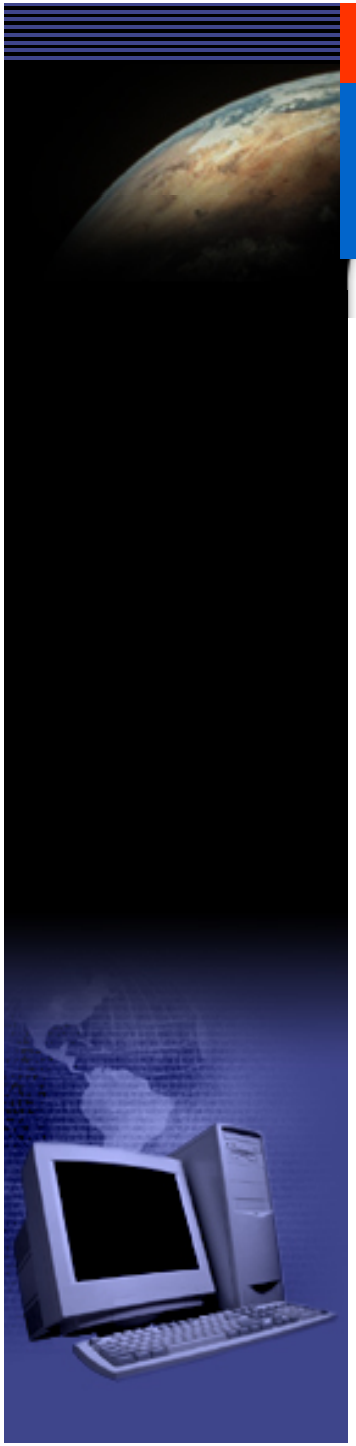
- **Phân đoạn kết hợp phân trang**
- **Chuyển đổi địa chỉ**
  - **Mỗi địa chỉ: <s, p, d>**
    - **S: số hiệu phân đoạn**
    - **P: số hiệu trang**
    - **D: địa chỉ tương đối**

# Cấp phát không liên tục

- Phân đoạn kết hợp phân trang
  - Cơ chế MMU

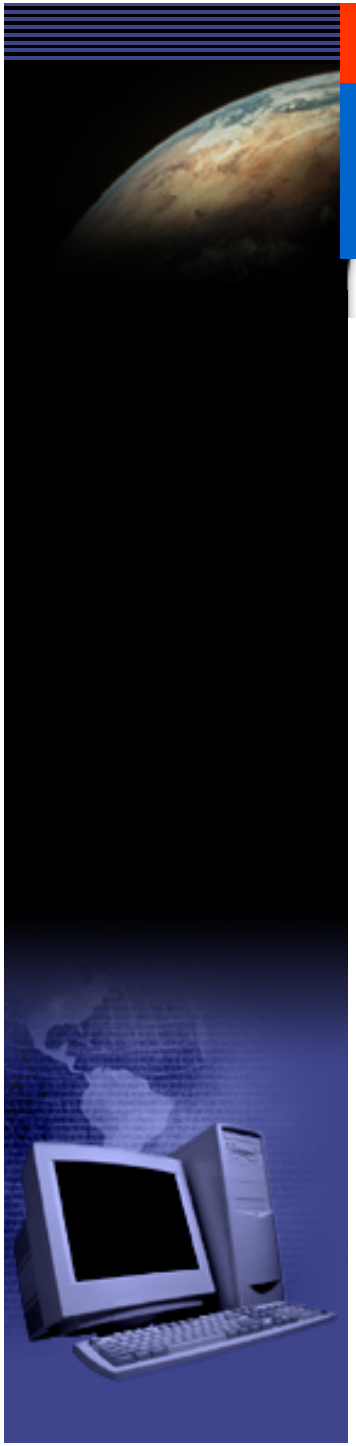



Cơ chế phần cứng phân đoạn kết hợp phân trang



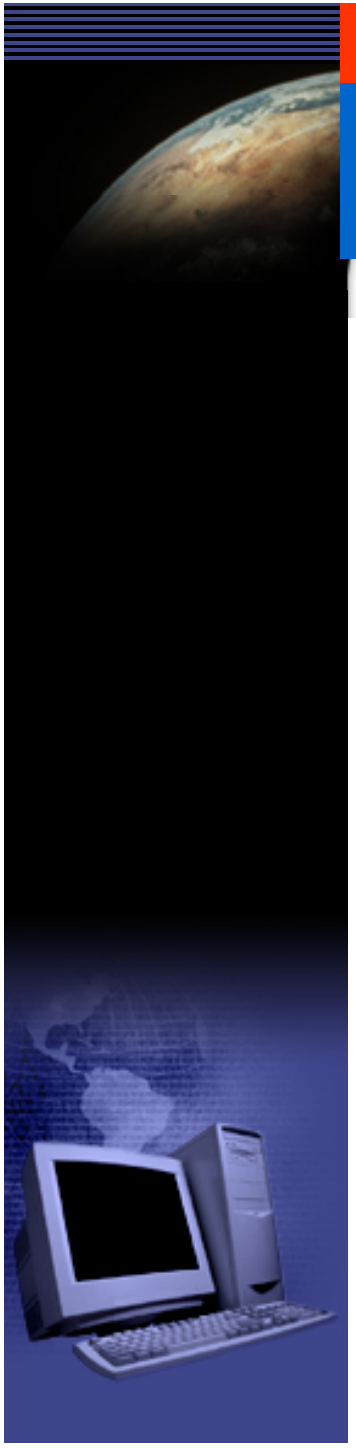
# Bộ nhớ ảo

- **Nếu đặt toàn thể không gian địa chỉ vào bộ nhớ vật lý thì kích thước của chương trình bị giới hạn bởi kích thước bộ nhớ.**
- **Nạp từng phần của chương trình.**
- **Tại một thời điểm, chỉ nạp vào bộ nhớ vật lý các chỉ thị và dữ liệu của ct cần thiết cho việc thi hành lệnh ở thời điểm đó.**



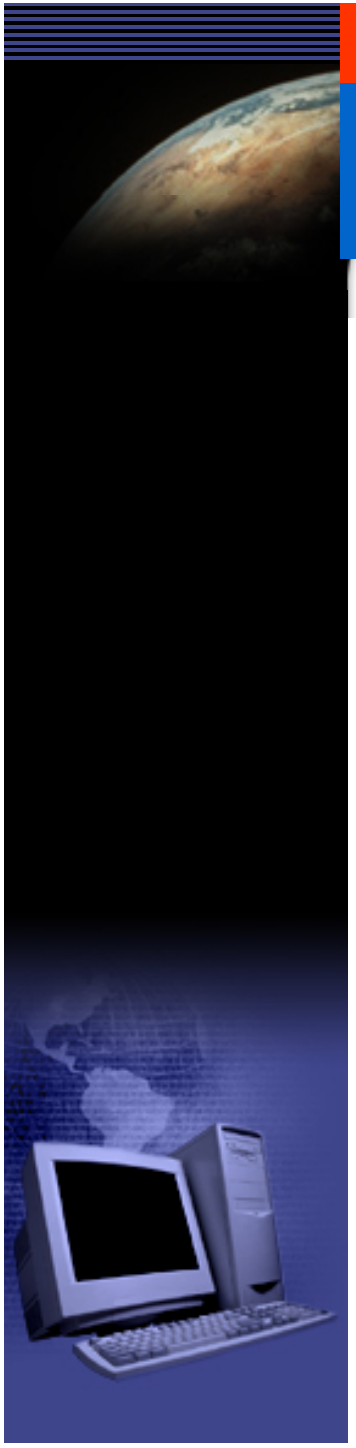
# Bộ nhớ ảo

- **Bộ nhớ ảo: kỹ thuật cho phép xử lý một tiến trình k0 được nạp toàn bộ vào bộ nhớ vật lý.**
- **Bộ nhớ ảo: mô hình hoá bộ nhớ như một bảng lưu trữ rất lớn và đông nhất.**
- **NSD làm việc với địa chỉ ảo. Việc chuyển đổi sang địa chỉ vật lý do HĐH đảm nhiệm bằng cơ chế phần cứng**



# Mô hình Client-Server

- Hệ thống nguyên khối (Monolithic System)
- Hệ thống phân lớp (Layer System)
- Máy ảo (Virtual Machine)
- Mô hình Client-Server (Client-Server Model)



# Mô hình Client-Server

- Hệ thống nguyên khối (Monolithic System)
- Hệ thống phân lớp (Layer System)
- Máy ảo (Virtual Machine)
- Mô hình Client-Server (Client-Server Model)