Chương IV

QUẢN LÝ BỘ NHỚ CHÍNH

(Main Memory)

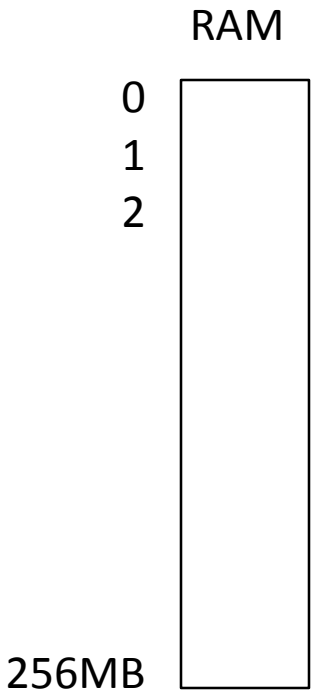
Nội Dung Chương IV

I. Mục đích bộ nhớ chính
II. Vấn đề quản lý bộ nhớ trong hệ đa tiến trình
III. Cấp phát bộ nhớ theo mô hình phân trang
IV. Cấp phát bộ nhớ theo mô hình phân đoạn
V. Kết hợp phân đoạn và phân trang

I. Mục đích bộ nhớ vật lý(RAM)

1) Đặc điểm của bộ nhớ vật lý
   - Là một dãy byte liên tiếp
   - Mỗi byte có thể được CPU truy cập trực tiếp theo cơ chế đánh địa chỉ.

2) Mục đích bộ nhớ vật lý
   - Là nơi lưu giữ các mã lệnh của chương trình trước nạp vào CPU thi hành.
   - Khi các lệnh thực hiện, CPU đọc dữ liệu từ bộ nhớ, tính toán và ghi lại kết quả ra bộ nhớ.

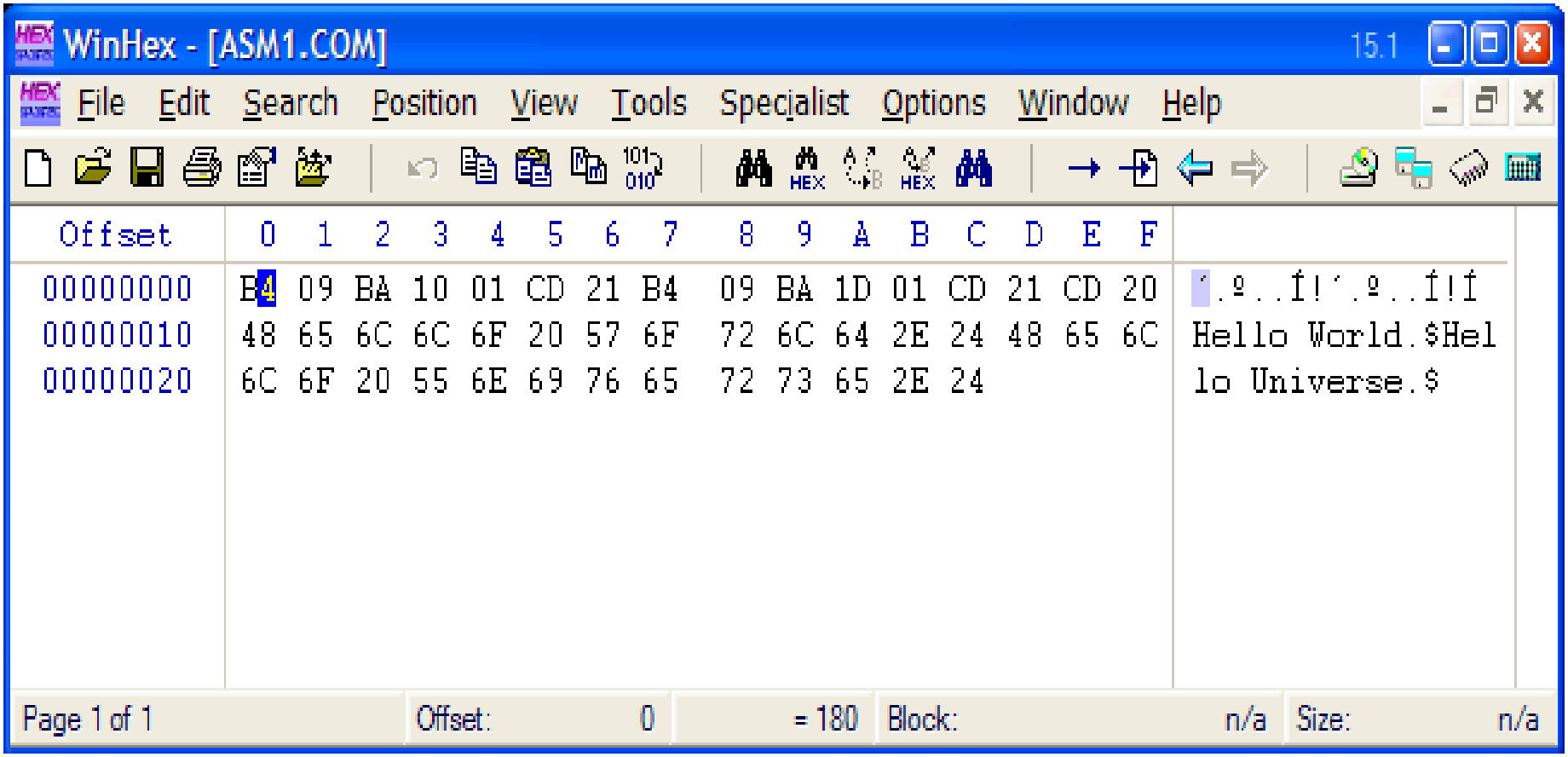


3) Cấu trúc vùng nhớ logic của một chương trình:

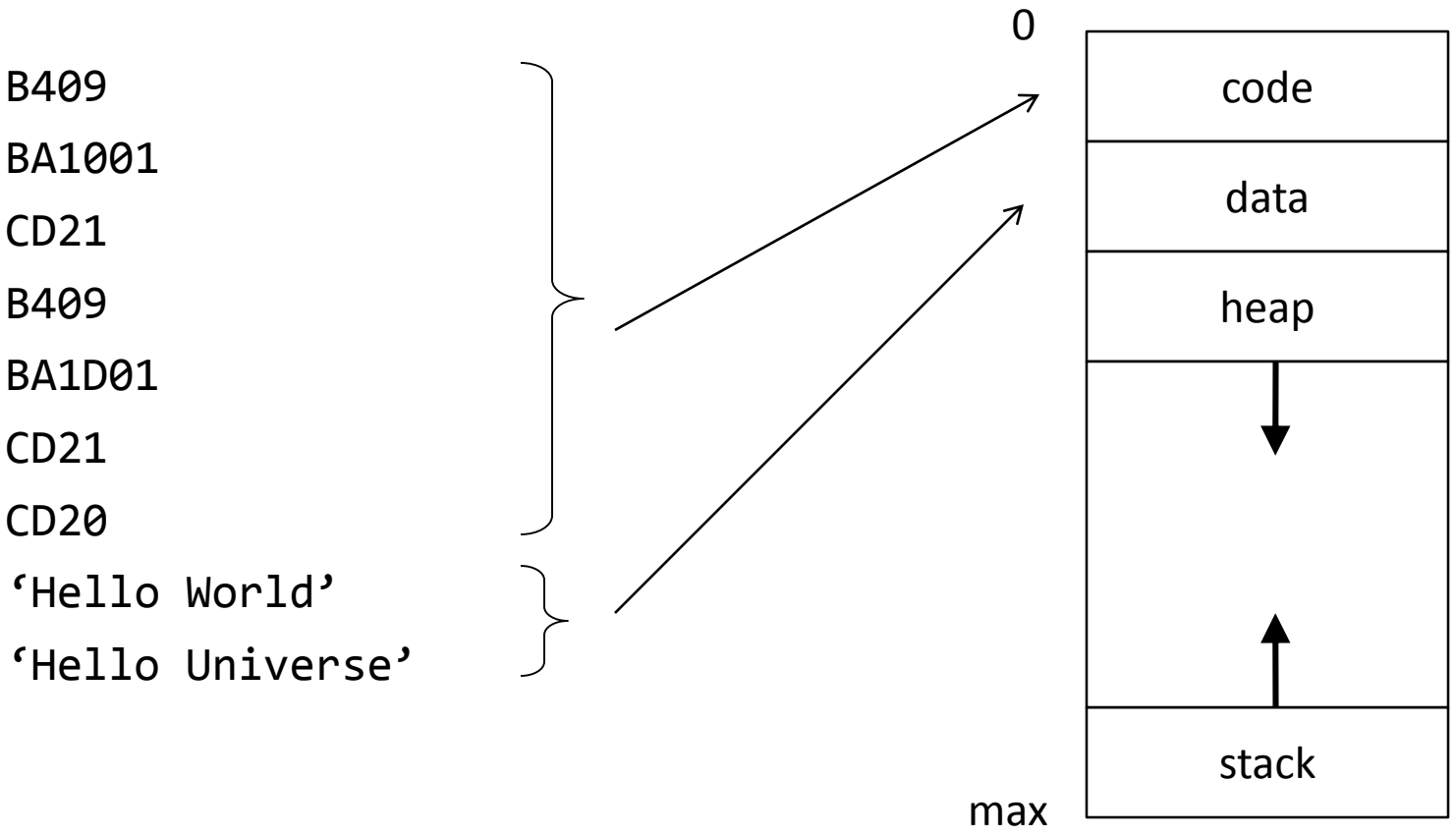
là một vùng nhớ *liên tục*.

```
.MODEL TINY
.CODE
ORG 100h
Begin:

   MOV ah, 9
   MOV dx, OFFSET Mess1
   INT 21h

   MOV ah, 9
   MOV dx, OFFSET Mess2
   INT 21h

   INT 20h

   Mess1 DB 'Hello World.$'
   Mess2 DB 'Hello Universe.$‘

END Begin
```

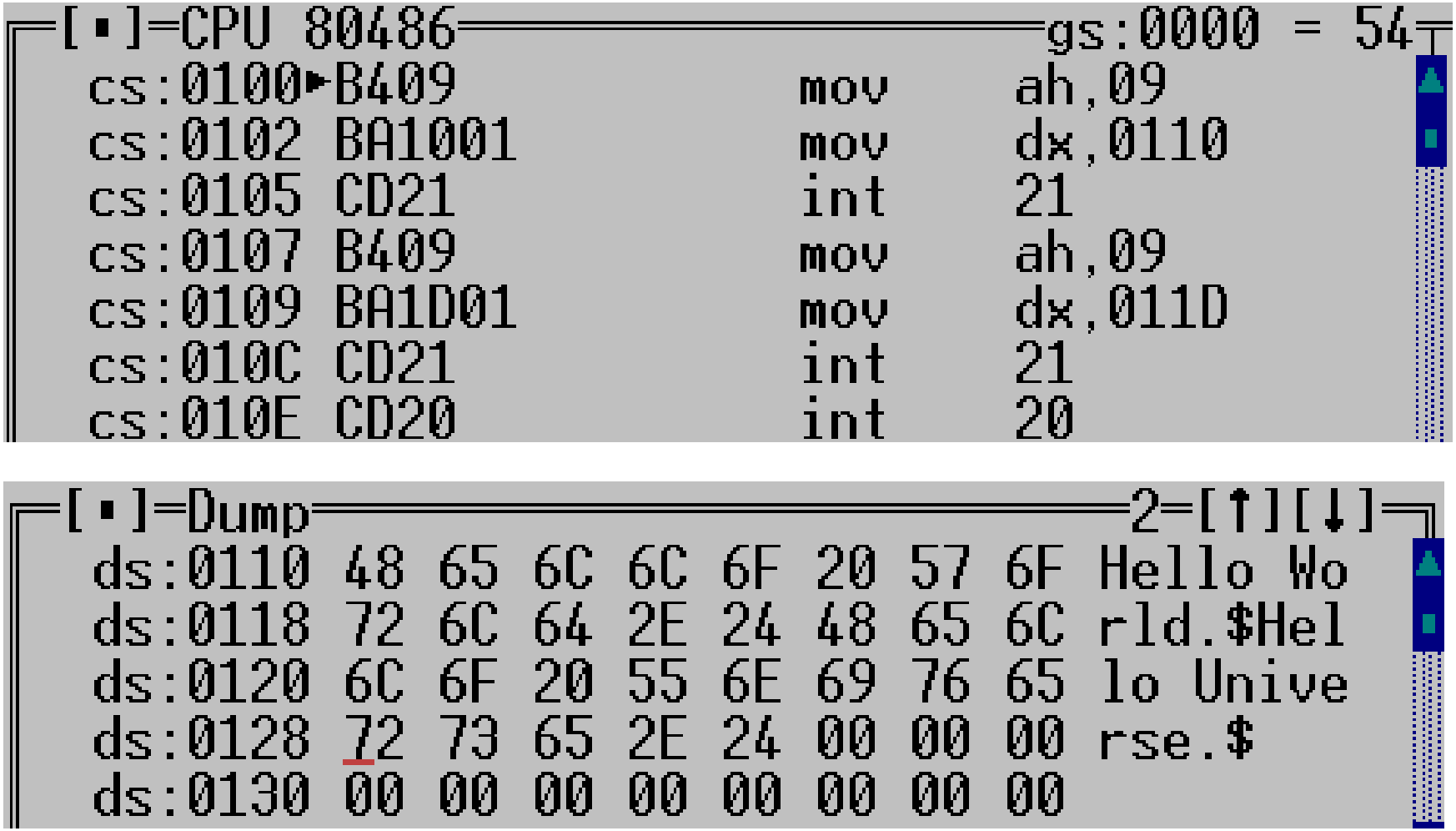
- Sau khi biên dịch thành chương trình COM trên đĩa (Hello.com):



Vùng nhớ logic được chia thành các đoạn: code, data..



* Ví dụ về vùng lệnh (code) và vùng dữ liệu (data) khi nạp vào bộ nhớ vật lý:

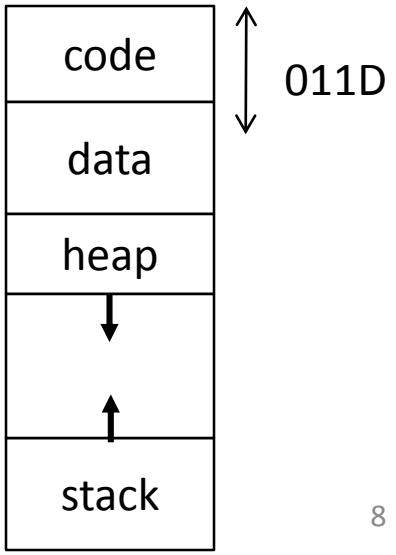


3) Từ tên biến đến địa chỉ logic

- Khi lập trình, dùng tên biến
- Sau khi biên dịch thành chương trình, mỗi biến có một địa chỉ trong vùng nhớ chương trình (địa chỉ logic).
- Địa chỉ logic là một số nguyên n bít, được tính từ vị trí đầu vùng nhớ đến vị trí biến.

Ví dụ:

- Biến Mess1 là biến đầu tiên, có địa chỉ logic 0110
- Biến Mess2 sắp sau biến Mess1, có địa chỉ logic 011D (do chiều dài Mess1 là 13 ký tự)



4) Từ địa chỉ logic đến địa chỉ vật lý

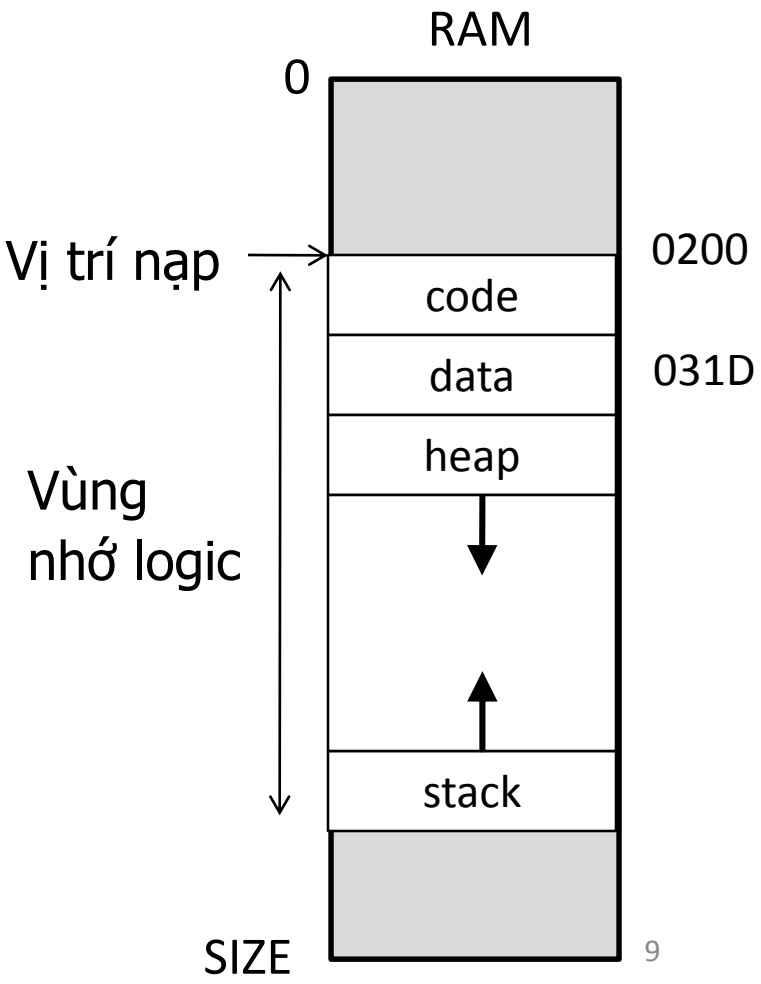
- Khi chương trình thực hiện thì vùng nhớ của chương trình được nạp vào một vị trí bất kỳ trong bộ nhớ vật lý

→ Địa chỉ logic được chuyển thành địa chỉ vật lý.

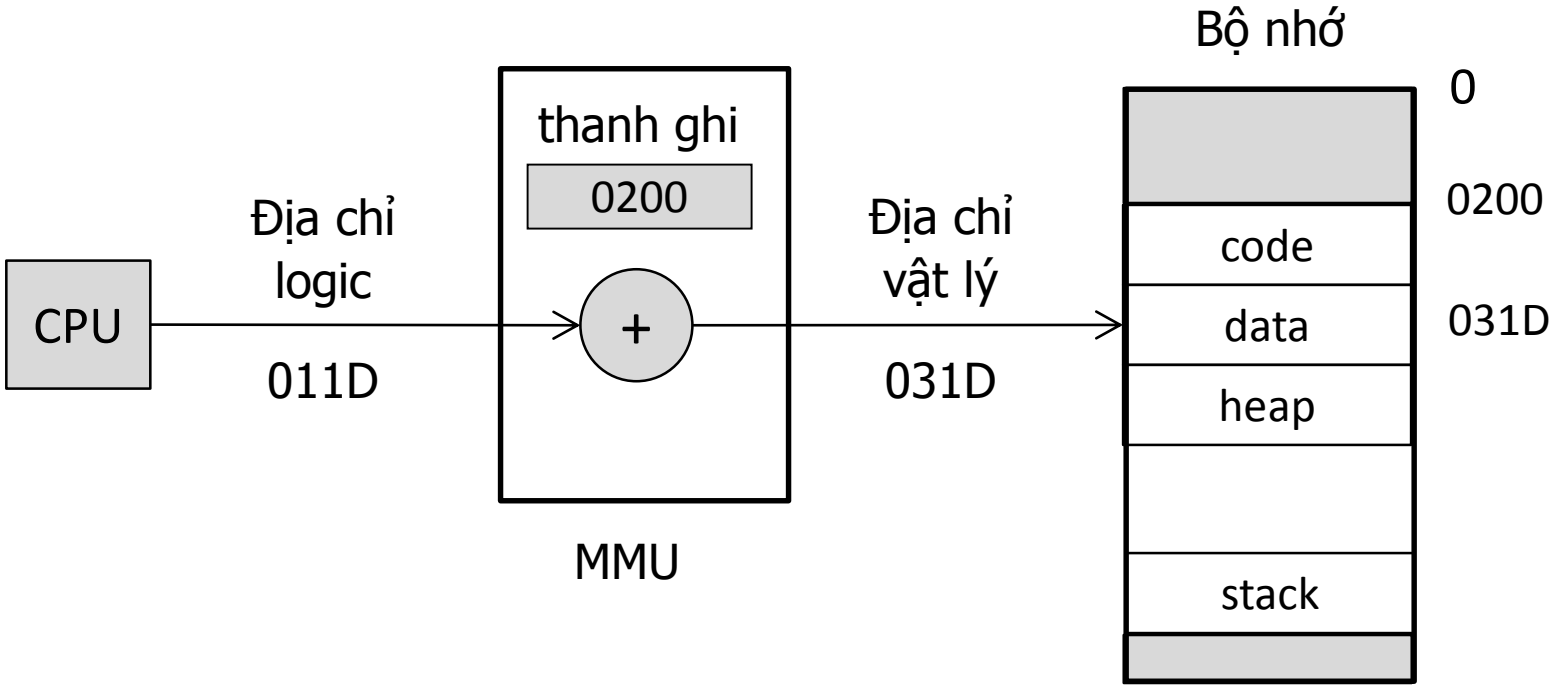
Ví dụ: vị trí nạp là 0200 thì biến Mess2 (địa chỉ logic 011D) có địa chỉ vật lý là 031D

→ CPU truy xuất 1 biến dùng địa chỉ vật lý.

```
MOV ah, [ds:011D]
⇔ ah ← [031D]
```



Cách thức thực hiện: dùng thanh ghi tái định vị để chuyển từ logical address sang physical address.
Ví dụ: 8x86 của Intel có thanh ghi định vị CS, DS.



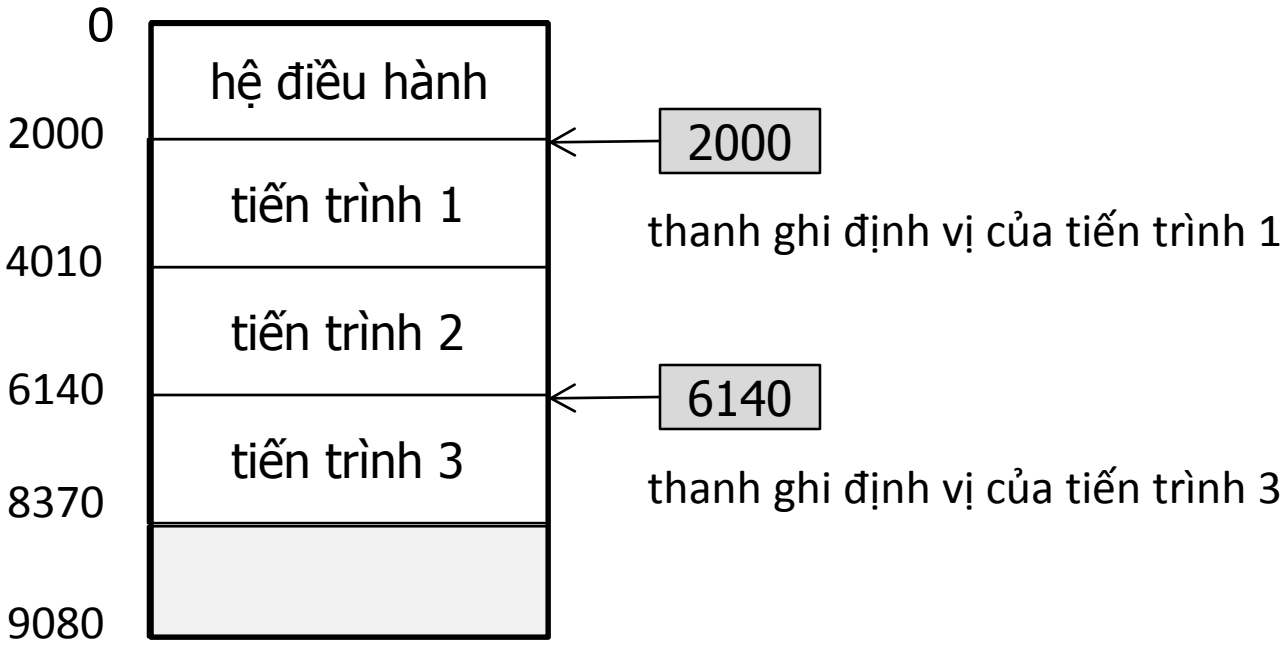
MMU (bộ quản lý bộ nhớ): là thiết bị phần cứng dùng để chuyển đổi từ logical addess sang physical address

II. Quản lý bộ nhớ trong hệ đa tiến trình

Tổ chức bộ nhớ như thế nào trong hệ đa tiến trình?

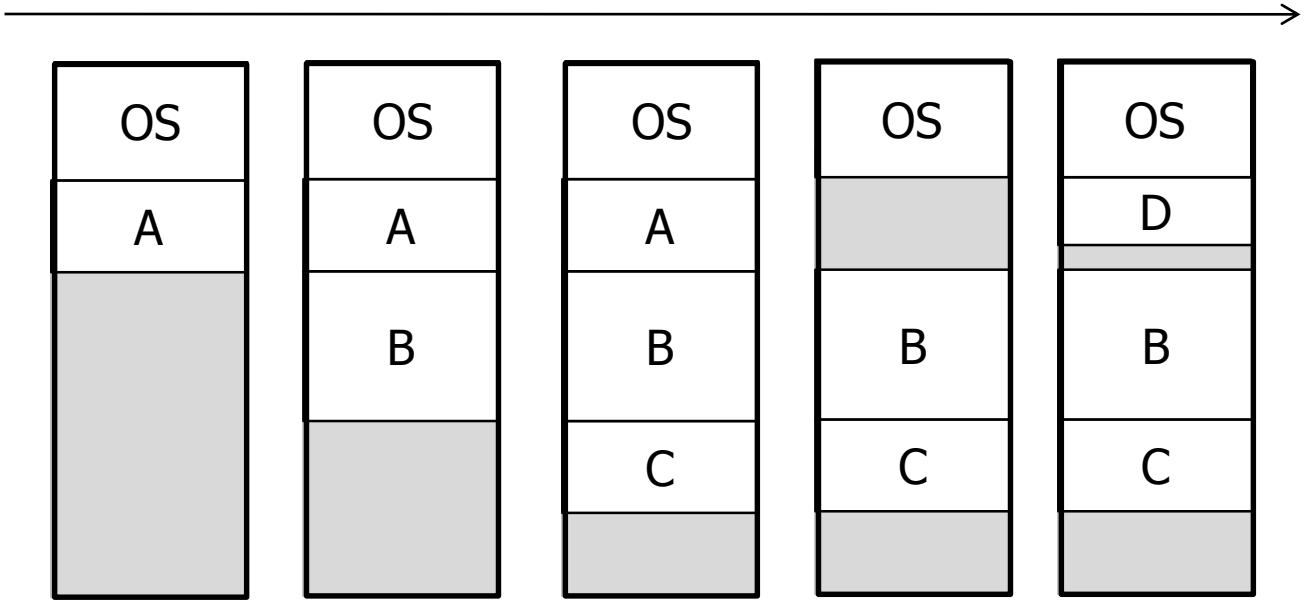
1. Giải pháp thực hiện

   Vùng nhớ logic của các tiến trình đồng thời nạp vào bộ nhớ. Mỗi tiến trình có thanh ghi định vị riêng.



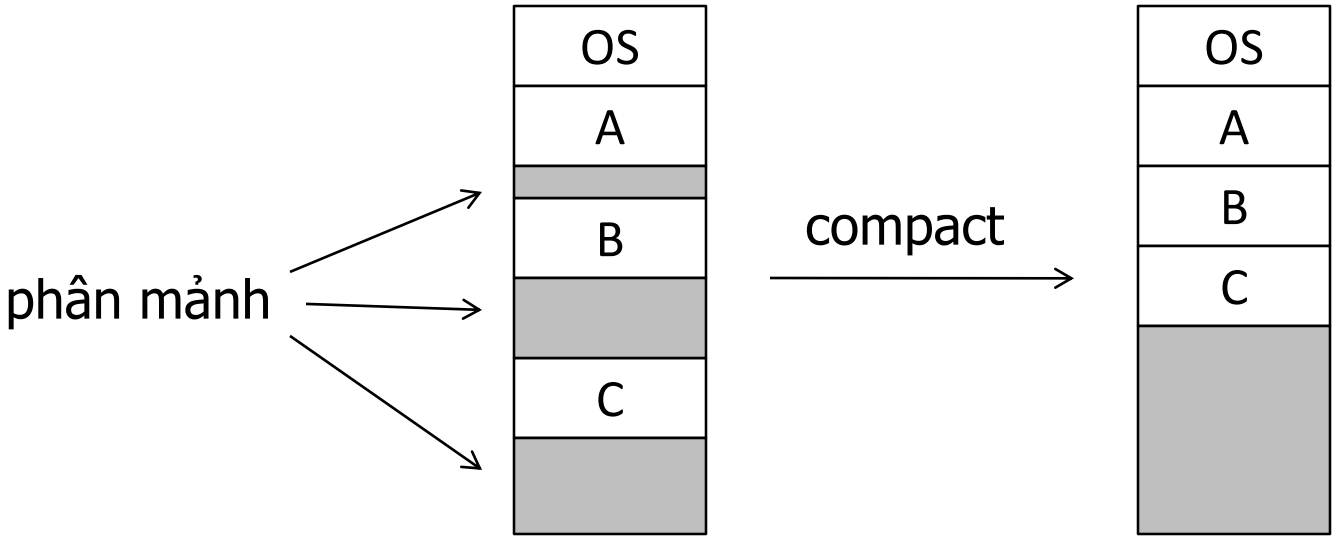
3) Đặc điểm của giải pháp

- Bộ nhớ gồm 2 loại vùng nhớ: vùng nhớ đã cấp cho các tiến trình và vùng nhớ trống.
- Khi một tiến trình vào hệ thống, HĐH xác định một vùng nhớ trống phù hợp để cấp cho tiến trình.
- Khi một tiến trình kết thúc, vùng nhớ của tiến trình được giải phóng trở thành vùng nhớ trống.



4) Sự phân mảnh vùng nhớ trống

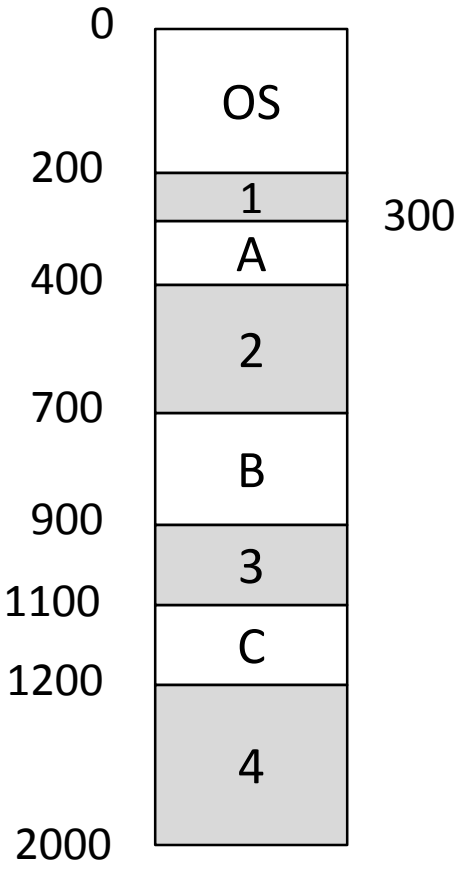
- Sau một thời gian cấp phát và thu hồi, vùng nhớ trống không còn liên tục mà bị chia thành nhiều vùng khác nhau (phân mảnh ngoại vi - external fragmentation)
- Có những mảnh kích thước nhỏ không thể sử dụng → hao phí bộ nhớ. Giải pháp: tiến hành sắp xếp lại các vùng nhớ (compact).



5) Các chiến lược cấp phát vùng nhớ trống

- Việc compact chỉ tiến hành khi có nhiều mảnh nhỏ. Nếu không vẫn có thể cấp phát vùng nhớ cho tiến trình mới trên các vùng trống.
- Các chiến lược cấp phát:
  - First Fit: tìm vùng nhớ đầu tiên đủ lớn để chứa tiến trình
  - Best Fit: tìm vùng nhớ nhỏ nhất mà có thể chứa tiến trình
  - Worst Fit: tìm vùng nhớ lớn nhất cấp cho tiến trình.

Ví dụ: tiến trình D mới vào có kích thước 150. Cần cấp cho D vùng nhớ nào?

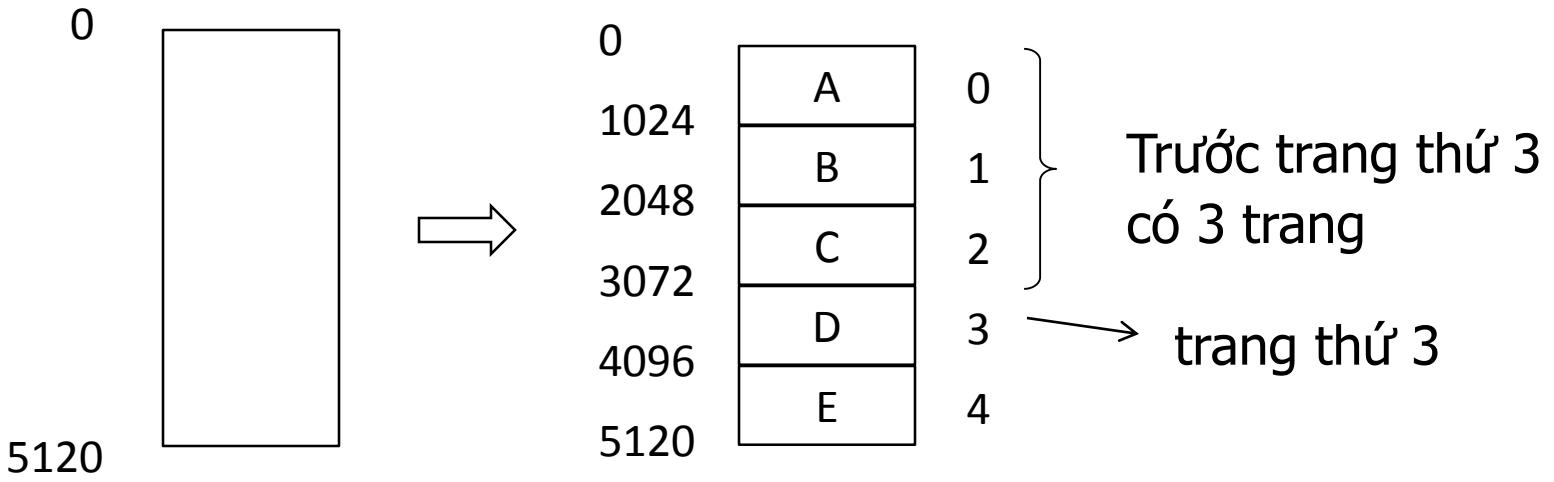


- First Fit: cấp vùng 2
- Best Fit: cấp vùng 3
- Worst Fit: cấp vùng 4

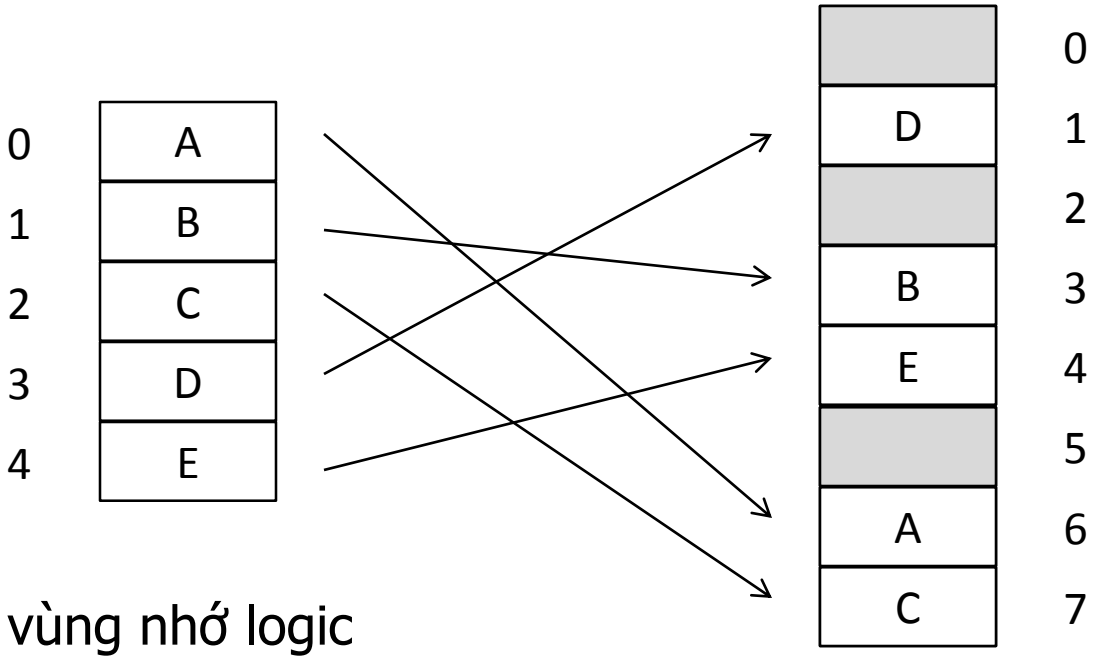
III. Cấp phát bộ nhớ phân trang (paging)

1) Ý tưởng:

- Vùng nhớ logic tiến trình được chia thành các khối có kích thước cố định (trang - page).
- Ví dụ: kích thước vùng nhớ logic: 5120, kích thước trang: 1024

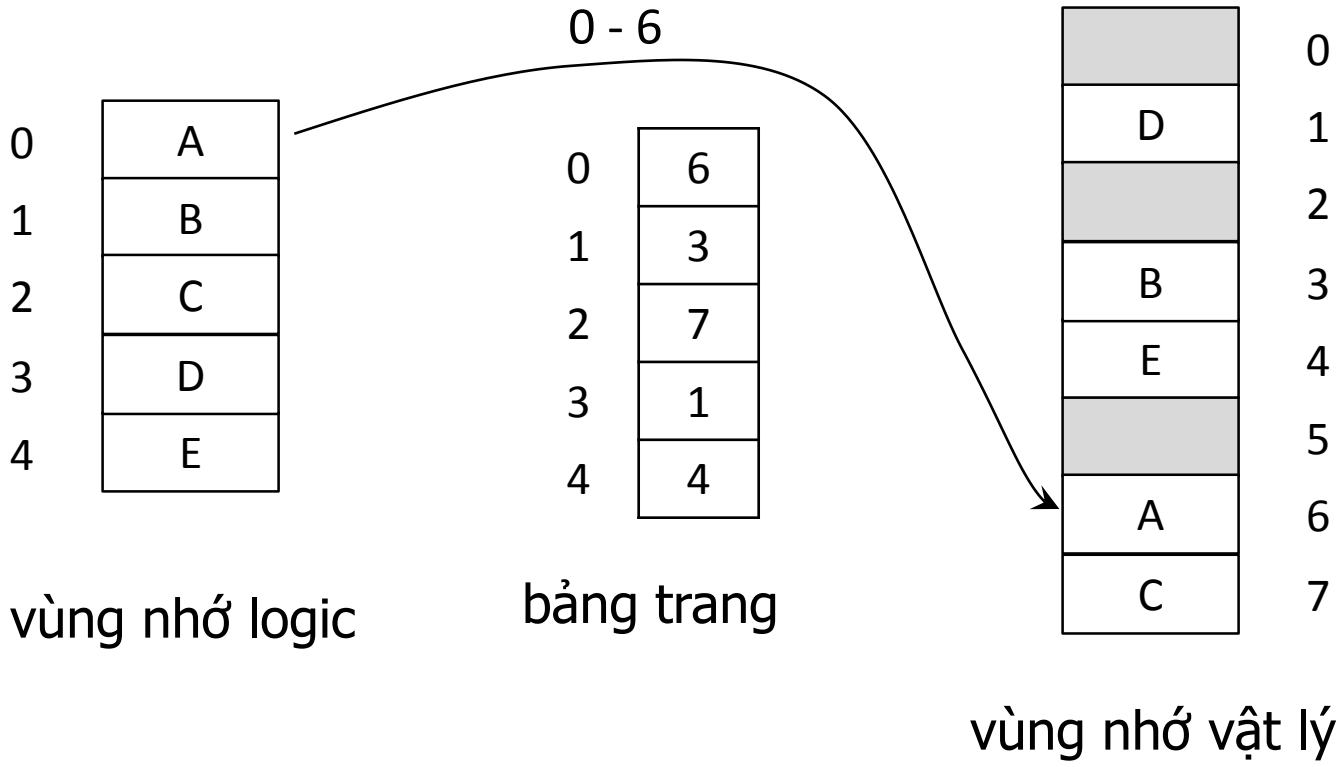


- Bộ nhớ vật lý cũng được chia thành các khung trang (frame) có kích thước như các trang.
- Các trang logic của tiến trình khi nạp vào bộ nhớ vật lý *không nhất thiết phải liên tục* mà có thể phân bố tại các khung trang khác nhau.



2) Cơ chế thực hiện: dùng bảng trang (page table):

Bảng trang dùng để xác định vị trí của trang logic trong bộ nhớ vật lý. Bảng trang do HĐH quản lý.



2) Chuyển đổi địa chỉ logic sang địa chỉ vật lý

a) Chọn kích thước trang

Kích thước trang (PS) thường được chọn là 4K hay 8K, có dạng 2^m ($4028 = 2^{10}$)

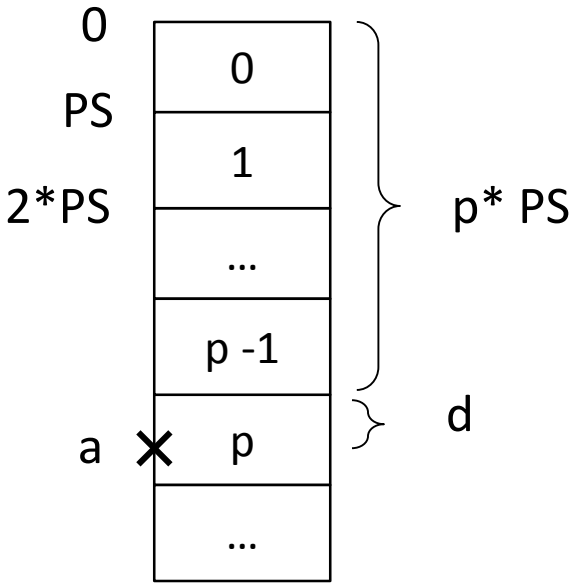
b) Chuyển địa chỉ logic từ dạng liên tục sang dạng trang

Địa chỉ logic liên tục: a

Địa chỉ trang gồm 2 phần: <p, d>

- p: số thứ tự trang
- d: vị trí của byte trong trang

a $\Leftrightarrow$ <p, d>



Mối liên hệ: $a = p*PS + d$

→ $p = a \text{ div } PS$

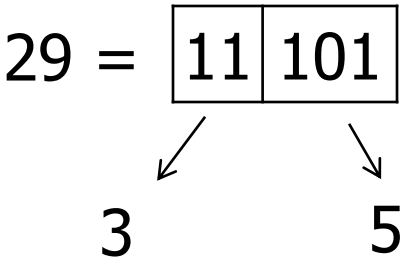
$d = a \text{ mod } PS$

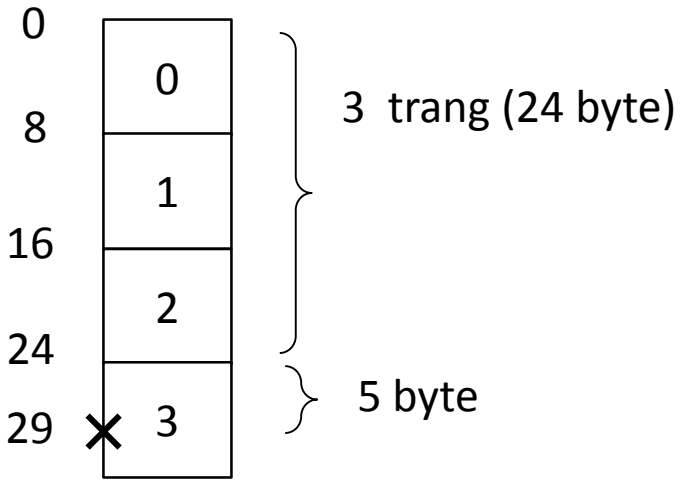
Nếu PS có dạng 2^m thì d là m bít thấp của a, p là n-m bít cao của a.

Ví dụ: PS = 8 (2^3), a = 29

p = 29 div 8 = 3

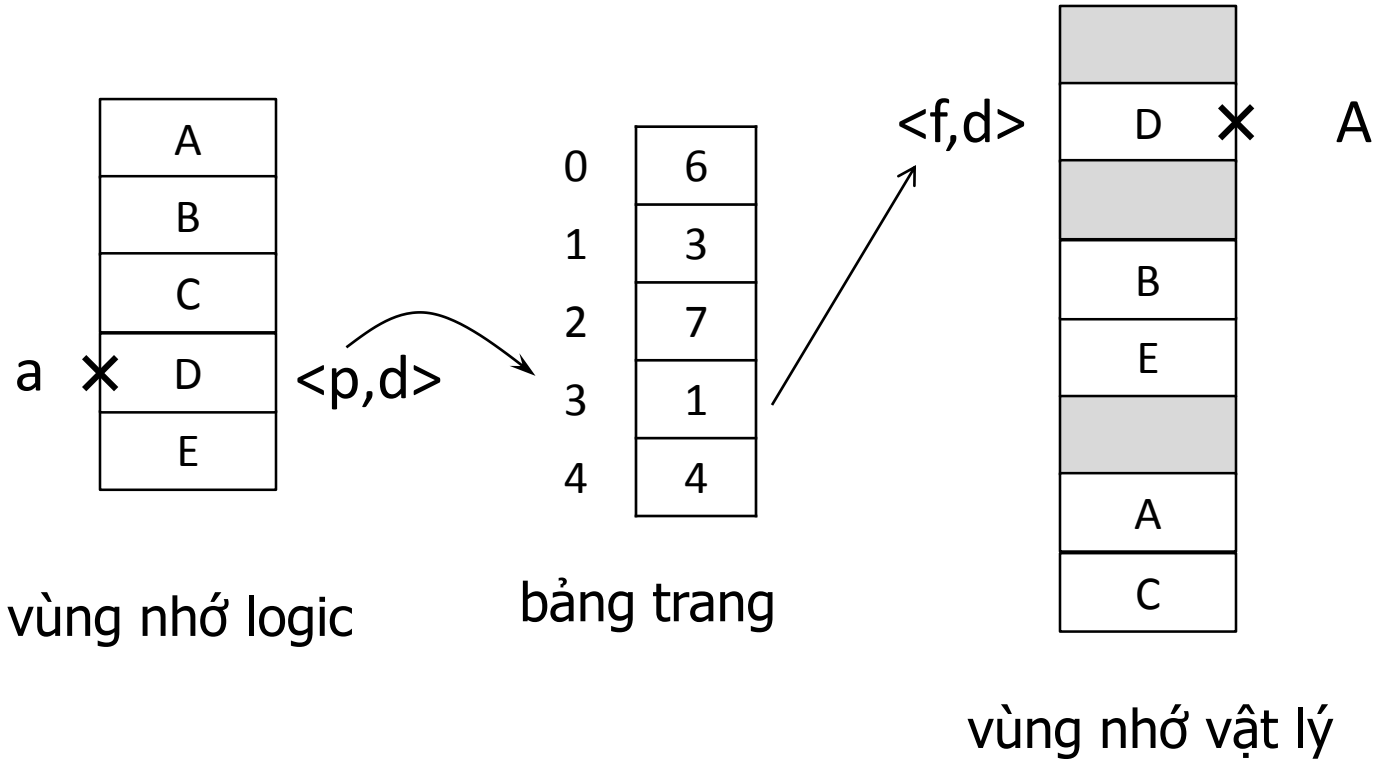
d = 29 mod 8 = 5





d) Cách chuyển đổi:

Có địa chỉ logic a, muốn tính địa chỉ vật lý A ?



- a $\rightarrow$ <p, d> (đổi địa chỉ liên tục thành địa chỉ trang)
- p $\rightarrow$ f (dùng bảng trang, tính khung trang vật lý)
- <f, d> $\rightarrow$ A (đổi địa trang vật lý thành địa chỉ liên tục)

Công thức:

$$p = a \text{ div } PS$$

$$d = a \bmod PS$$

$$f = PageTable(p)$$

$$A = f*PS + d$$

e) Ví dụ minh họa

Giả sử không gian logic gồm 16 byte, kích thước trang là 4 byte. Không gian vậy lý gồm 28 byte.

0	a
1	b
2	c
3	d
4	e
5	f
6	g
7	h
8	i
9	j
10	k
11	l
12	m
13	n
14	o
15	p

0	5
1	6
2	1
3	2

0	
4	i j k l
8	m n o p
12	
16	
20	a b c d
24	e f g h

Tìm địa chỉ vật lý của địa chỉ logic a = 13?

- p = 13 div 4 =3
- d = 13 mod 4 = 1
- p = 3 → f = 2
- A = 2 * 4 + 1 = 9

Tìm địa chỉ vật lý của địa chỉ logic a = 10?

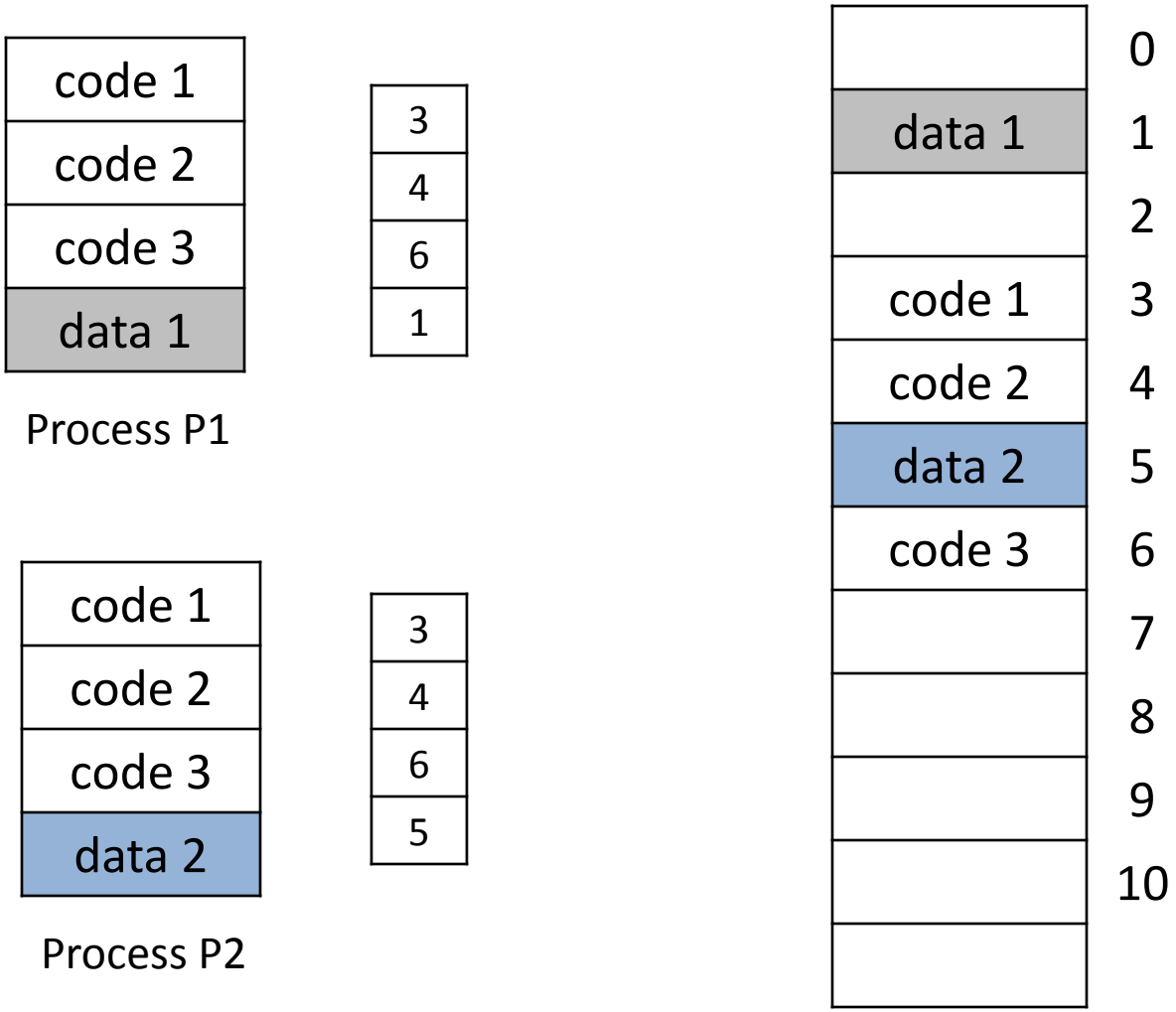
3) Đặc điểm của phương pháp phân trang

- Mỗi tiến trình có một page table riêng → tăng chi phí quản lý PCB cũng như thời gian context switch
- Tránh được tình trạng phân mảnh ngoại và sắp xếp lại bộ nhớ.
- Thể hiện sự tách biệt giữa không gian logic và không gian vật lý, một tiến trình không thể truy xuất đến các trang của tiến trình khác.

4) Chia sẻ trang giữa các tiến trình

Cơ chế phân trang cho phép chia sẻ trang giữa các tiến trình. Ví dụ:

- Một máy tính đang thi hành 2 tiến trình đồng thời của chương trình soạn thảo văn bản. Phân đoạn code của 2 tiến trình là hoàn toàn giống nhau, chỉ khác ở phân đoạn data.
- Giả sử đoạn code chiếm 3 trang bộ nhớ, đoạn data chiếm 1 trang → cả 2 tiến trình chiếm 8 trang.
- Có thể chia sẻ đoạn code như sau:



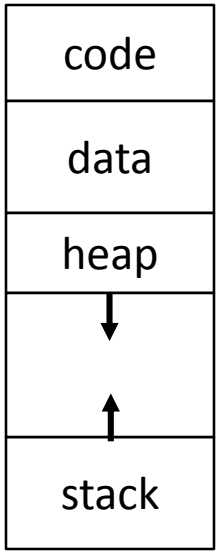
- Số trang cần dùng là 5
- Các file dll của Windows (Dynamic Linking Library) dùng cơ chế chia sẻ trang

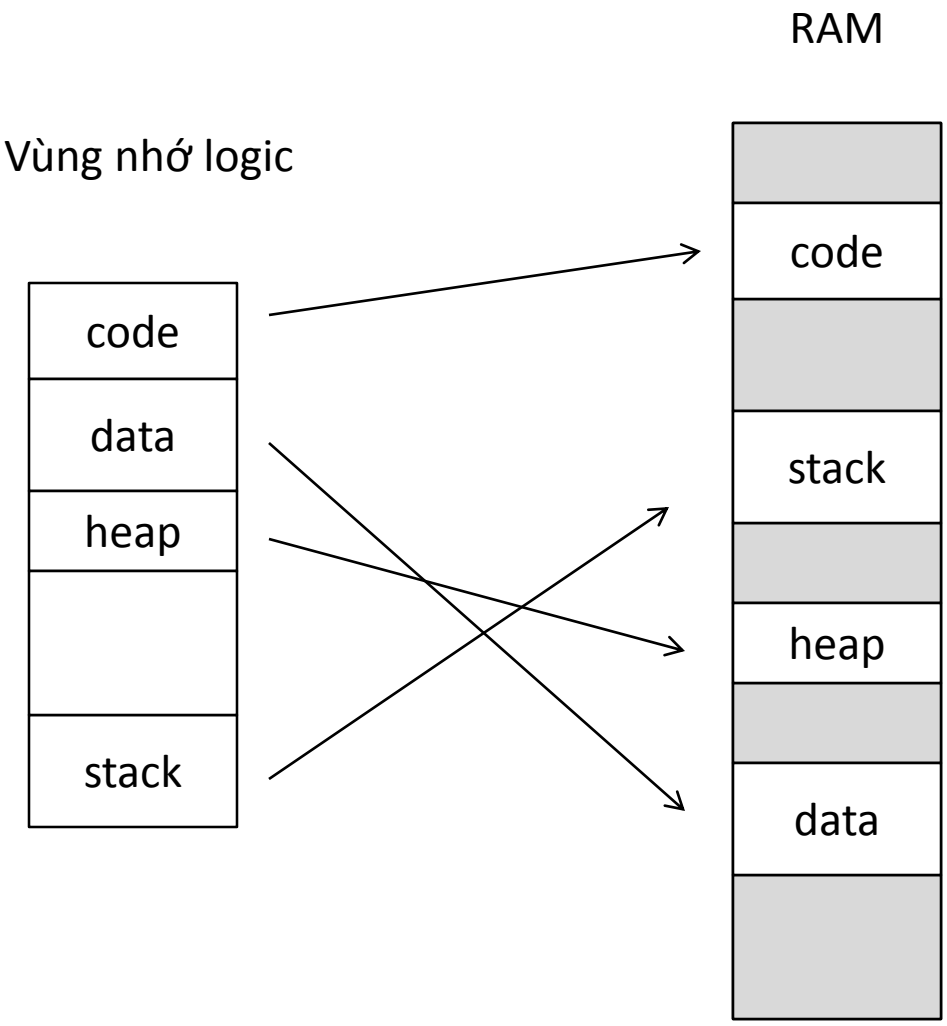
Q & A

IV. Cấp phát bộ nhớ phân đoạn (Segmentation)

1. Đặt vấn đề:
   - Vùng nhớ logic chương trình có thể được chia thành nhiều đoạn (segment): đoạn code, đoạn data, đoạn stack…

   - Vùng code cũng có thể chia làm nhiều đoạn khác nhau: đoạn chương trình chính, đoạn các chương trình con…
   - **Cấp phát bộ nhớ phân đoạn**: các đoạn không cần cấp phát liên tục nhau mà có thể ở các nơi khác nhau trong bộ nhớ.

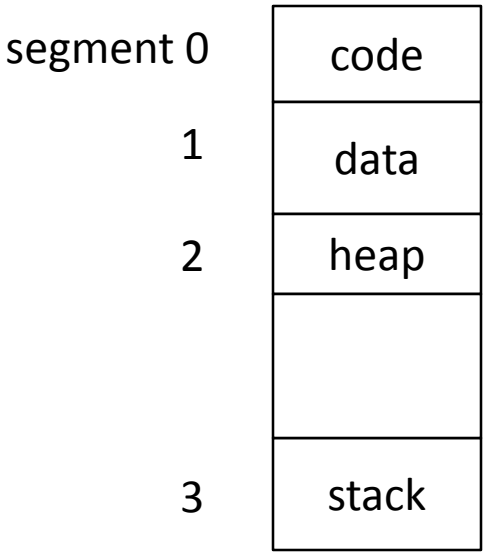



RAM
Vùng nhớ logic
code
data
heap
stack
code
stack
heap
data

2) Cơ chế thực hiện:

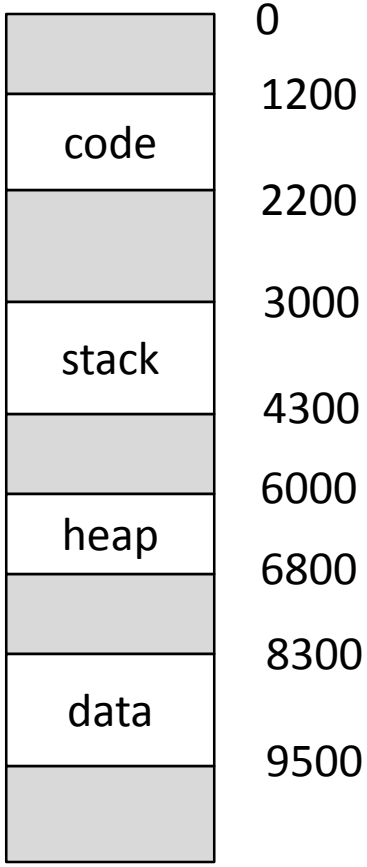
a) Bảng phân đoạn (Segment table)

Dùng để xác định segment nào bắt đầu tại vị trí bộ nhớ nào. Đồng thời xác định giới hạn của segment đó nhằm mục đích bảo vệ không xâm phạm tiến trình khác.



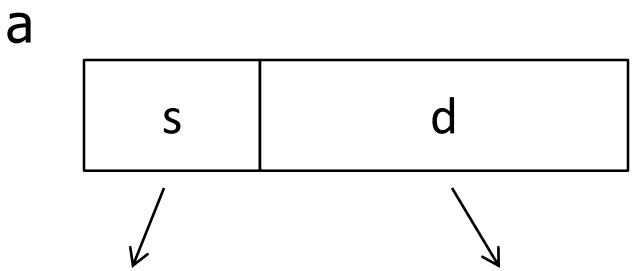
	base	limit
0	1200	1000
1	8300	1200
2	6000	800
3	3000	1300

Bảng phân đoạn



b) Cơ chế đánh địa chỉ logic

Địa chỉ logic a của tiến trình là một số nguyên n bít được chia thành 2 phần:



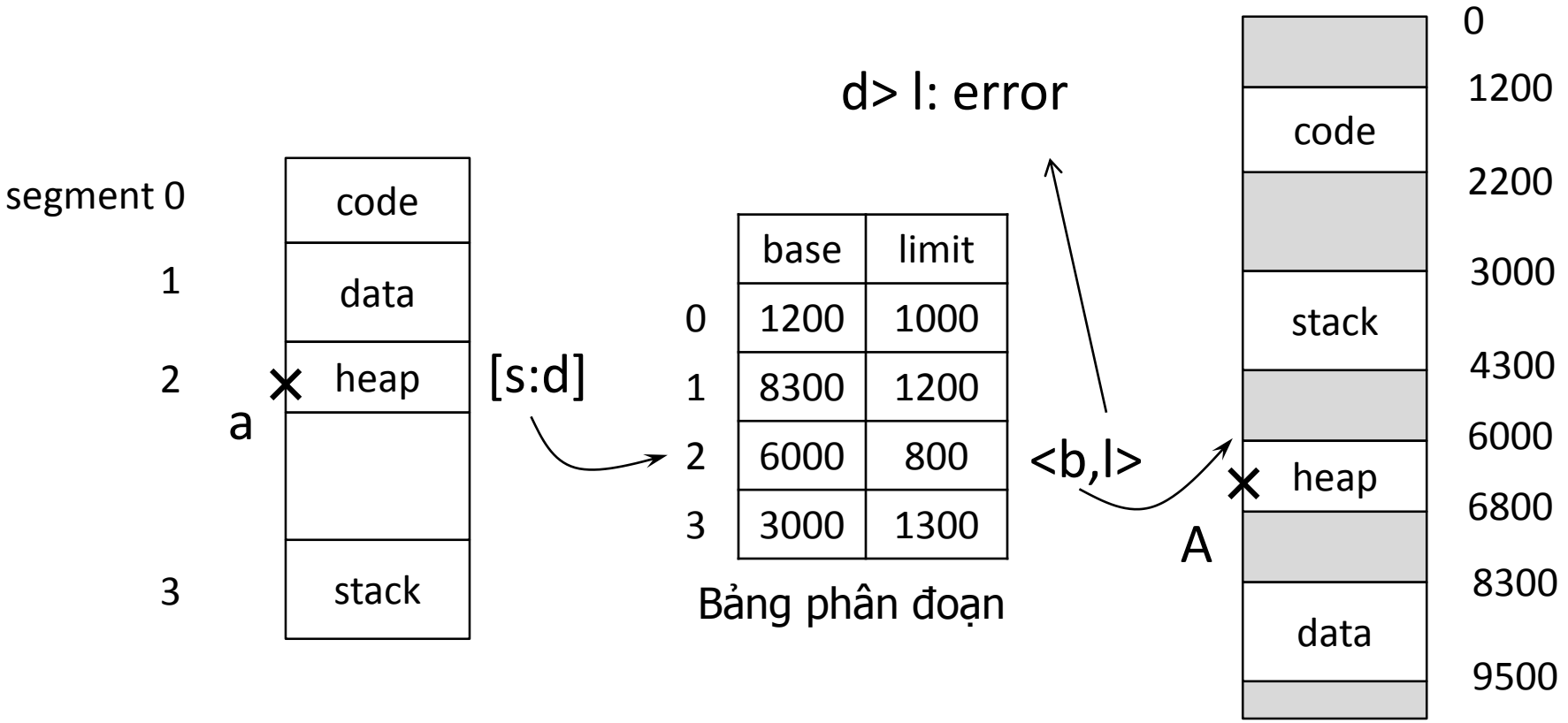
số thứ tự phân đoạn

vị trí byte trong phân đoạn
offset

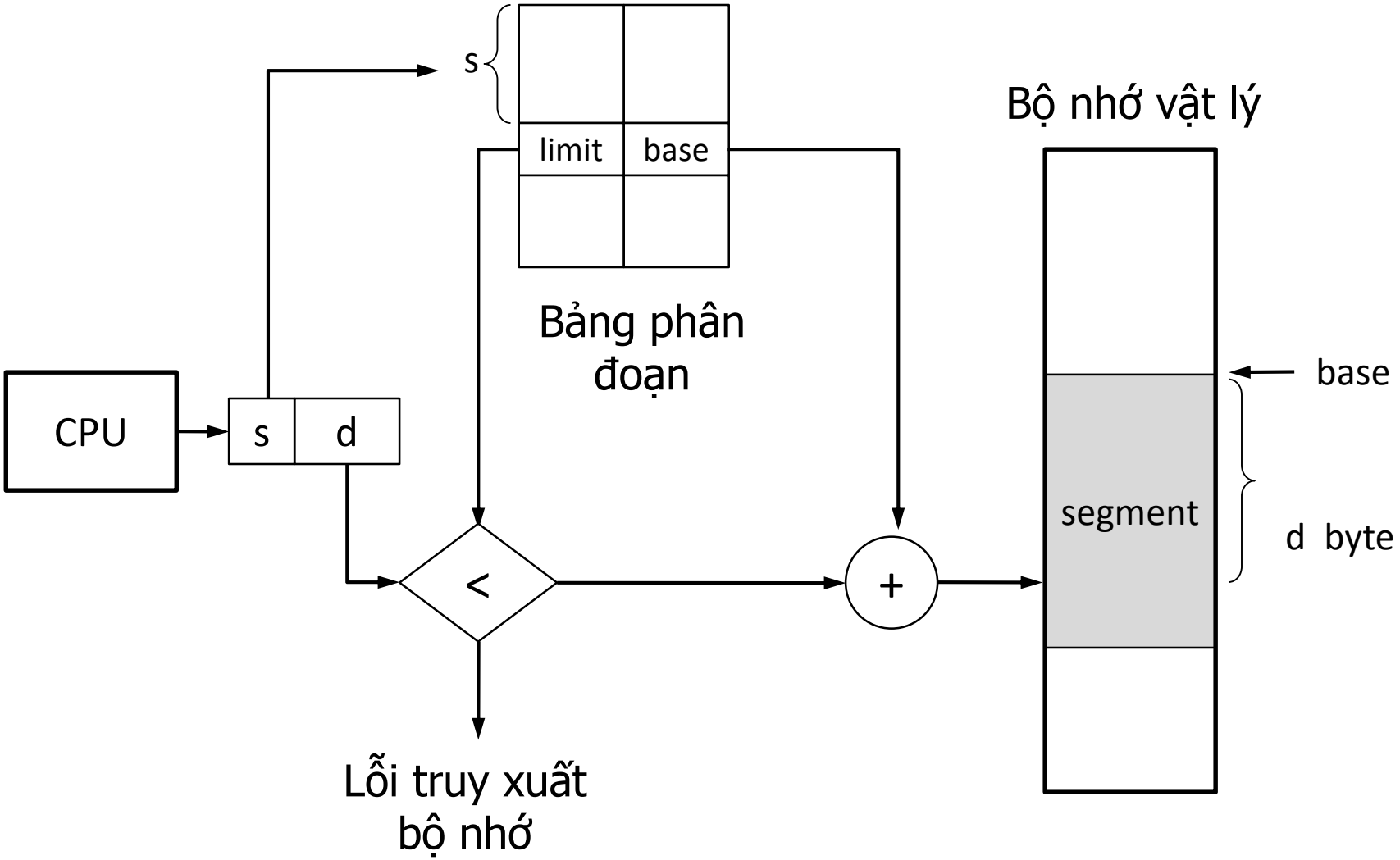
Ký hiệu [segment : offset]

c) Cơ chế phần cứng MMU

Có địa chỉ logic a ([s:d]), muốn tính địa chỉ vật lý A ?



- s $\rightarrow$ <b, l> (dùng bảng segment, tính base và limit)
- Nếu d > l : lỗi địa chỉ vượt kích thước segment
- b $\rightarrow$ A (tính địa chỉ vật lý A = b+d)


s
limit
base
Bảng phân đoạn
Bộ nhớ vật lý
CPU
s
d
<
+
segment
base
d byte
Lỗi truy xuất bộ nhớ

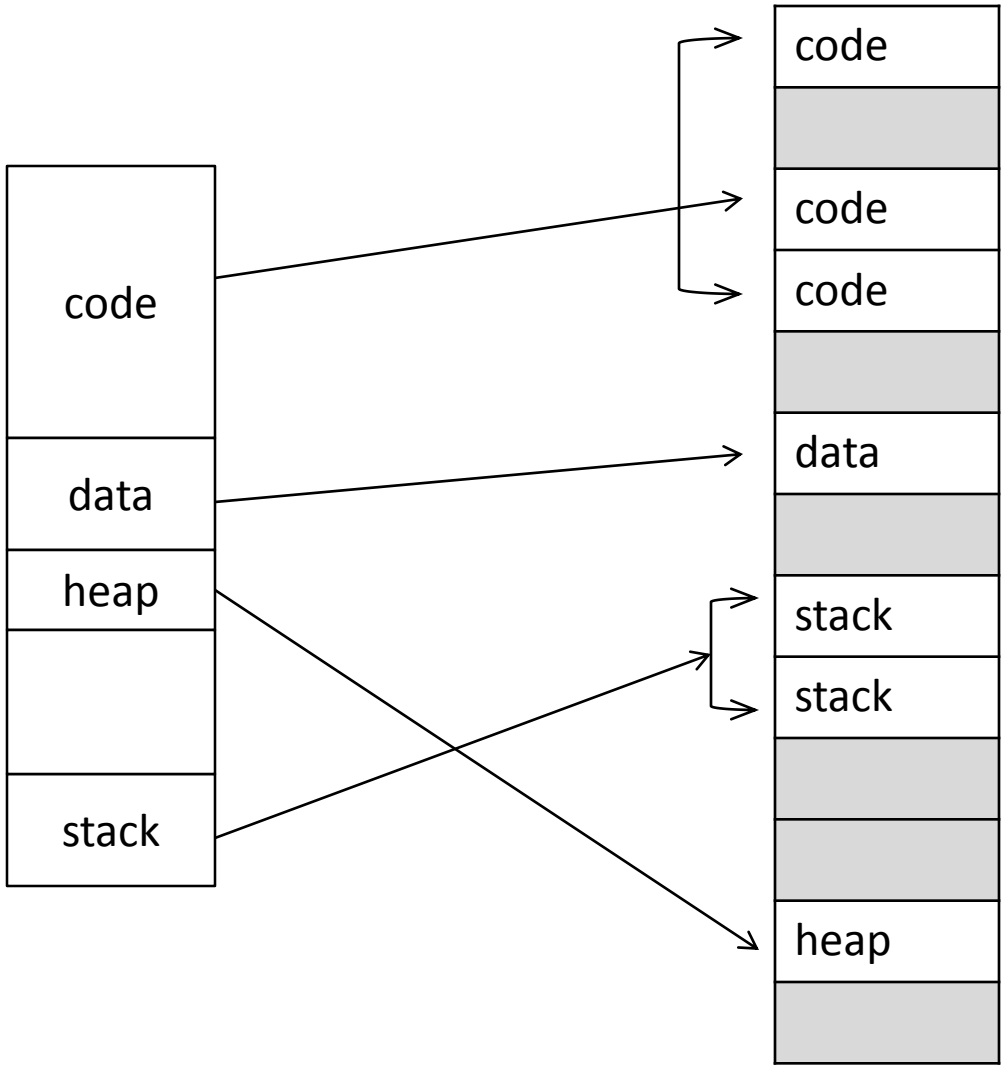
3. Ví dụ về phân đoạn:
   - Kiến trúc vi xử lý Intel 8086 cho phép thực hiện phân đoạn với 4 thanh ghi (thay cho segment table):

   - Thanh ghi CS: xác định phân đoạn code.
   - Thanh ghi DS: phân đoạn data.
   - Thanh ghi ES: phân đoạn data bổ sung. (heap)
   - Thanh ghi SS: phân đoạn stack.

```
MOV ah, [ds:011D]
```

V. Kết hợp phân đoạn và phân trang

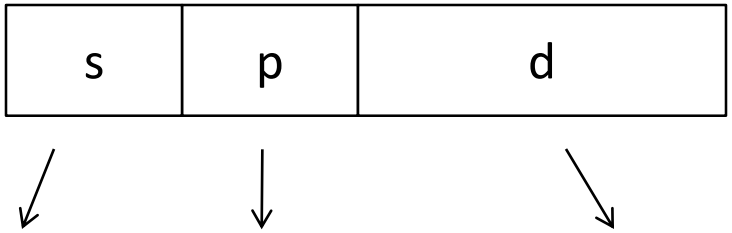
1. Ý tưởng:
   - Trong mô hình phân đoạn, kích thước các đoạn có thể dài → phân mảnh bộ nhớ.
   - Cần thực hiện phân trang cho các đoạn → mỗi đoạn có một bảng phân trang riêng.


code
data
heap
stack
code
code
code
data
stack
stack
heap

2) Cơ chế thực hiện:

a) Cơ chế đánh địa chỉ logic

Địa chỉ logic của tiến trình gồm 3 phần:



số thứ tự phân đoạn

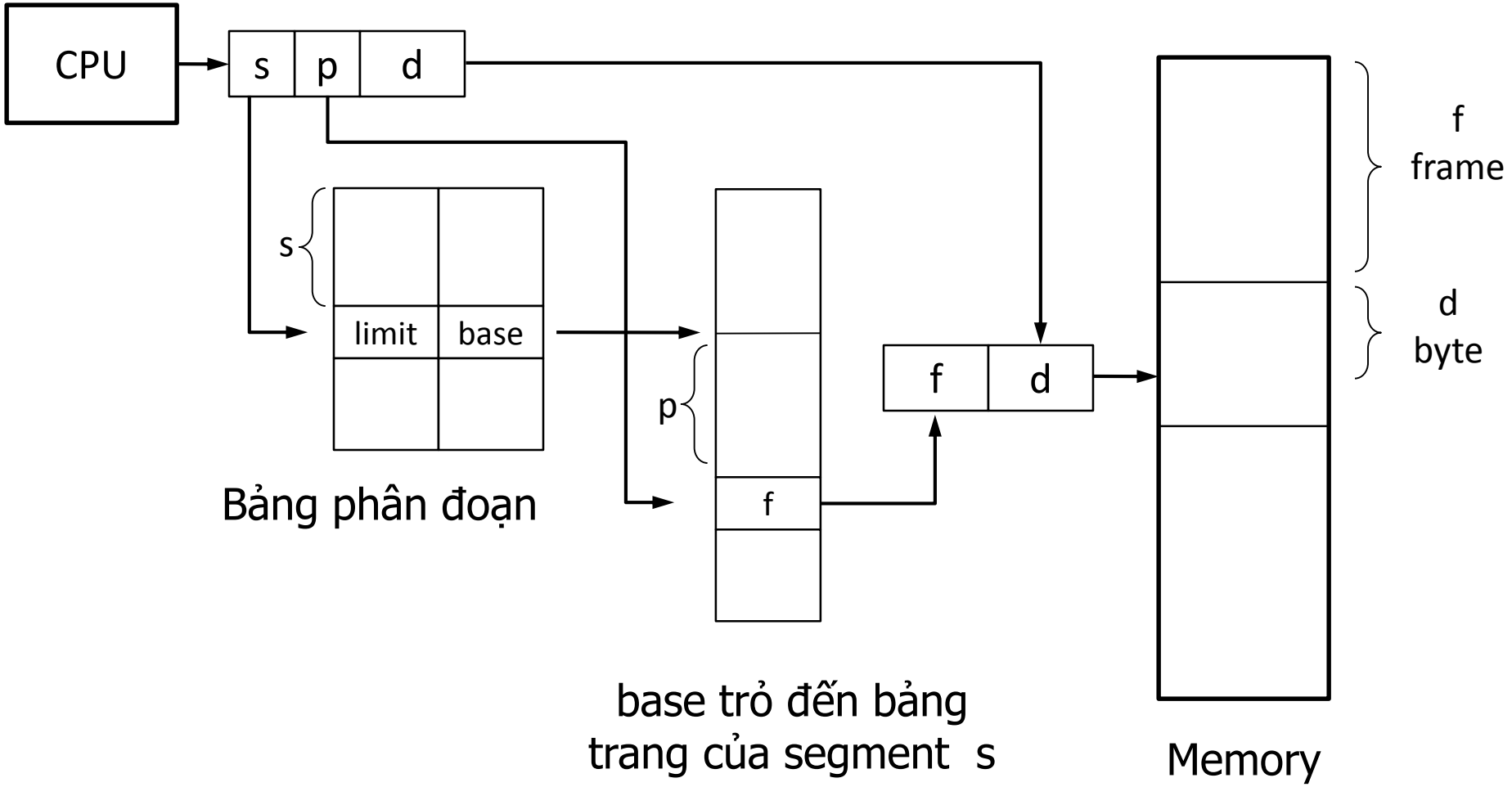
số thứ tự trang trong đoạn

vị trí byte trong trang (offset)

b) Cơ chế chuyển đổi địa chỉ MMU

Sử dụng kết hợp bảng phân đoạn và bảng trang

- Bộ nhớ chính được tổ chức theo mô hình phân trang
- Bảng trang được tổ chức theo mô hình phân đoạn



3. Ví dụ về phân đoạn kết hợp phân trang:
   Kiến trúc vi xử lý Intel Pentium:

- Số lượng segment tối đa cho 1 tiến trình: 16384
- Kích thước tối đa của 1 segment: 4GB
- Sử dụng mô hình phân trang 2 cấp.
- Kích thước của một trang: 2^{12} = 4KB

Q & A

Câu hỏi ôn tập

1. Vùng nhớ logic của 1 tiến trình là gì?
2. Nêu khái niệm và phân biệt sự khác nhau giữa địa chỉ logic và địa chỉ vật lý.
3. Giải thích ý nghĩa của lệnh sau đây trong 3 giai đoạn:
   - Giai đoạn lập trình assembly: `MOV dl, KyTu`
   - Giai đoạn biên dịch mã máy: `MOV dl, [0131]`
   - Giai đoạn CPU thi hành lệnh: `dl` $\leftarrow$ `[0531]`

   (giá trị của thanh ghi định vị là 0400)
4. Trong mô hình cấp phát bộ nhớ liên tục cho hệ đa tiến trình. Sau một thời gian nạp và kết thúc các chương trình thì bộ nhớ xảy ra tình trạng gì? Thao tác nào khắc phục tình trạng đó.

5. Trong mô hình bộ nhớ phân trang, vùng nhớ vật lý dành cho một tiến trình có nhất thiết phải liên tục không? Tại sao?
6. Giải thích sự khác biệt giữa hiện tượng phân mảnh nội vi và phân mảnh ngoại vi.
7. Trong mô hình bộ nhớ phân trang, nếu kích thước trang là lũy thừa của 2 thì có thuận lợi gì?
8. Trong mô hình phân trang, nhờ vào đặc điểm gì mà 2 tiến trình khác nhau có thể cùng chia sẻ một vùng nhớ vật lý?
9. Mô hình phân đoạn có khắc phục được khuyết điểm của hiện tượng phân mảnh ngoại vi không? Tại sao?

Bài tập

1) Trong mô hình cấp phát bộ nhớ liên tục, có bốn phân mảnh bộ nhớ theo thứ tự với kích thước là 500KB, 600KB, 200KB, 300KB. Giả sử có 4 tiến trình đang chờ cấp phát bộ nhớ theo thứ tự P1, P2, P3, P4. Kích thước tương ứng của các tiến trình trên là: 212KB, 417KB, 125KB, 326KB. Hãy cấp phát bộ nhớ cho các tiến trình trên theo thuật toán First-fit, Best-fit, Worst-fit.

2) Một tiến trình được nạp vào bộ nhớ theo mô hình phân trang. Vùng nhớ logic được chia thành 6 trang A, B, C, D, E, F. Bộ nhớ vật lý gồm 10 khung trang. Bảng trang như sau:

6
8
3
5
1
7

Hãy vẽ sơ đồ nạp trang vào bộ nhớ vật lý

3) Một tiến trình được nạp vào bộ nhớ theo mô hình phân trang với kích thước trang là 1024 byte. Bảng trang như sau:

1
5
3
6

Hãy chuyển các địa chỉ logic sau thành địa chỉ vật lý:

a) 1521 b) 3249

4) Xét bảng phân đoạn sau:

Segment	*Base*	*Limit*
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

Cho biết địa chỉ vật lý tương ứng với các địa chỉ logic sau:

a) [0:430]
b) [1:10]
c) [2:500]
d) [3:400]
e) [4:112]

5) Xét mô hình bộ nhớ phân trang có kích thước trang là 1024 byte. Giả sử có 3 tiến trình với kích thước là 1234 byte, 5051 byte, 6279 byte. Xác định tổng kích thước phân mảnh nội vi.

Bài tập thực hành

1. Dùng Assembly viết chương trình Hello World – Universe (ví dụ minh họa đầu chương). Sau khi biên dịch dùng chương trình Turbo Debugger (trên DOS) hay Olly Debugger (trên Win) để xem cách thức nạp chương trình vào bộ nhớ, các phân đoạn code, data (cs, ds)