

# NGUYÊN LÝ HỆ ĐIỀU HÀNH

Giảng viên: TS. Đoàn Thị Quế  
Bộ môn Mạng và an toàn thông tin



# Chương 4

## **QUẢN LÝ BỘ NHỚ**

# Nội dung

- Giới thiệu về quản lý bộ nhớ
- Bộ nhớ và chương trình
- Phân chương bộ nhớ
- Phân trang bộ nhớ
- Phân đoạn bộ nhớ
- Quản lý bộ nhớ trên IBM - PC

## 4.1 - Giới thiệu về quản lý bộ nhớ

- Bộ nhớ là tài nguyên quan trọng để thi hành chương trình.
- Muốn thi hành một chương trình thì mã lệnh và dữ liệu của nó phải được nạp vào bộ nhớ.
- Cách thức tổ chức và quản lý bộ nhớ sẽ ảnh hưởng tới tốc độ và hiệu quả thi hành chương trình.

## 4.1 - Giới thiệu về quản lý bộ nhớ

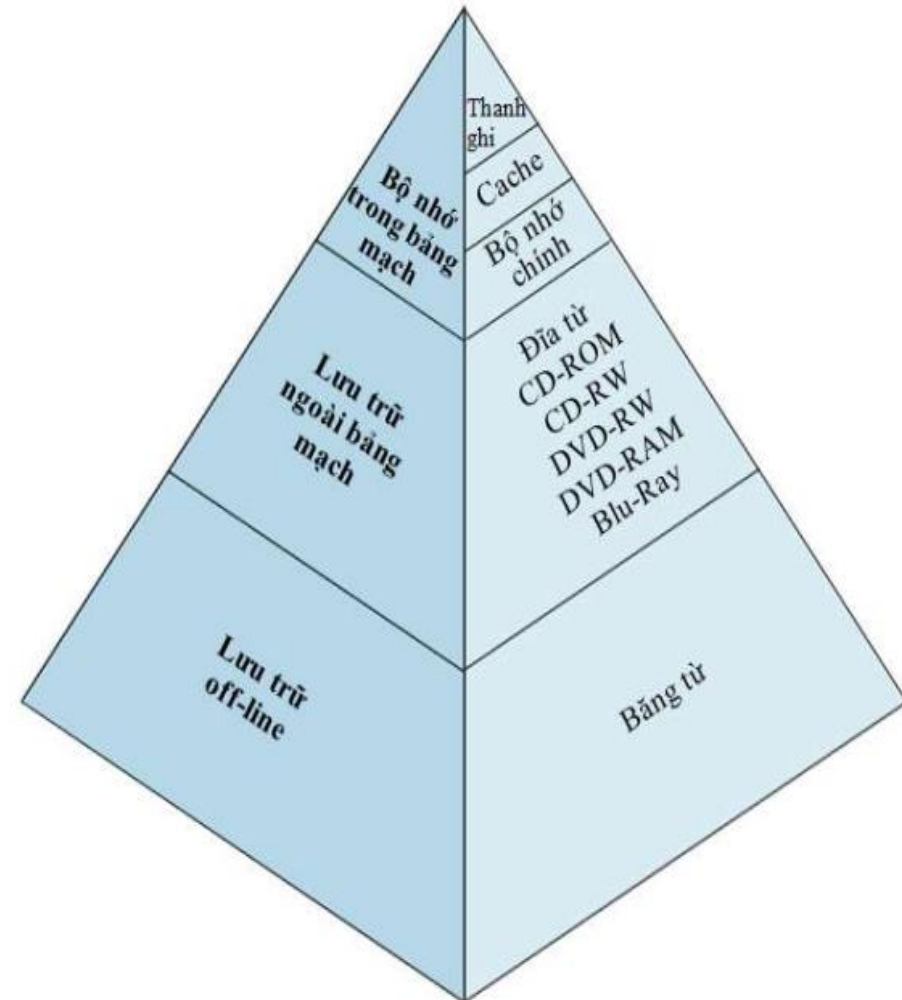
- Quản lý bộ nhớ gồm:
  - Cấp phát bộ nhớ trống cho các tiến trình và giải phóng bộ nhớ đã cấp phát
  - Ánh xạ giữa địa chỉ logic và địa chỉ vật lý
  - Ngăn chặn việc truy cập trái phép tới các vùng bộ nhớ

## 4.2 – Bộ nhớ và chương trình

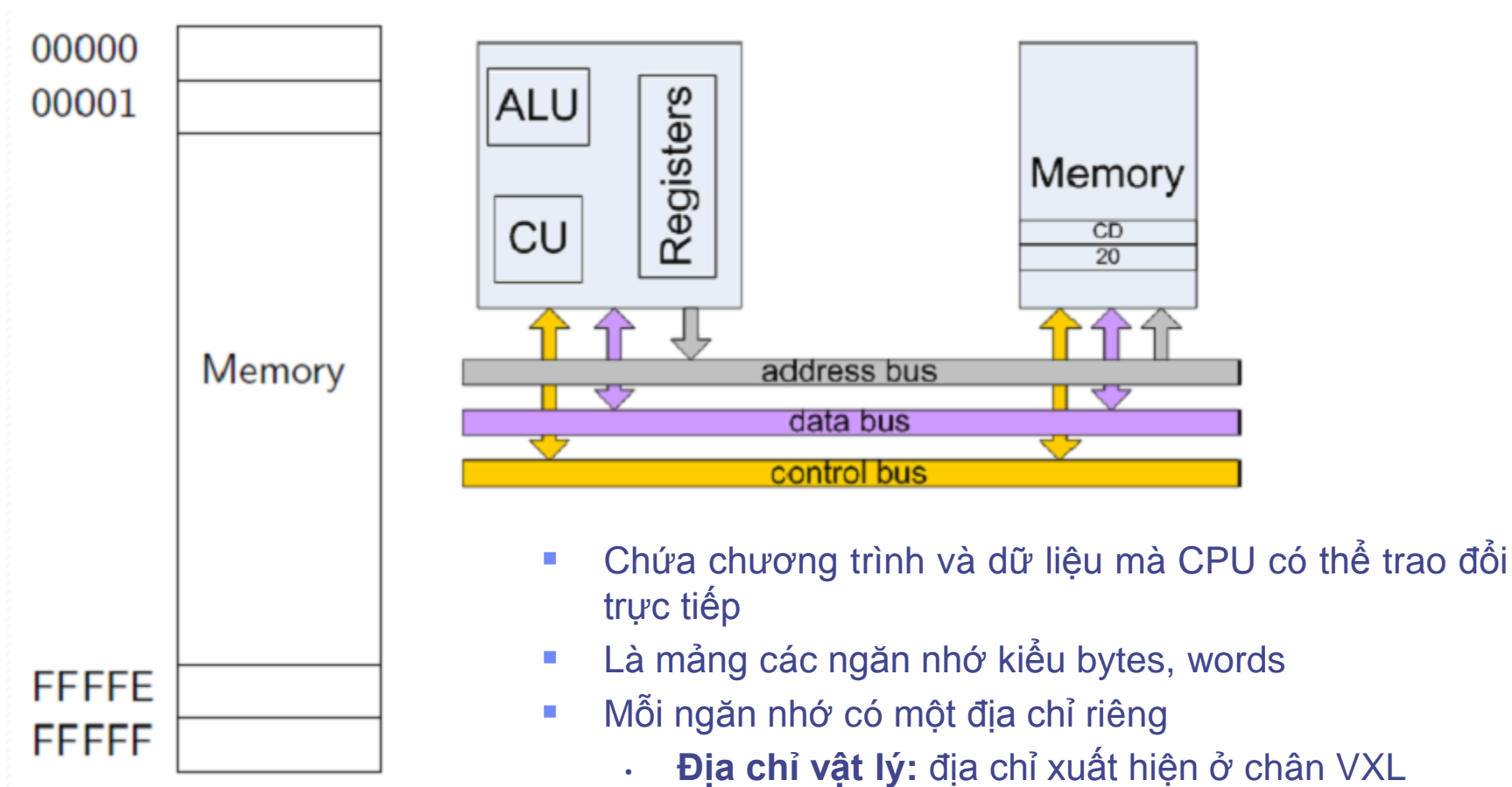
- Tổng quan về bộ nhớ
- Các bước thực hiện một chương trình
- Địa chỉ logic và địa chỉ vật lý

# Tổng quan về bộ nhớ

- Bộ nhớ dùng lưu trữ chương trình và dữ liệu
- Bộ nhớ được đặc trưng bởi tốc độ truy nhập và dung lượng
- Hệ thống nhớ máy tính được phân cấp theo tốc độ và dung lượng

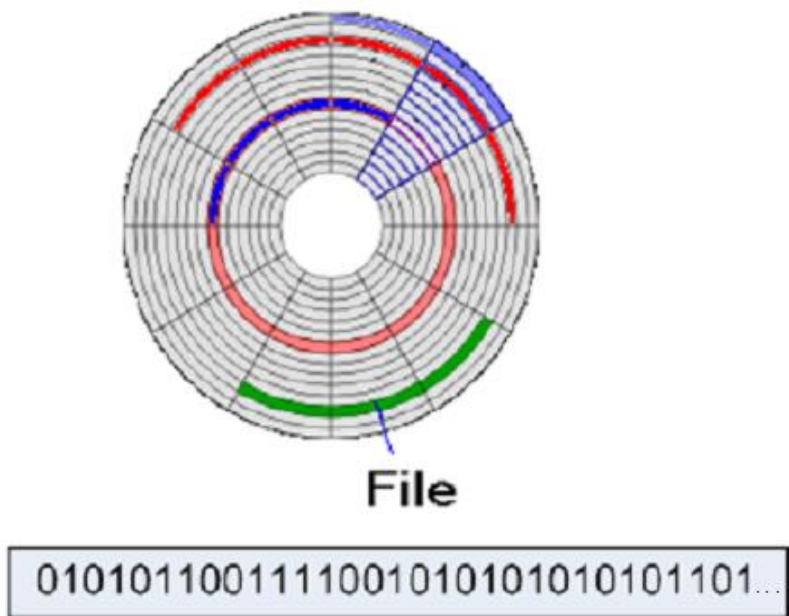


# Bộ nhớ chính



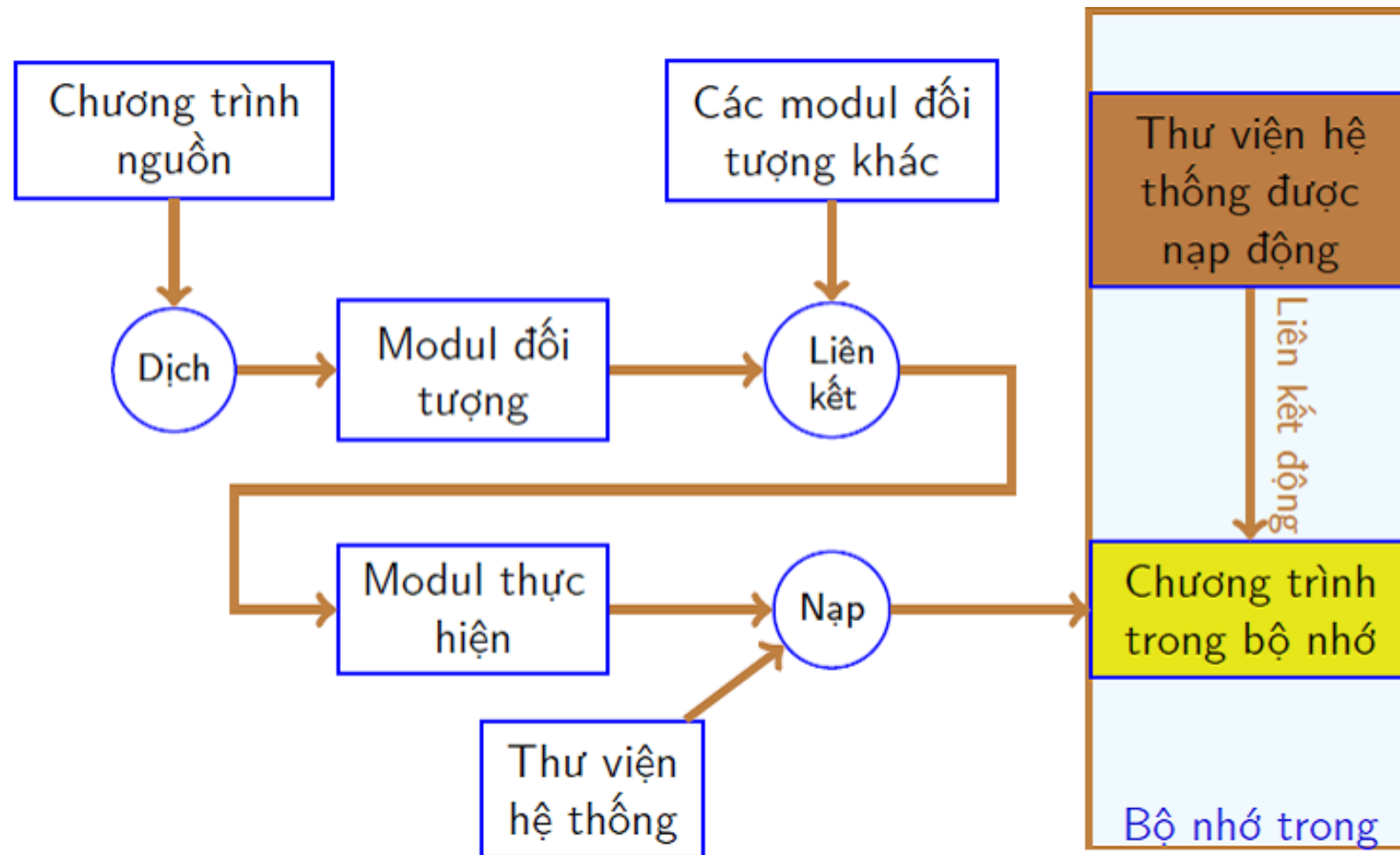


# Chương trình



- Tồn tại trên thiết bị lưu trữ ngoài
- Là các file nhị phân thực thi được
  - Mã lệnh
  - Dữ liệu
  - Ngăn xếp
- Phải được đưa vào bộ nhớ trong để thực hiện

# Các bước thực hiện chương trình



- *Bước 1: Dịch*
- *Bước 2: Biên tập*
- *Bước 3: Thi hành*

# Bước 1: Dịch

- Dịch các modul chương trình (từ ngôn ngữ thuật toán) sang ngôn ngữ máy (nhị phân), bao gồm:
  - Chuyển đổi các tên biến sang địa chỉ ô nhớ logic (tức là địa chỉ tương đối của ô nhớ trong đoạn dữ liệu của chương trình)
  - Chuyển đổi các câu lệnh sang mã máy

## Bước 2: Biên tập

- Liên kết các modul đã dịch để tạo thành một chương trình hoàn chỉnh

# Bước 3: Thi hành

- Nạp chương trình vào bộ nhớ vật lý cụ thể, chuyển đổi các địa chỉ logic sang địa chỉ vật lý
  - Việc tổ chức, sắp xếp các đoạn chương trình trong bộ nhớ có ảnh hưởng rất lớn tới tốc độ thi hành chương trình và hiệu quả sử dụng bộ nhớ
- Thực thi chương trình
  - CPU lấy các lệnh trong bộ nhớ tại vị trí được xác định bởi bộ đếm chương trình (*Program counter*)
    - ◆ Cặp thanh ghi CS:IP với VXL họ Intel
  - CPU giải mã lệnh
  - CPU có thể lấy thêm toán hạng từ bộ nhớ
  - Thực hiện lệnh với toán hạng
  - Nếu cần thiết, lưu kết quả vào bộ nhớ tại một địa chỉ xác định
- Thực hiện xong
  - Giải phóng vùng không gian nhớ dành cho chương trình

# Địa chỉ logic và địa chỉ vật lý

## ■ Địa chỉ logic

- Còn gọi là địa chỉ tương đối
- Địa chỉ logic là địa chỉ được gán cho các lệnh và dữ liệu không phụ thuộc vào vị trí cụ thể của tiến trình trong bộ nhớ
- CPU sử dụng địa chỉ logic để trở đến các phần khác nhau của lệnh, dữ liệu

## ■ Địa chỉ vật lý

- Còn gọi là địa chỉ tuyệt đối
- Là địa chỉ của ngăn nhớ trong bộ nhớ
- Để truy cập bộ nhớ, địa chỉ logic cần được biến đổi thành địa chỉ vật lý

## ■ Chuyển đổi địa chỉ logic sang địa chỉ vật lý

- Địa chỉ logic được ánh xạ sang địa chỉ vật lý nhờ khối ánh xạ bộ nhớ (*MMU=Memory Management Unit*)

## 4.3 – Phân chương bộ nhớ

- Phân chương bộ nhớ:
  - Bộ nhớ được chia thành các phần liên tục gọi là chương (*partition*)
  - Mỗi tiến trình sẽ được cung cấp một chương để chứa lệnh và dữ liệu
    - ♦  $\Rightarrow$  Tiến trình được cung cấp một vùng nhớ liên tục
- Tùy vào việc lựa chọn vị trí và kích thước của chương, phân chương gồm:
  - Phân chương cố định
  - Phân chương động

# Phân chương cố định

- Bộ nhớ được chia thành  $n$  phần
    - Mỗi phần gọi là một chương (*partition*)
    - Mỗi chương có kích thước cố định ở những vị trí cố định
    - Các chương không nhất thiết có kích thước bằng nhau
    - Mỗi chương có thể dùng để nạp và chạy một tiến trình
      - ♦ Tại một thời điểm chỉ cho phép một tiến trình tồn tại
      - ♦ Các tiến trình nằm trong vùng nhớ cho tới khi kết thúc
- ⇒ Có thể chạy đồng thời nhiều tiến trình

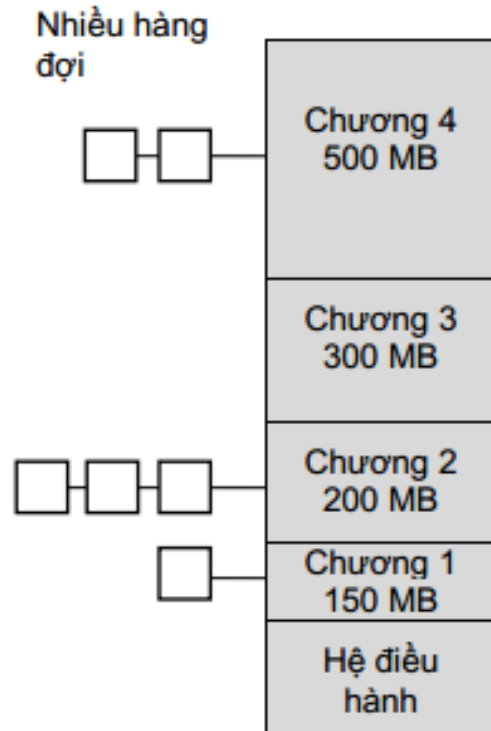


# Phân chương cố định

- Lựa chọn kích thước chương
  - Các chương có kích thước bằng nhau
    - Kích thước chương nhỏ
      - Tiến trình có kích thước lớn hơn kích thước chương sẽ không thể tải vào chương và chạy được
    - Kích thước của chương bằng kích thước của tiến trình lớn nhất
      - Chương chứa được các tiến trình lớn
      - Tiến trình có kích thước nhỏ  $\Rightarrow$  phần bỏ trống lớn gây lãng phí. Hiện tượng này gọi là *phân mảnh trong (internal fragmentation)*
  - Các chương có kích thước không bằng nhau
    - Cách lựa chọn chương nhớ để cấp cho tiến trình đang chờ đợi?

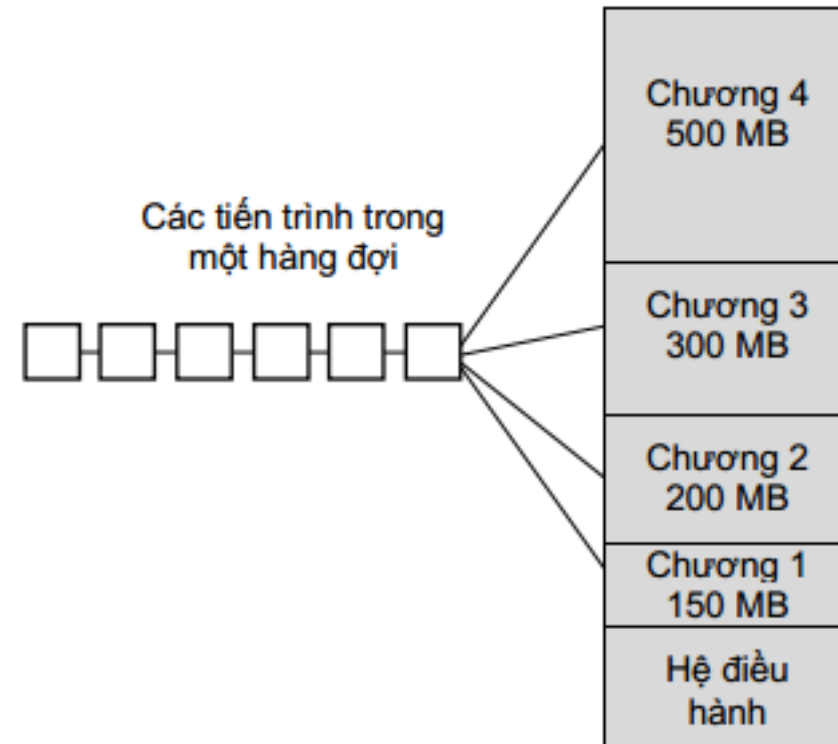
# Cấp phát bộ nhớ sử dụng phân chương cố định

(a) Mỗi chương có hàng đợi riêng



(a)

(b) Một hàng đợi chung cho tất cả các chương



(b)

# Nhận xét

- Đơn giản trong việc cấp phát bộ nhớ
- Hệ số song song không thể vượt quá  $n$
- Bị phân mảnh  $\Rightarrow$  bộ nhớ được sử dụng không hiệu quả

# Phân chương động

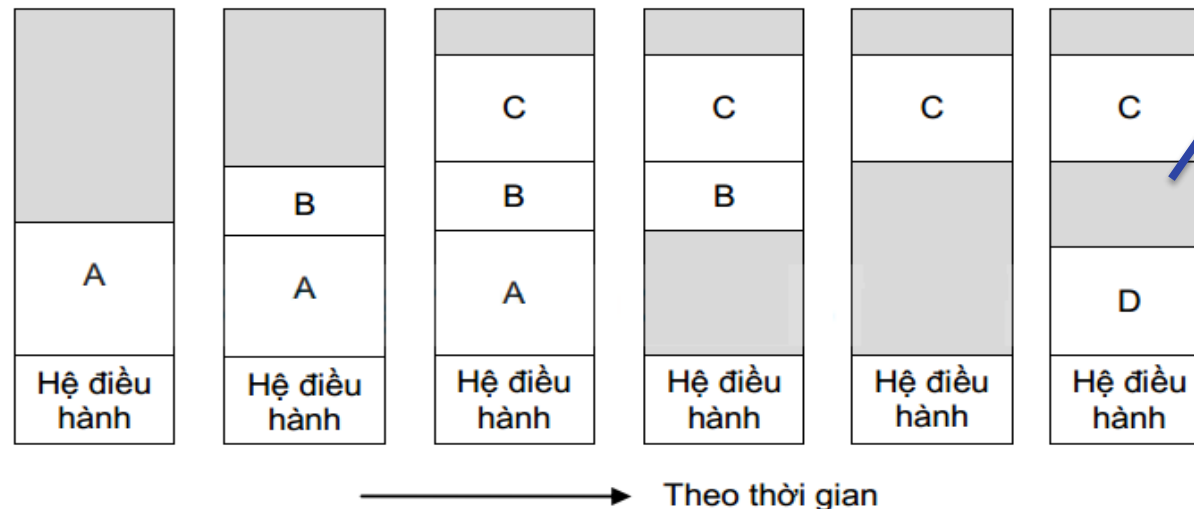
- Kích thước và số lượng chương đều không cố định
- Bộ nhớ được phân phối tùy theo kích thước tiến trình
  - Tiến trình được cấp một lượng bộ nhớ đúng bằng lượng bộ nhớ mà tiến trình cần
- Tiến trình nào thực hiện xong thì bộ nhớ của nó sẽ được giải phóng, dành chỗ cho tiến trình khác
- Tồn tại một tập hợp các vùng trống có kích thước khác nhau

# Cấp phát bộ nhớ sử dụng phân chương động

- Hệ điều hành sử dụng một danh sách để quản lý các vùng nhớ trống
- Các tiến trình được xếp vào hàng đợi để chờ đến lượt được cấp bộ nhớ
  - Khi tới lượt, tìm trong DS vùng trống một phần tử đủ lớn cho yêu cầu
  - Nếu tìm thấy:
    - ♦ Vùng trống được chia thành hai phần
    - ♦ Một phần cung cấp theo yêu cầu
    - ♦ Phần còn lại được bổ sung vào DS vùng trống
  - Nếu không tìm thấy
    - ♦ Chờ đợi cho tới khi một vùng bộ nhớ đủ lớn được giải phóng và cấp phát
    - ♦ Cho phép tiến trình khác trong hàng đợi được thực hiện (*nếu độ ưu tiên đảm bảo*)

# Cấp phát bộ nhớ sử dụng phân chương động (tiếp)

- Khi tiến trình kết thúc:
  - Tạo ra vùng trống mới và được trả về DS quản lý vùng trống
  - Kết hợp với các vùng trống liền kề để tạo ra vùng trống có kích thước lớn hơn (nếu có)
- Hiện tượng *phân mảnh ngoài*



*Kích thước quá nhỏ  
không thể cấp cho  
bất kỳ tiến trình nào*

⇒ giải quyết vấn đề phân mảnh ngoài: *dồn bộ nhớ*

# Các chiến lược chọn vùng trống

- Vùng thích hợp đầu tiên (first fit)
  - Vùng trống đầu tiên thỏa mãn
- Vùng thích hợp nhất (best fit)
  - Vùng trống vừa vặn nhất
- Vùng không thích hợp nhất (worst fit)
  - Vùng trống kích thước lớn nhất

# Các chiến lược chọn vùng trống

## ■ Ví dụ:

- Bộ nhớ có 4 vùng trống có kích thước lần lượt như sau 3MB, 8MB, 7MB và 10MB
- Yêu cầu cấp phát vùng nhớ kích thước 6MB

## ■ Cấp phát vùng nhớ nào?

- **First fit:** 8MB
- **Best fit:** 7MB
- **Worst fit:** 10MB



# Nhận xét về phân chương động

- Tăng/giảm hệ số song song tùy theo số lượng và kích thước chương trình
- Phức tạp trong việc quản lý vùng trống và lựa chọn vùng trống
- Gây ra phân mảnh ngoài

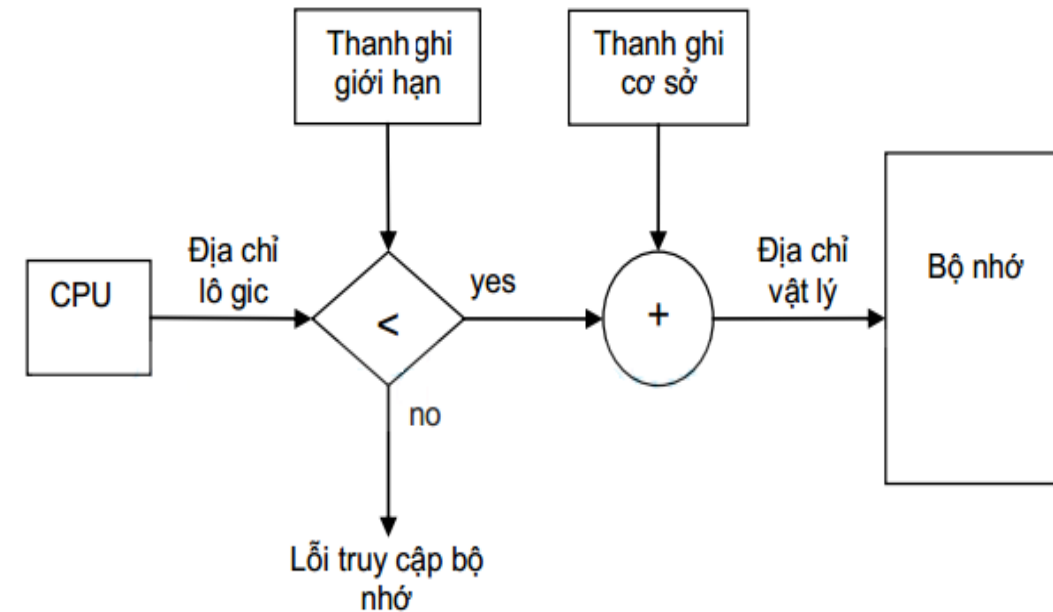
# Ánh xạ địa chỉ và chống truy cập trái phép trong phân chương

## ■ Hai vấn đề:

- **Ánh xạ địa chỉ:** khi vị trí các chương không được biết trước và có thể bị thay đổi
  - ◆ Ánh xạ địa chỉ logic của tiến trình thành địa chỉ vật lý
- **Truy cập trái phép vào vùng nhớ:** gây ra lỗi hoặc phá vỡ tính an toàn bảo mật của thông tin

## ■ Giải quyết hai vấn đề trên: sử dụng 2 thanh ghi

- Thanh ghi cơ sở chứa địa chỉ bắt đầu của tiến trình trong bộ nhớ.
- Thanh ghi giới hạn chứa giới hạn địa chỉ logic của tiến trình tức độ dài chương chứa tiến trình



## 4.4 - Phân trang bộ nhớ

- Khái niệm phân trang bộ nhớ
- Ánh xạ địa chỉ

# Khái niệm phân trang bộ nhớ

- Bộ nhớ vật lý được chia thành từng khối có kích thước bằng nhau: *khung trang/ trang vật lý (frames)*
  - Trang vật lý được đánh số 0, 1, 2, ...: địa chỉ vật lý của trang
  - Trang được dùng làm đơn vị phân phối bộ nhớ
- Chương trình ở bộ nhớ ngoài được chia thành từng trang có kích thước bằng trang vật lý: *trang logic (pages)*
- Khi thực hiện chương trình
  - Nạp các trang logic từ bộ nhớ ngoài vào các trang vật lý (có thể nằm ở vị trí không liên kề nhau)
  - Xây dựng một bảng quản lý trang (PCB: Page Control Block) dùng để xác định mối quan hệ giữa trang logic và trang vật lý

# Ví dụ về phân trang bộ nhớ

0	A.0
1	A.1
2	A.2
3	D.0
4	D.1
5	C.0
6	C.1
7	C.2
8	D.2
9	D.3
10	
11	
12	
13	
14	
15	

Bộ nhớ

0	A.0
1	A.1
2	A.2

Tiến trình A

0	C.0
1	C.1
2	C.2

Tiến trình C

0	D.0
1	D.1
2	D.2
3	D.3

Tiến trình D

Không gian nhớ logic của các tiến trình  
A và C : 3 trang  
D: 4 trang

# Ánh xạ địa chỉ

- Bảng quản lý trang PCB (Pages control block):
  - Dùng để xác định mối quan hệ giữa trang logic trang vật lý
  - Mỗi phần tử của PCB tương ứng với một trang chương trình, chứa *số thứ tự của trang và số thứ tự của khung trang tương ứng*
  - Ví dụ: Bảng trang của các tiến trình trong ví dụ trên

0	0
1	1
2	2

Bảng trang  
tiến trình A

0	5
1	6
2	7

Bảng trang  
tiến trình C

0	3
1	4
2	8
3	9

Bảng trang  
tiến trình D

Trang thứ 2 được  
 nạp vào khung trang  
thứ 8

# Ánh xạ địa chỉ (tiếp)

- *Địa chỉ cơ sở của khung trang =  $f*s$* 
  - $f$ : số thứ tự của khung trang
  - $s$ : kích thước của khung trang
- Ví dụ: Xét bảng trang của các tiến trình D
  - Trang thứ 2 của tiến trình D nằm trong khung trang có vị trí bắt đầu bằng bao nhiêu?
  - Trả lời:  $8*s$

0	3
1	4
2	8
3	9

Bảng trang  
tiến trình D

# Ánh xạ địa chỉ (tiếp)

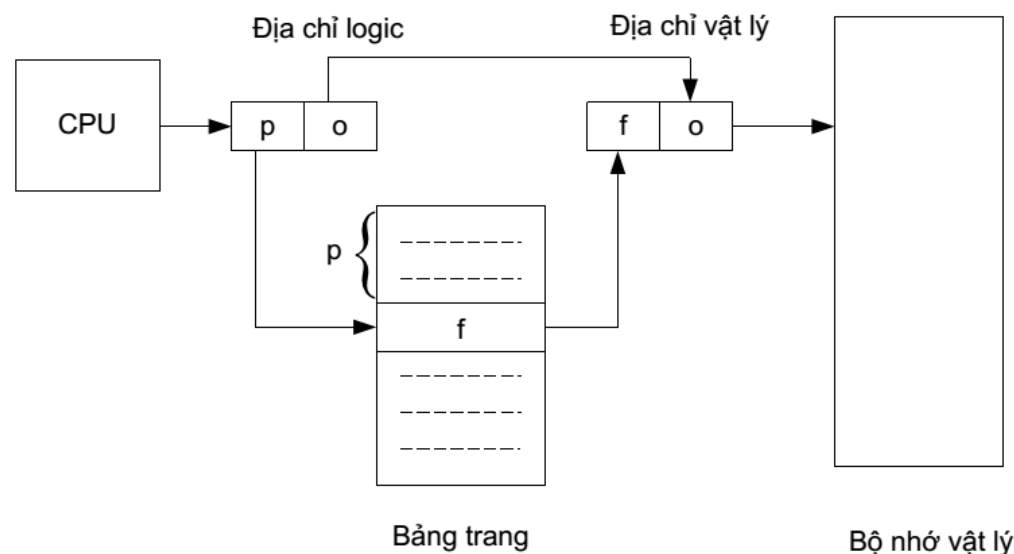
- Địa chỉ logic của ngăn nhớ có hai thành phần:

*<số thứ tự trang (p), độ dịch trong trang (o)>*

*m* bit

*n* bit

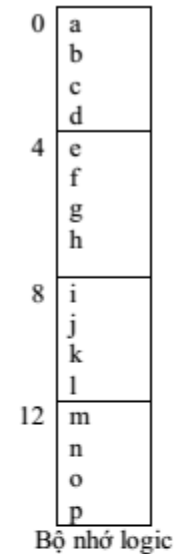
- Kích thước trang nhớ:  $s = 2^n$  ngăn nhớ
- Tra bảng trang tìm được số thứ tự khung trang: *f*
- Địa chỉ cơ sở của khung trang:  $f \cdot s$
- Địa chỉ vật lý của ô nhớ:  $f \cdot s + o$





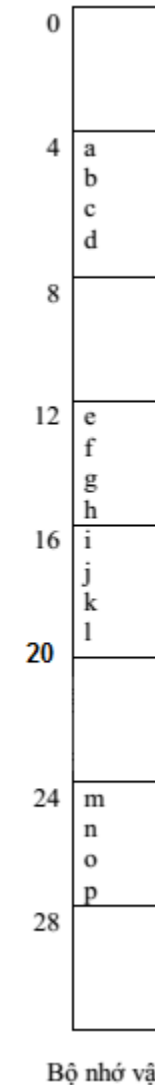
# Ví dụ ánh xạ địa chỉ trong phân trang bộ nhớ

- a) Xác định địa chỉ vật lý của ngăn nhớ có địa chỉ logic 0001?
- b) Xác định địa chỉ vật lý của ngăn nhớ có địa chỉ logic 0110?



0	1
1	3
2	4
3	6

Bảng trang



# Các phương pháp nạp trang và đổi trang

- Chiến lược phân trang thường xuyên phải thực hiện việc nạp trang, đưa trang ra bộ nhớ ngoài...
- ➔ Làm tăng hao phí thời gian và chậm tốc độ hệ thống
- ➔ Cần có các biện pháp nạp trang và đổi trang sao cho thích hợp nhất, tiết kiệm thời gian nhất!

# Các chiến lược nạp trang

- Nạp theo yêu cầu
- Nạp trước

# Các chiến lược đổi trang

- Đổi trang ngẫu nhiên
- Nạp trước thì thay trước (FIFO)
- Đổi trang có lần sử dụng cuối cùng cách đây lâu nhất (LRU-Least Recently Used)
- ...

## 4.5 - Phân đoạn bộ nhớ

- Phân đoạn của chương trình
- Quản lý và cấp phát phân đoạn bộ nhớ
- Ánh xạ địa chỉ và chống truy cập trái phép

# Phân đoạn của chương trình (tiếp)

- Chương trình được chia thành những phần khác nhau gọi là đoạn (segment): Đoạn chương trình, Đoạn dữ liệu, Đoạn ngăn xếp,
  - Mỗi đoạn tương ứng với không gian địa chỉ riêng, được phân biệt bởi tên (số thứ tự) và độ dài của nó
    - ◆ Mỗi phần tử của đoạn được xác định bởi vị trí tương đối của nó so với đầu đoạn

# Quản lý và cấp phát phân đoạn bộ nhớ

- Khi nạp chương trình vào bộ nhớ thực hiện:
  - Mỗi đoạn sẽ được cấp một vùng nhớ liên tục có kích thước bằng kích thước của đoạn cần nạp
    - ♦ Chiến lược cấp phát bộ nhớ cho đoạn: first fit, best fit
  - Mỗi chương trình được cấp những đoạn bộ nhớ không nằm liền kề nhau
  - Để quản lý các đoạn nhớ, người ta sử dụng bảng quản lý đoạn SCB (Segment control block)

# Bảng SCB

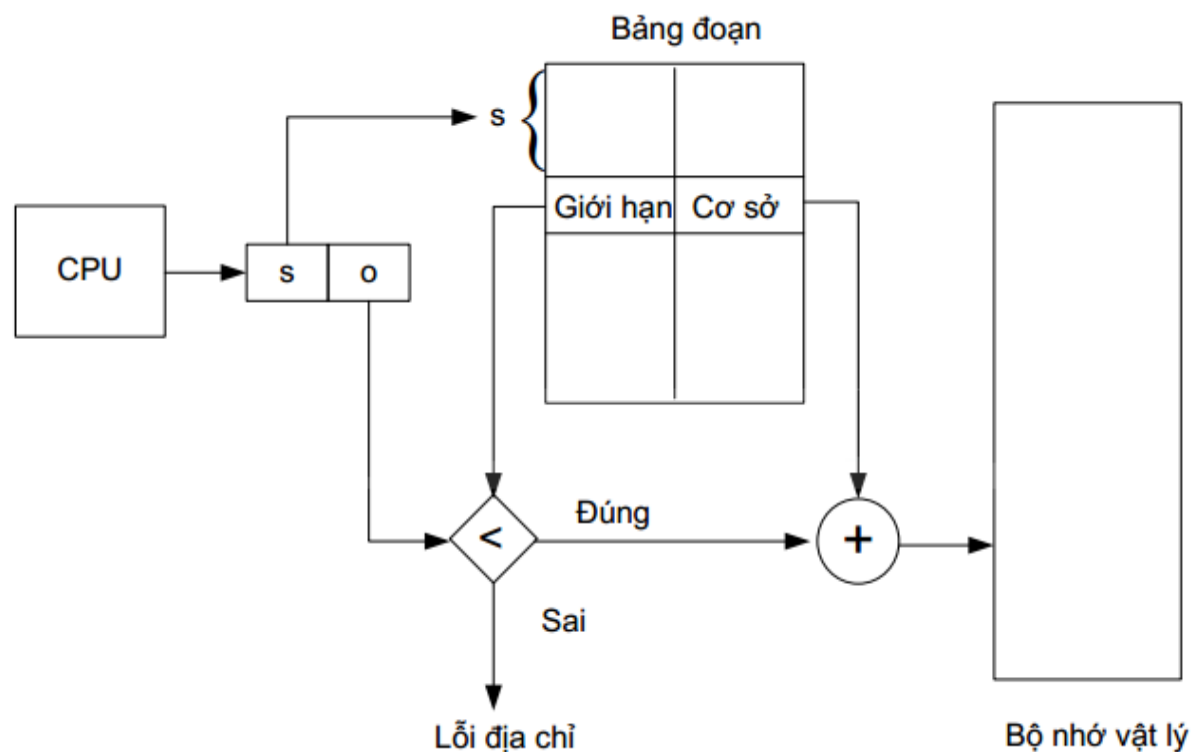
- Mỗi ô của bảng ứng với một đoạn của chương trình:
  - Dấu hiệu (Mark = 0/1): Đoạn đã tồn tại trong bộ nhớ
  - Địa chỉ (Address): Vị trí cơ sở của đoạn
  - Độ dài (Length): Độ dài đoạn

	Mark	Address	Length
0			
.			
.			
.			
n	...	...	...



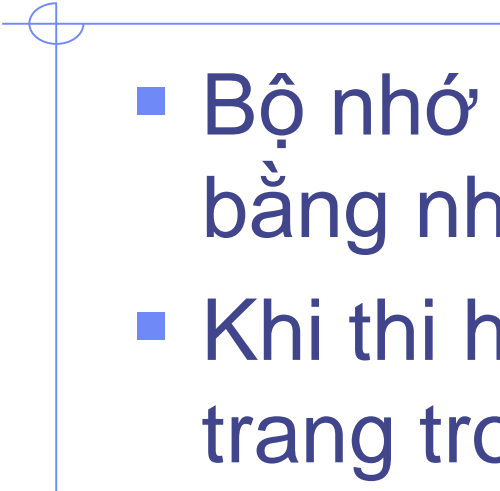
# Ánh xạ địa chỉ và chống truy cập trái phép

- Trong mỗi đoạn, địa chỉ logic của ngăn nhớ có hai thành phần:  
*<số thứ tự đoạn (s), độ dịch trong đoạn (o)>*



# Kết hợp phân trang - đoạn

- Chương trình được biên tập theo kiểu phân đoạn, các đoạn được quản lý bằng bảng SCB
- Mỗi đoạn (mỗi modul chương trình) lại được chia thành từng trang → Mỗi đoạn sẽ có bảng PCB của riêng nó

- 
- Bộ nhớ vật lý được chia thành các trang có kích thước bằng nhau.
  - Khi thi hành, các trang của đoạn sẽ được nạp vào các trang trong bộ nhớ vật lý.
  - Trường *Address* trong bảng SCB sẽ là nơi lưu giữ địa chỉ bảng PCB của từng đoạn.

## 4.6 – Quản lý bộ nhớ trên IBM - PC

- Quản lý bộ nhớ trên hệ thống 8086
- Quản lý bộ nhớ trên hệ thống 80286
- Quản lý bộ nhớ trên hệ thống 80386

# Quản lý bộ nhớ trên hệ thống 8086

- Địa chỉ vật lý dài 20 bit
- Chỉ quản lý được 1MB bộ nhớ
- Bộ nhớ được chia thành các đoạn có kích thước tối đa 64 KB (xếp chồng lên nhau)
- Địa chỉ logic có dạng segment:offset, trong đó segment dài 16 bit, offset dài 16 bit

# Quản lý bộ nhớ trên hệ thống 80286

## ***Có hai chế độ:***

- Chế độ thực (Real mode)
- Chế độ bảo vệ (Protected mode)

# Chế độ thực (Real mode)

- Áp dụng phương pháp quản lý bộ nhớ tương tự hệ thống 8086
- Địa chỉ vật lý dài 24 bit
- Về lý thuyết, nó có thể quản lý được 16 MB bộ nhớ vật lý, nhưng do phải tương thích với 8086 (vẫn sử dụng segment dài 16 bit, offset dài 16 bit), nên trên thực tế nó chỉ có bộ nhớ  $\sim 1\text{MB} + 64\text{KB}$

# Chế độ bảo vệ (Protected mode)

- Áp dụng phương pháp quản lý bộ nhớ kiểu phân đoạn
- Có thể quản lý tối đa 16MB bộ nhớ vật lý (nhờ sử dụng địa chỉ vật lý 24 bit)
- Cho phép chạy các chương trình có kích thước lớn hơn bộ nhớ vật lý nhờ sử dụng các địa chỉ ảo (Virtual Address)
- Có cơ chế “bảo vệ các đoạn nhớ”, tránh xung đột khi sử dụng bộ nhớ



# Địa chỉ logic (địa chỉ ảo)

- Địa chỉ logic dạng *segment:offset*  
(*segment* dài 16 bit, *offset* dài 16 bit)
- Địa chỉ này không xác định một ô nhớ vật lý cụ thể, nó ứng với một ô nhớ logic (có thể nằm trên bộ nhớ vật lý hoặc không)
- + *Segment*: chứa các thông tin giúp tìm tới địa chỉ đoạn nhớ
- + *Offset*: Xác định vị trí của ô nhớ trong đoạn

# Bảng quản lý đoạn

- Các đoạn nhớ được quản lý bởi các bảng mô tả (Descriptor table), mỗi bản ghi của bảng sẽ chứa thông tin về một đoạn nhớ

# Nội dung bảng:

0...15	16...39	40	41...43	44	45	46	47	48	...	63
Limit (16 bít)	Base (24 bít)	A	Type	S	DPL	P	Reserved (16 bít)			
...	...	...	...	...	...	...	...			
...	...	...	...	...	...	...	...			
...	...	...	...	...	...	...	...			

# Giải thích:

- Trường **Limit**: chứa độ dài của đoạn nhớ (tối đa là  $2^{16} = 64$  KB)
- Trường **Base**: Địa chỉ của đoạn nhớ trong bộ nhớ (địa chỉ 24 bit)
- Trường **P**:
  - +  $P = 0$ : Đoạn chưa được nạp vào bộ nhớ
  - +  $P = 1$ : Đoạn đã được nạp vào bộ nhớ
- Trường **DPL**: Chứa mức ưu tiên của đoạn (Descriptor Privilege Level)

# Segment

- Segment còn được gọi là *Bộ chọn đoạn*, nó chứa các thông tin sau:



- *Từ bit 3 -> 15: chứa số hiệu đoạn logic, tức là số hiệu bản ghi trong bảng mô tả*

- Trường *RPL* (dài 2 bít): Chứa mức ưu tiên mong muốn (*Request Privilege Level*)
- Trường *TI*:
  - + *TI = 0*: Đoạn do bảng mô tả toàn cục quản lý  
(*GDT - Global Descriptor Table*)
  - + *TI = 1*: Đoạn do bảng mô tả cục bộ quản lý (*LDT – Local Descriptor Table*)



***Các bước để truy cập vào ô nhớ có địa chỉ là segment:offset?***

# Các mức ưu tiên

- *Mức 0:* Là mức ưu tiên cao nhất, dành cho các chương trình cấp thấp như quản lý CPU, quản lý bộ nhớ...
- *Mức 1:* Dành cho các chương trình quản lý vào/ra
- *Mức 2:* Dành cho các chương trình quản lý file
- *Mức 3:* Là mức ưu tiên thấp nhất, dành cho các chương trình ứng dụng



# Quản lý bộ nhớ trên hệ thống 80386

- Cũng có hai chế độ (chế độ thực và bảo vệ) giống 80286
- Trong chế độ bảo vệ: có thể áp dụng phương pháp quản lý bộ nhớ kiểu phân đoạn hoặc kết hợp phân trang - đoạn

# Quản lý bộ nhớ kiểu phân đoạn

*Cũng tương tự như trên 80286, có thêm một số thay đổi sau:*

- Sử dụng địa chỉ vật lý dài 32 bit nên có thể quản lý tối đa 4 GB bộ nhớ vật lý
- Địa chỉ *segment:offset* dài 16 bit: 32 bit

# Bảng mô tả:

0	...	15	16	...	39	40	...	47	48	...	51	52...	55	56	...	63
Limit (16 bít)		Base (24 bít)		...		Limit (4 bít)		x x DG (4 bít)		Base (8 bít)						
...		...		...		...		...		...		...		...		...
...		...		...		...		...		...		...		...		...
...		...		...		...		...		...		...		...		...

# Giải thích:

- Trường **Limit**: dài  $16 + 4 = 20$  bit (độ dài đoạn tối đa là  $2^{20}$  [đơn vị])
  - + Nếu bit **G = 0**: Đơn vị đo là Byte  
(Một đoạn dài tối đa  $2^{20}$  byte = 1 MB)
  - + Nếu bit **G = 1**: Đơn vị đo là Trang
- Trường **Base**: dài  $24 + 8 = 32$  bit (địa chỉ vật lý 32 bit)

# Bộ nhớ kết hợp phân trang - đoạn

- Chương trình được chia thành nhiều đoạn, mỗi đoạn ứng với một modul
- Các modul được “**nạp**” vào bộ nhớ tuyến tính (bộ nhớ ảo).
- Bộ nhớ tuyến tính được chia thành các trang có kích thước **4 KB**.
- Bộ nhớ vật lý được chia thành các trang **4 KB**, các trang của bộ nhớ tuyến tính sẽ được nạp vào các trang của bộ nhớ vật lý

# Quản lý các trang

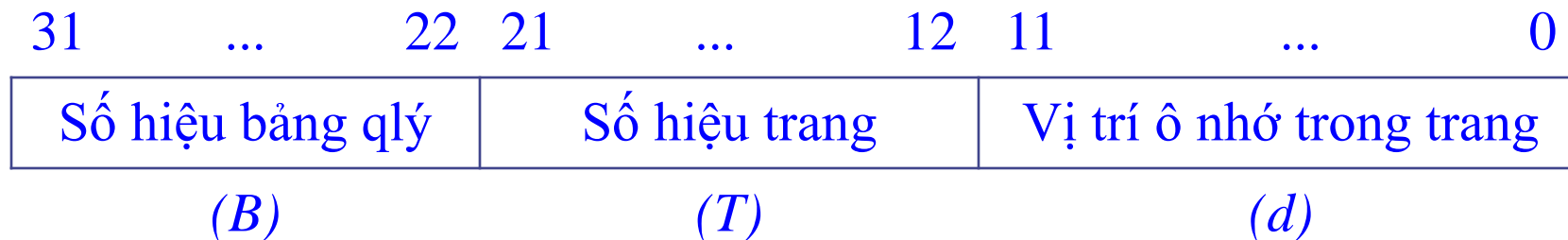
- Bộ nhớ tuyến tính có tới  $2^{20}$  trang nên kích thước bảng quản lý trang sẽ rất lớn
- Cần thực hiện phân cấp để giảm kích thước bảng quản lý trang

- Một *bảng quản lý* có thể chứa thông tin của  $2^{10} = 1024$  trang
- Một *thư mục trang* có thể chứa thông tin của  $2^{10} = 1024$  *bảng quản lý trang*

➔ Một *thư mục trang* sẽ quản lý được  $2^{10} \times 2^{10} = 2^{20}$  trang

# Địa chỉ tuyến tính

- Địa chỉ tuyến tính dài 32 bít, được chia làm 3 trường:







*Hết Chương 4*