Chương 3: Quản lý bộ nhớ

Hệ điều hành

ThS. Đinh Xuân Trường

truongdx@ptit.edu.vn

Posts and Telecommunications Institute of Technology

Faculty of Information Technology 1

CNTT1

Học viện Công nghệ Bưu chính Viễn thông

August 15, 2022

Mục lục

Nội dung chương 3

1. Địa chỉ và các vấn đề liên quan
2. Một số cách tổ chức chương trình
3. Các yêu cầu quản lý bộ nhớ
4. Phân chương bộ nhớ
5. Phân trang bộ nhớ
6. Phân đoạn bộ nhớ
7. Bộ nhớ ảo

Phân chương bộ nhớ

- Để thực hiện tiến trình, HĐH cần cấp phát cho tiến trình không gian nhớ cần thiết.
- Việc cấp phát và quản lý vùng nhớ là chức năng quan trọng của HĐH
- Một kỹ thuật cấp phát đơn giản nhất là mỗi tiến trình được cấp một vùng bộ nhớ liên tục
- HĐH tiến hành chia bộ nhớ thành các phần liên tục là chương (partition), mỗi tiến trình sẽ được cung cấp một chương để chứa lệnh và dữ liệu của mình.
- Quá trình phân chia bộ nhớ thành chương như vậy gọi là *phân chương bộ nhớ*.
- Tùy thuộc việc lựa chọn vị trí và kích thước của chương, có thể phân biệt phân chương cố định và phân chương động

Phân chương bộ nhớ (cont.)

Các chiến lược quản lý bộ nhớ:

- Phân chương cố định
- Phân chương động
- Phân trang
- Phân đoạn
- Kết hợp phân đoạn - phân trang

Nguyên tắc

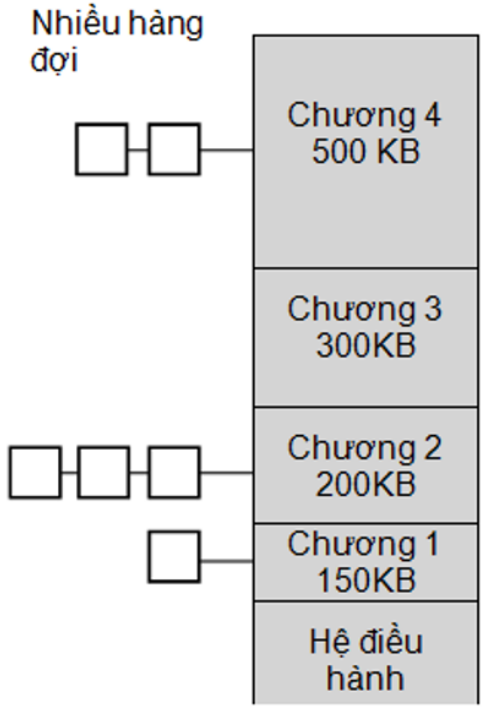
- Bộ nhớ được chia thành n phần
  - Mỗi phần được gọi là một chương (partition)
  - Mỗi chương ở một vị trí cố định
  - Chương được sử dụng như một vùng nhớ độc lập
    - Mỗi chương chứa được đúng một tiến trình
    - Khi được tải vào, tiến trình được cấp phát một chương. Sau khi tiến trình kết thúc, HĐH giải phóng chương và chương có thể được cấp phát cho tiến trình mới.
  - Chương có thể có kích thước bằng nhau hoặc khác nhau

Ví dụ: Xét hệ thống dưới đây bộ nhớ được tổ chức theo phân chương cố định, tính thời gian HĐH thực hiện xong các tiến trình sau:

Process	Size	time
P_1	120	20
P_2	80	15
P_3	70	5
P_4	50	5
P_5	140	12
Hàng đợi		

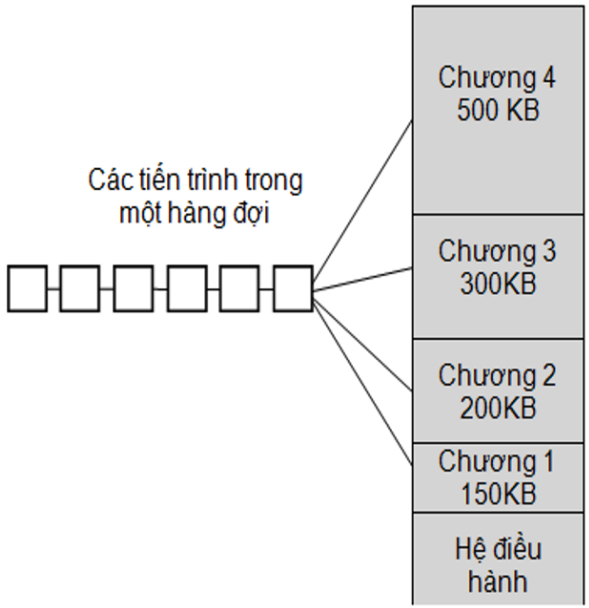
- Kích thước các chương bằng nhau:
  - Ưu điểm: Đơn giản
  - Nhược điểm: Kích thước chương trình > kích thước chương dẫn đến không thể cấp phát
  - Gây phân mảnh trong
- Kích thước các chương khác nhau:
  - Có hai cách lựa chọn chương nhớ để cấp cho tiến trình đang chờ đợi bằng cách chọn chương có kích thước nhỏ nhất
    - Mỗi chương có một hàng đợi riêng
    - Một hàng đợi chung cho tất cả các chương

Mỗi chương có một hàng đợi riêng: *tiến trình có kích thước phù hợp với chương nào sẽ nằm trong hàng đợi của chương đó*



- Ưu điểm: Tiết kiệm bộ nhớ, giảm phân mảnh trong
- Nhược điểm: Hệ thống không tối ưu, có thời điểm hàng đợi chương lớn thì rỗng, hàng đợi chương nhỏ hơn chứa nhiều tiến trình.

Một hàng đợi chung cho tất cả các chương: *Mỗi khi có một chương trống tiến trình nằm gần đầu hàng đợi nhất và có kích thước phù hợp với chương nhất sẽ được tải và thực hiện*



- Ưu điểm: Tiết kiệm bộ nhớ, giảm phân mảnh trong và tối ưu hệ thống

Nhận xét về phân chương cố định

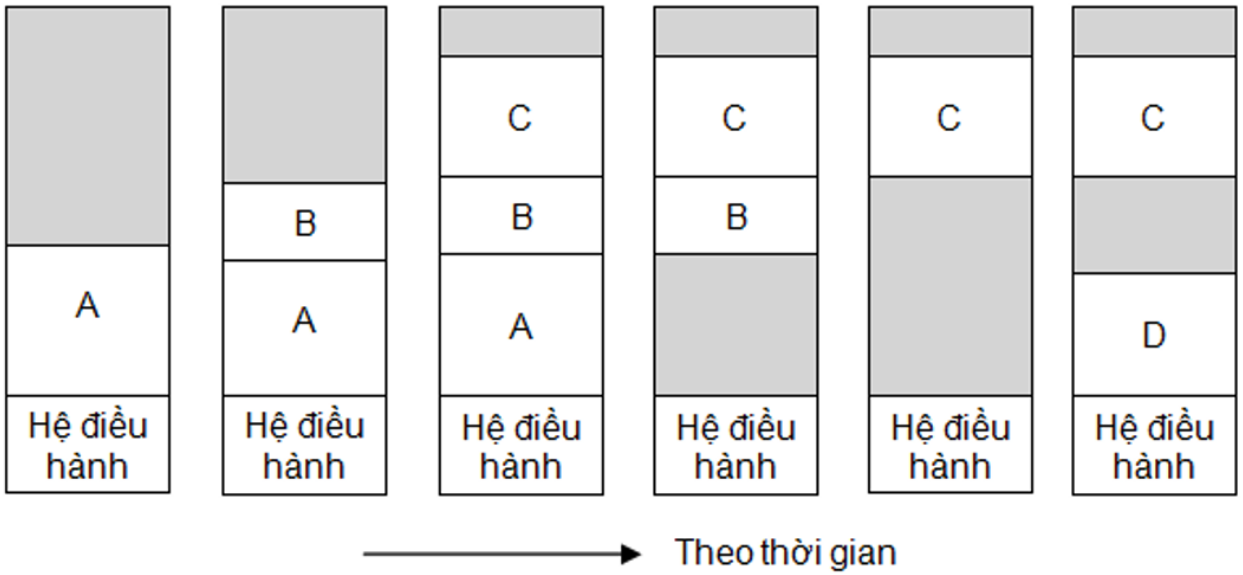
- Ưu điểm
  - Đơn giản và ít xử lý
  - Giảm thời gian tìm kiếm
- Nhược điểm
  - Hệ số song song không thể vượt quá số lượng chương của bộ nhớ
  - Bị phân đoạn bộ nhớ
    - Kích thước chương trình lớn hơn kích thước chương lớn nhất
    - Tổng bộ nhớ tự do còn lớn, nhưng không dùng để nạp các chương trình khác

Nguyên tắc

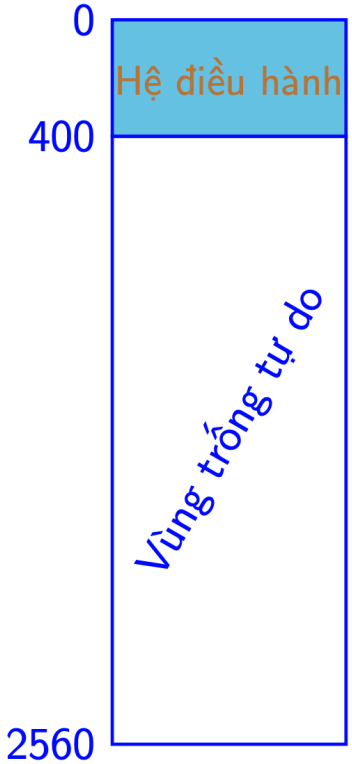
- Kích thước, số lượng và vị trí chương đều có thể thay đổi
- Các vùng trống bộ nhớ cũng có thể được liên kết thành một danh sách kết nối
- Khi một tiến trình yêu cầu bộ nhớ
  - HĐH cấp cho tiến trình 1 chương có kích thước đúng bằng tiến trình đó đang cần
  - Nếu tìm thấy vùng nhớ tương ứng
    - Vùng trống được chia thành 2 phần
    - Một phần cung cấp theo yêu cầu
    - Một phần trả lại danh sách vùng trống tự do
  - Nếu không tìm thấy
    - Phải chờ tới khi có được một vùng trống thỏa mãn

- Cho phép tiến trình khác trong hàng đợi thực hiện (nếu độ ưu tiên đảm bảo)

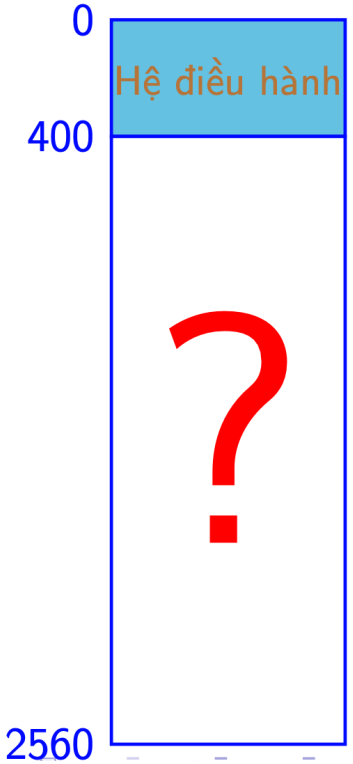
- Khi tiến trình kết thúc sẽ tạo vùng trống trong Bộ nhớ và các vùng trống nằm cạnh nhau được nhập lại thành vùng lớn hơn



Ví dụ: Xét hệ thống dưới đây bộ nhớ được tổ chức theo phân chương động, tính thời gian HĐH thực hiện xong các tiến trình sau:



Process	Size	time
P_1	600	10
P_2	1000	5
P_3	300	20
P_4	700	8
P_5	500	15
File đợi		



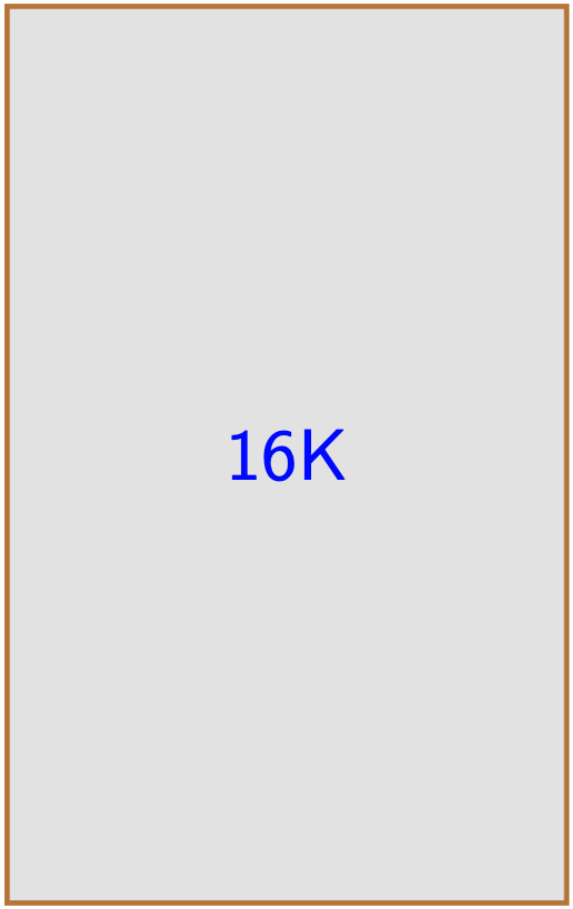
- Phân chương động tránh việc phân mảnh trong
- Gây phân mảnh ngoài: dồn những vùng trống nhỏ thành lớn (nén)
- Các chiến lược cấp chương chọn vùng chống:
  - **First Fit**: Chọn vùng thích hợp đầu tiên
  - **Best Fit**: Vùng thích hợp nhất
  - **Worst Fit**: Vùng trống thích hợp nhất - vùng kích thước lớn nhất

Phương pháp kề cận - Buddy system

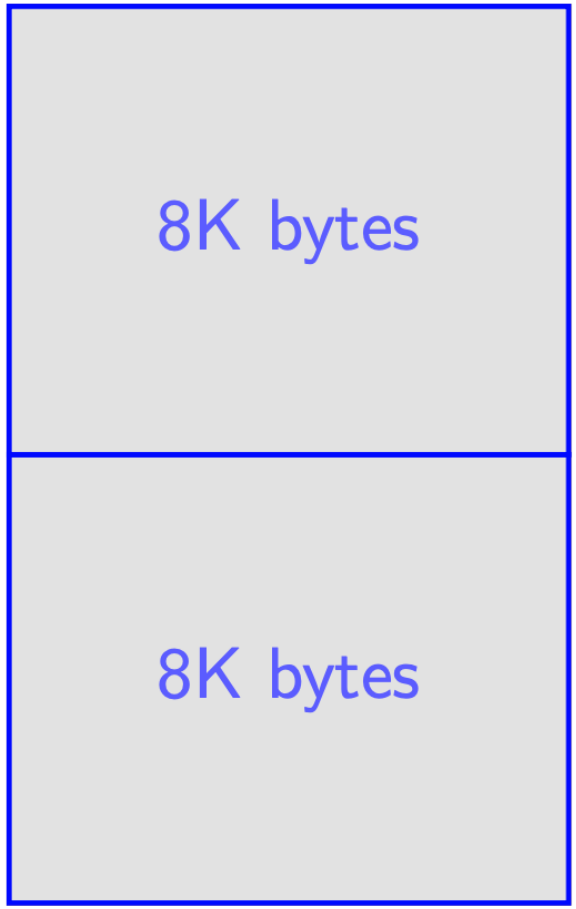
Nguyên tắc: *Chia đôi liên tiếp vùng trống tự do cho tới khi thu được vùng trống nhỏ nhất thỏa mãn*

- Các chương và khối trống có kích thước là $2^k (L \leq k \leq H)$
- 2^L: kích thước nhỏ nhất của chương
- 2^H : kích thước của toàn bộ không gian nhớ
- Yêu cầu cấp vùng nhớ S
  - $2^{H-1} < S \leq 2^H$: cấp cả 2^H
  - $S \leq 2^{H-1}$ Chia vùng trống tìm được thành 2 khối bằng nhau (gọi là buddies) 2^{H-1}
    - Nếu $2^{H-2} < S \leq 2^{H-1}$: cấp cả 2^{H-1}
    - Tiếp tục chia đôi tới khi tìm được vùng thỏa mãn $2^{k-1} < S \leq 2^k$

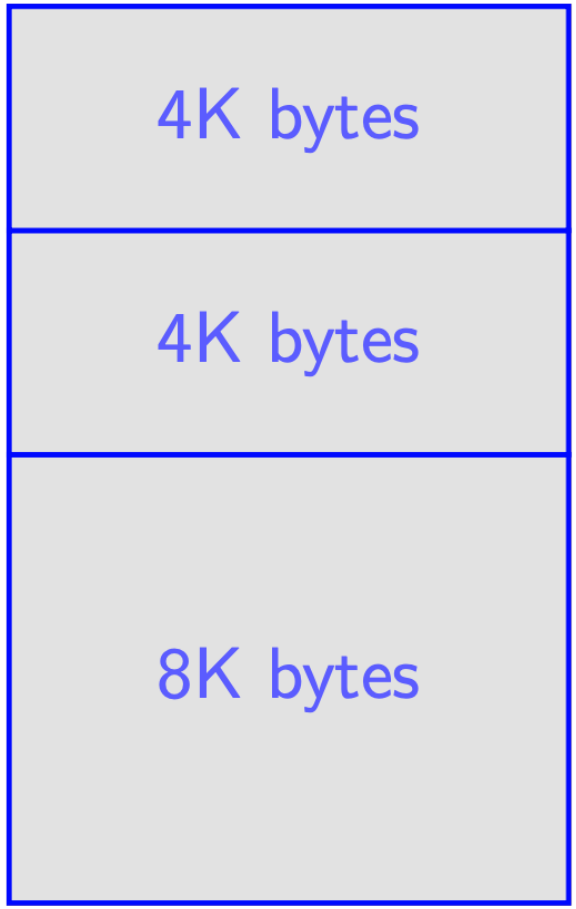
Ví dụ: Vùng trống 16K Bytes, yêu cầu cấp phát một chương có kích thước 735 Bytes



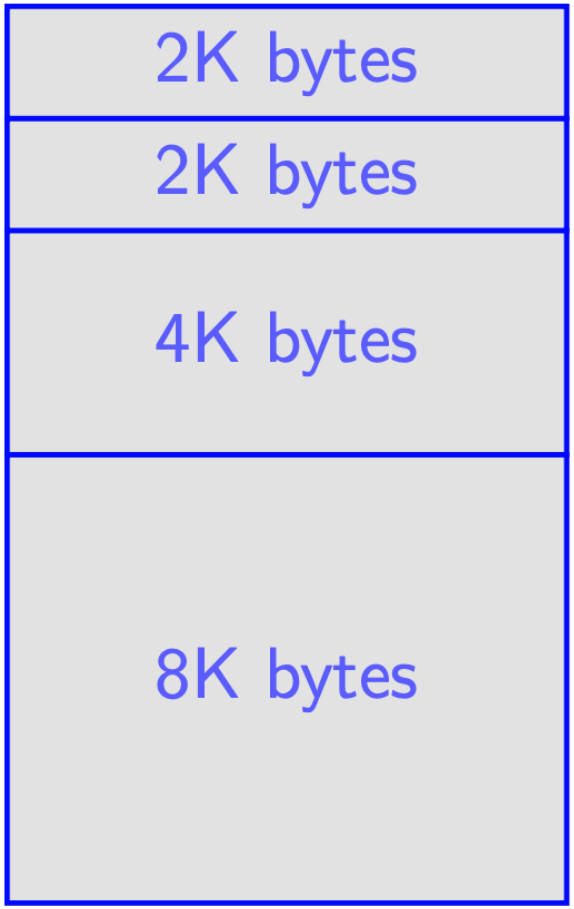
Ví dụ: Vùng trống 16K Bytes, yêu cầu cấp phát một chương có kích thước 735 Bytes



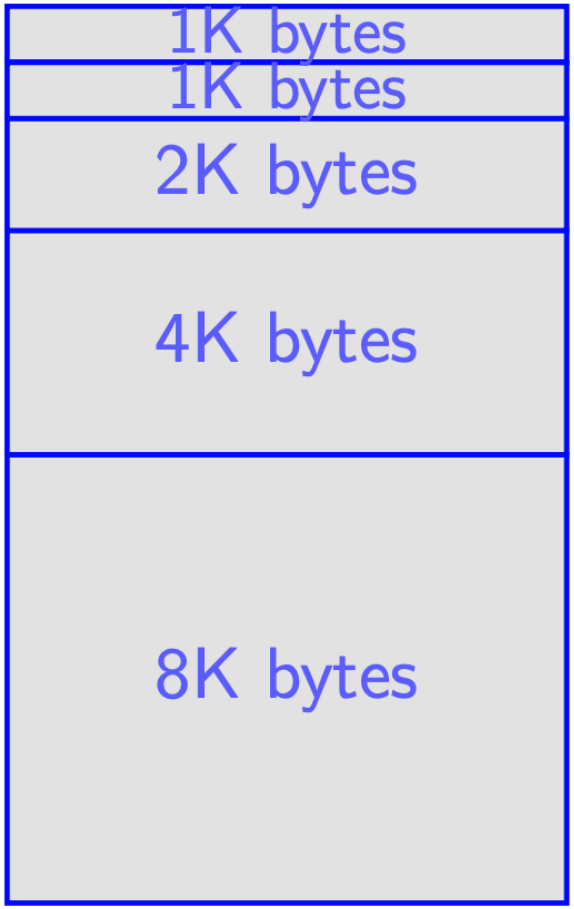
Ví dụ: Vùng trống 16K Bytes, yêu cầu cấp phát một chương có kích thước 735 Bytes



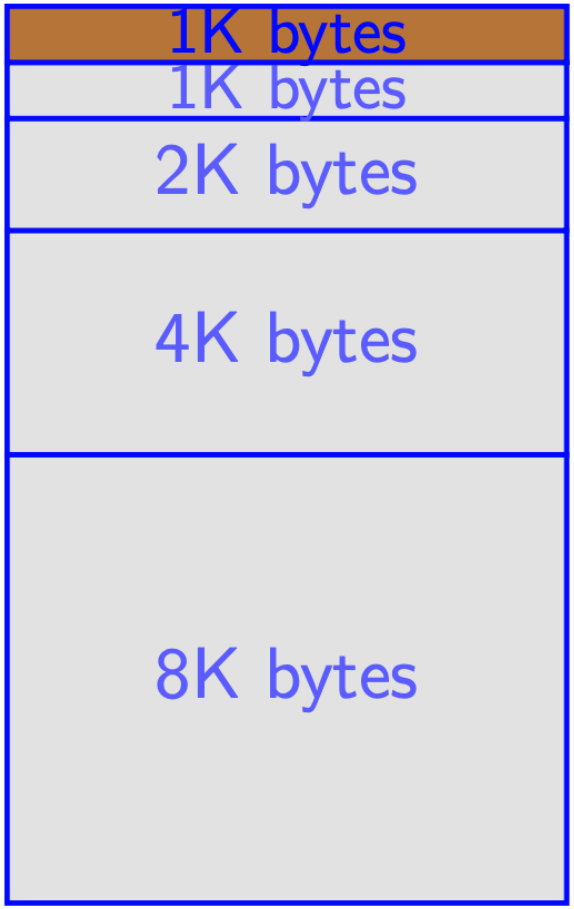
Ví dụ: Vùng trống 16K Bytes, yêu cầu cấp phát một chương có kích thước 735 Bytes



Ví dụ: Vùng trống 16K Bytes, yêu cầu cấp phát một chương có kích thước 735 Bytes



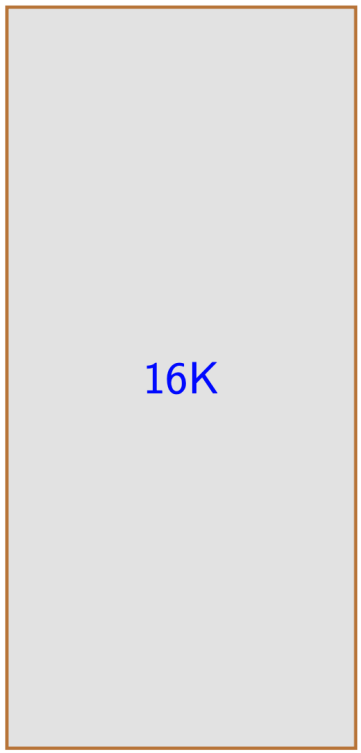
Ví dụ: Vùng trống 16K Bytes, yêu cầu cấp phát một chương có kích thước 735 Bytes



Phương pháp kề cận cung cấp bộ nhớ nhanh: với bộ nhớ kích thước n, cần duyệt $log_2 n$ danh sách

Ví dụ: Vùng trống 16K Bytes, cấp phát một chương có kích thước

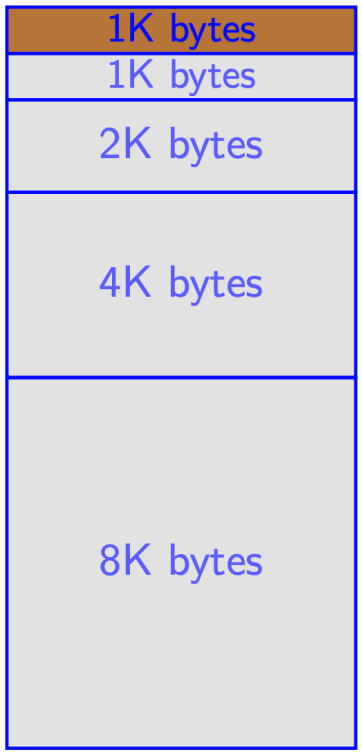
- 835 Bytes



Phương pháp kề cận cung cấp bộ nhớ nhanh: với bộ nhớ kích thước n, cần duyệt $log_2 n$ danh sách

Ví dụ: Vùng trống 16K Bytes, cấp phát một chương có kích thước

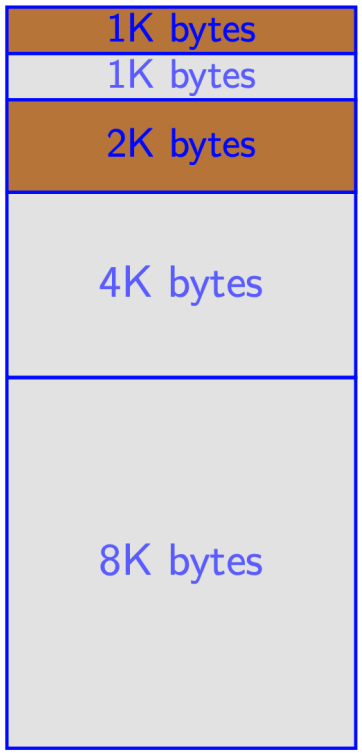
- 835 Bytes



Phương pháp kề cận cung cấp bộ nhớ nhanh: với bộ nhớ kích thước n, cần duyệt $log_2 n$ danh sách

Ví dụ: Vùng trống 16K Bytes, cấp phát một chương có kích thước

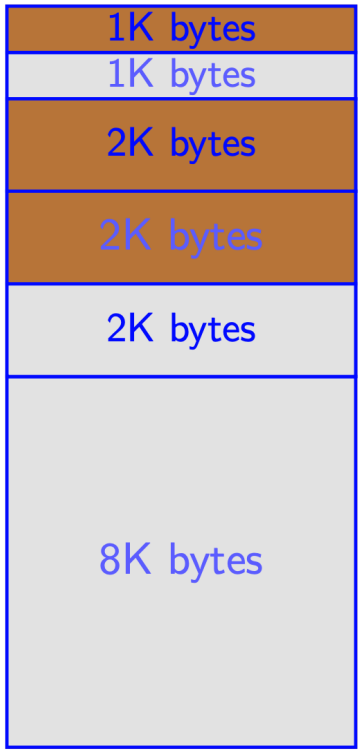
- 1101 Bytes



Phương pháp kề cận cung cấp bộ nhớ nhanh: với bộ nhớ kích thước n, cần duyệt $log_2 n$ danh sách

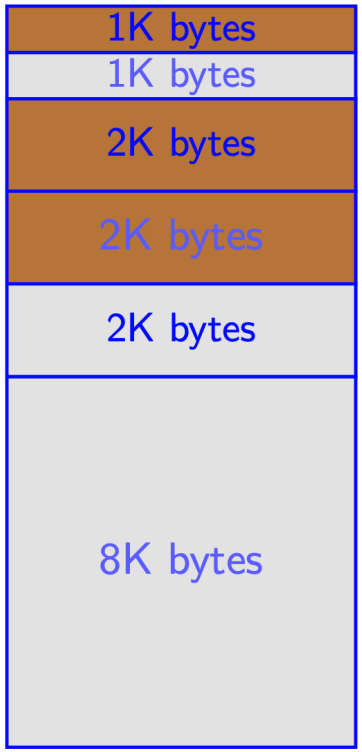
Ví dụ: Vùng trống 16K Bytes, cấp phát một chương có kích thước

- 2022 Bytes



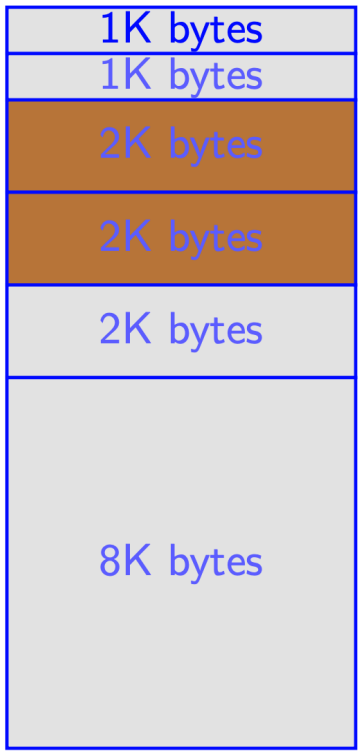
Thu hồi vùng nhớ:
Có thể kết hợp 2 vùng kề nhau có cùng kích thước, tiếp tục kết hợp liên tiếp cho tới khi tạo ra vùng trống lớn nhất có thể.
Ví dụ:

- Giải phóng vùng nhớ thứ nhất (1K)



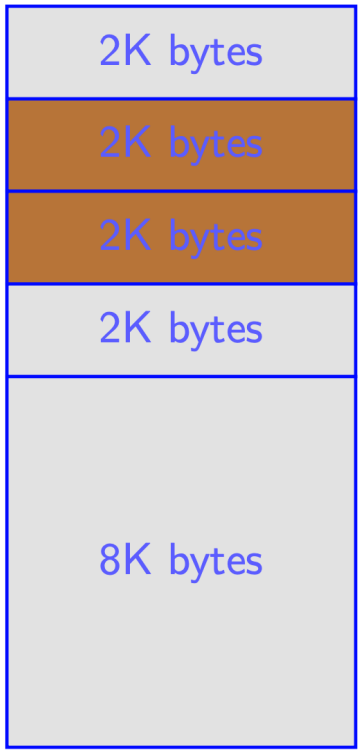
Thu hồi vùng nhớ:
Có thể kết hợp 2 vùng kề nhau có cùng kích thước, tiếp tục kết hợp liên tiếp cho tới khi tạo ra vùng trống lớn nhất có thể.
Ví dụ:

- Giải phóng vùng nhớ thứ nhất (1K)



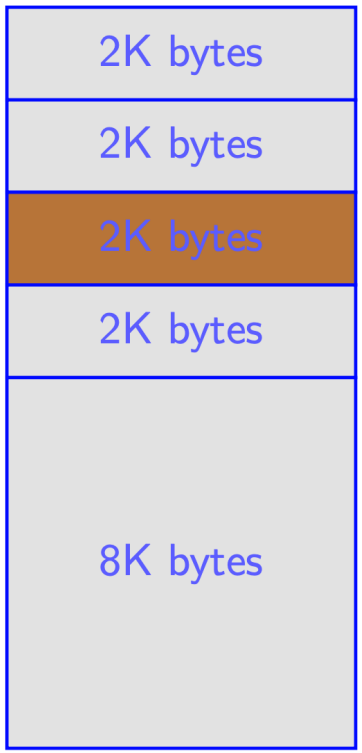
Thu hồi vùng nhớ:
Có thể kết hợp 2 vùng kề nhau có cùng kích thước, tiếp tục kết hợp liên tiếp cho tới khi tạo ra vùng trống lớn nhất có thể.
Ví dụ: Giải phóng vùng nhớ thứ nhất (1K)

- Kết hợp 2 vùng 1K thành vùng 2K



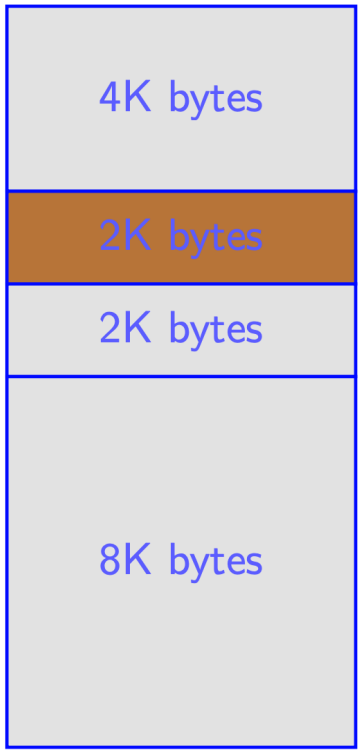
Thu hồi vùng nhớ:
Có thể kết hợp 2 vùng kề nhau có cùng kích thước, tiếp tục kết hợp liên tiếp cho tới khi tạo ra vùng trống lớn nhất có thể.
Ví dụ:

- Giải phóng vùng nhớ thứ hai (2K)



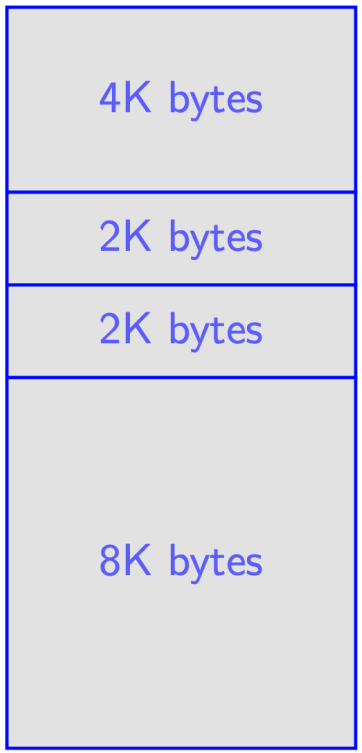
Thu hồi vùng nhớ:
Có thể kết hợp 2 vùng kề nhau có cùng kích thước, tiếp tục kết hợp liên tiếp cho tới khi tạo ra vùng trống lớn nhất có thể.
Ví dụ: Giải phóng vùng nhớ thứ hai (2K)

- Kết hợp 2 vùng 2K thành vùng 4K



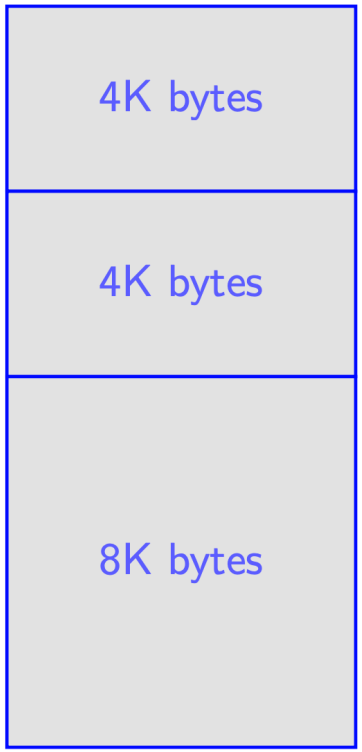
Thu hồi vùng nhớ:
Có thể kết hợp 2 vùng kề nhau có cùng kích thước, tiếp tục kết hợp liên tiếp cho tới khi tạo ra vùng trống lớn nhất có thể.
Ví dụ:

- Giải phóng vùng nhớ thứ ba (2K)



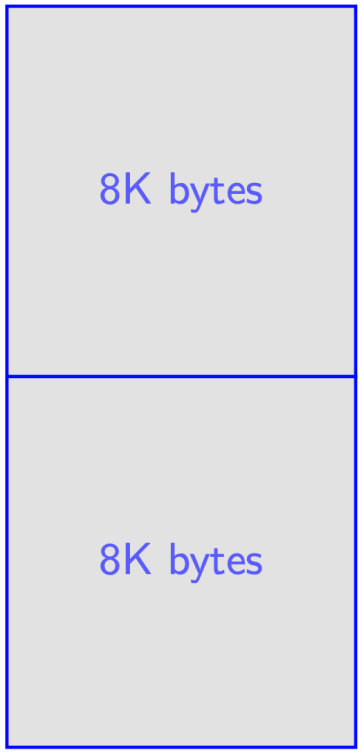
Thu hồi vùng nhớ:
Có thể kết hợp 2 vùng kề nhau có cùng kích thước, tiếp tục kết hợp liên tiếp cho tới khi tạo ra vùng trống lớn nhất có thể.
Ví dụ: Giải phóng vùng nhớ thứ ba (2K)

- Kết hợp 2 vùng 2K thành vùng 4K



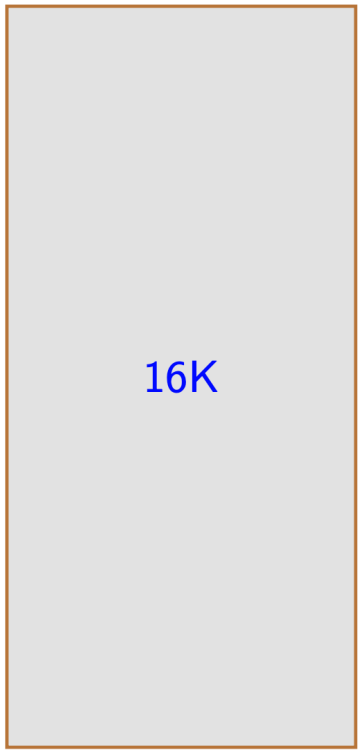
Thu hồi vùng nhớ:
Có thể kết hợp 2 vùng kề nhau có cùng kích thước, tiếp tục kết hợp liên tiếp cho tới khi tạo ra vùng trống lớn nhất có thể.
Ví dụ: Giải phóng vùng nhớ thứ ba (2K)

- Kết hợp 2 vùng 4K thành vùng 8K

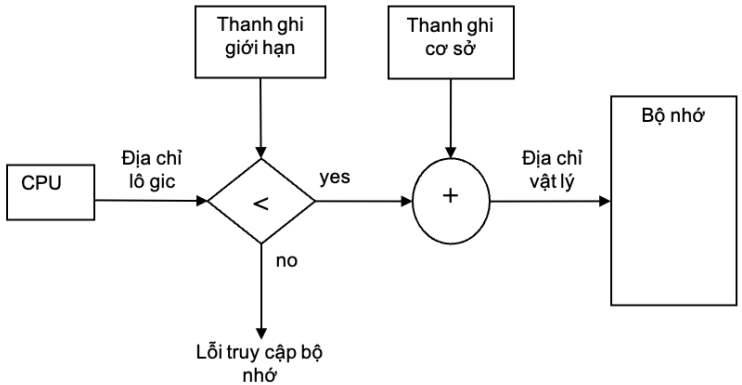


Thu hồi vùng nhớ:
Có thể kết hợp 2 vùng kề nhau có cùng kích thước, tiếp tục kết hợp liên tiếp cho tới khi tạo ra vùng trống lớn nhất có thể.
Ví dụ: Giải phóng vùng nhớ thứ ba (2K)

- Kết hợp 2 vùng 8K thành vùng 16K



- Khi tiến trình được tải vào bộ nhớ, CPU dành 2 thanh ghi:
  - Thanh ghi cơ sở: chứa địa chỉ bắt đầu của tiến trình
  - Thanh ghi giới hạn: chứa độ dài chương



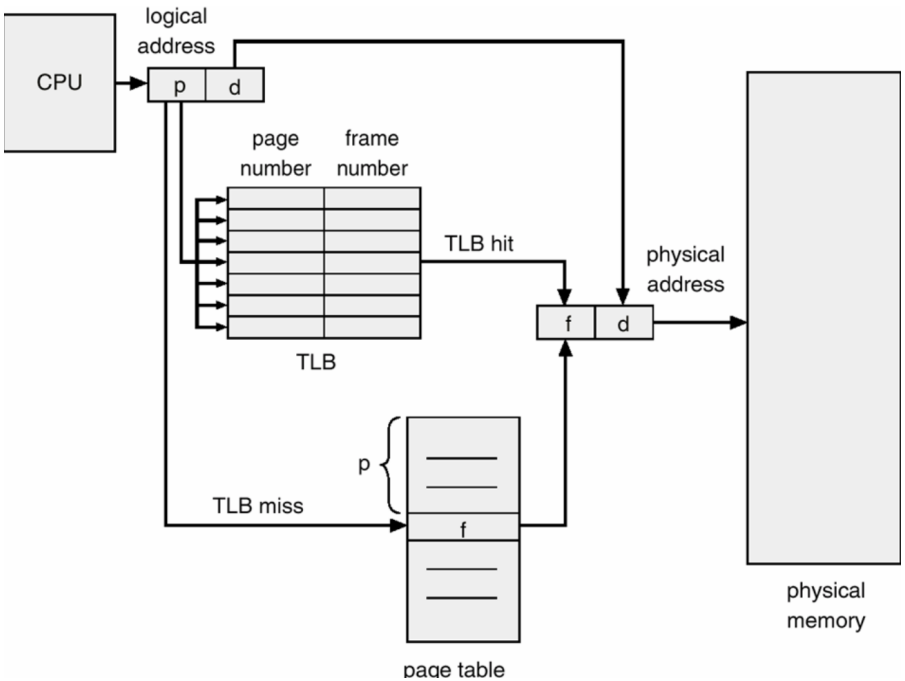
- Địa chỉ logic được so sánh với nội dung của thanh ghi giới hạn
  - Nếu lớn hơn: lỗi truy cập
  - Nhỏ hơn: đưa tới bộ cộng với thanh ghi cơ sở thành địa chỉ vật lý
- Nếu chương bị di chuyển thì nội dung của thanh ghi cơ sở bị thay đổi chứa địa chỉ mới

Phương pháp tráo đổi (*swapping*):

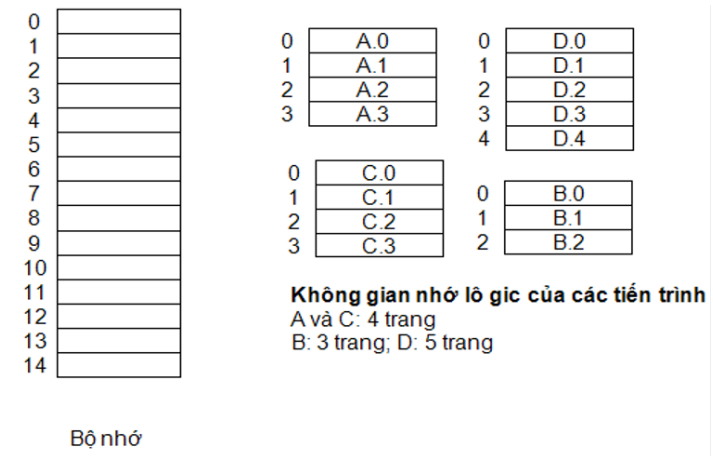
- Các tiến trình đang thực hiện có thể bị tạm thời tải ra đĩa nhường chỗ để tải các tiến trình khác vào
- Sau đó lại được tải vào (nếu chưa kết thúc) để thực hiện tiếp
  - Một tiến trình đã hết khoảng thời gian sử dụng CPU của mình
  - Nhường chỗ cho một tiến trình khác có thứ tự ưu tiên cao hơn
- Thời gian tải phụ thuộc vào tốc độ truy cập đĩa, tốc độ truy cập bộ nhớ và kích thước tiến trình
- Khi được tải vào lại, tiến trình có thể được chứa vào chương ở vị trí cũ hoặc được cấp cho một chương địa chỉ hoàn toàn mới
- Các tiến trình bị trao đổi phải ở trạng thái nghỉ, đặc biệt không thực hiện các thao tác vào ra

Phân trang bộ nhớ

Trong kỹ thuật phân chương, mỗi tiến trình chiếm một chương tức là một vùng liên tục trong bộ nhớ dẫn đến phân mảnh và không tận dụng hết được bộ nhớ.
Kỹ thuật phân trang (*paging*) để hạn chế nhược điểm của kỹ thuật phân chương:

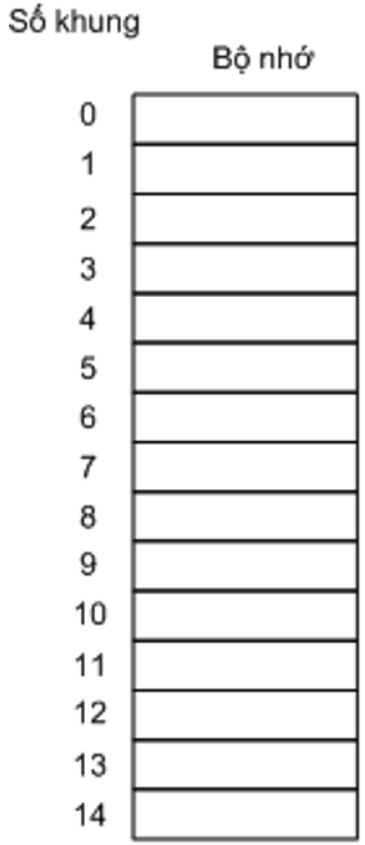


- Bộ nhớ vật lý được chia thành từng khối có kích thước bằng nhau: *khung trang vật lý (page frame)*:
  - Khung trang vật lý được đánh số 0,1,2,... : địa chỉ vật lý của khung trang
  - Khung trang được dùng làm đơn vị phân phối nhớ
- Không gian địa chỉ logic của tiến trình hay bộ nhớ logic (chương trình) được chia thành những khối gọi là *trang (page)* có kích thước bằng khung trang.



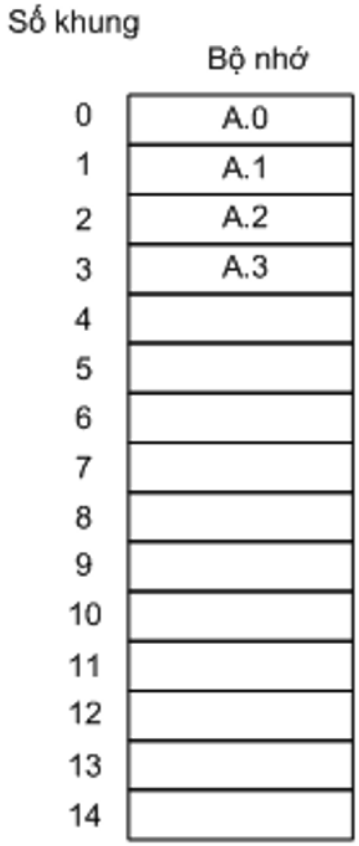
- Tiến trình được cấp các khung để chứa các trang của mình.
- Trang có thể chứa trong các khung nằm rải rác trong bộ nhớ

Ví dụ: Không gian nhớ cho các tiến trình A, B, C và D:



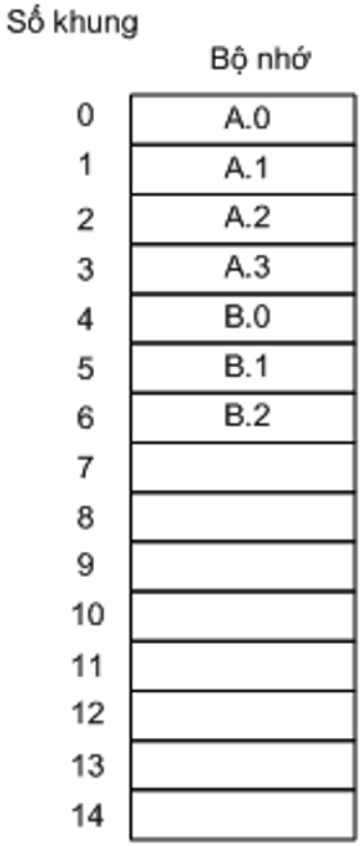
- Tiến trình được cấp các khung để chứa các trang của mình.
- Trang có thể chứa trong các khung nằm rải rác trong bộ nhớ

Ví dụ: Không gian nhớ cho các tiến trình A, B, C và D:



- Tiến trình được cấp các khung để chứa các trang của mình.
- Trang có thể chứa trong các khung nằm rải rác trong bộ nhớ

Ví dụ: Không gian nhớ cho các tiến trình A, B, C và D:



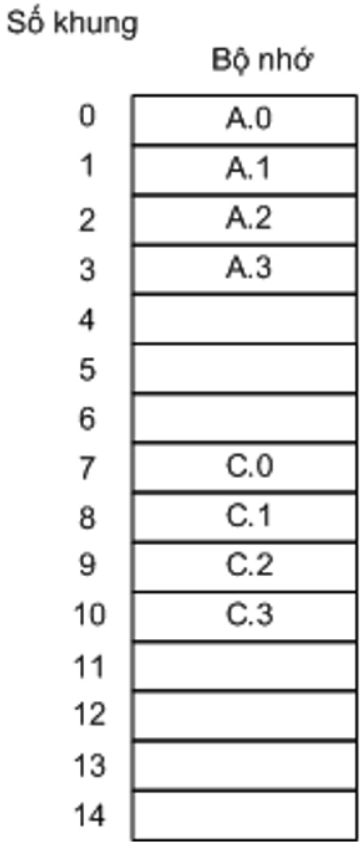
- Tiến trình được cấp các khung để chứa các trang của mình.
- Trang có thể chứa trong các khung nằm rải rác trong bộ nhớ

Ví dụ: Không gian nhớ cho các tiến trình A, B, C và D:

Số khung	Bộ nhớ
0	A.0
1	A.1
2	A.2
3	A.3
4	B.0
5	B.1
6	B.2
7	C.0
8	C.1
9	C.2
10	C.3
11	
12	
13	
14	

- Tiến trình được cấp các khung để chứa các trang của mình.
- Trang có thể chứa trong các khung nằm rải rác trong bộ nhớ

Ví dụ: Không gian nhớ cho các tiến trình A, B, C và D:

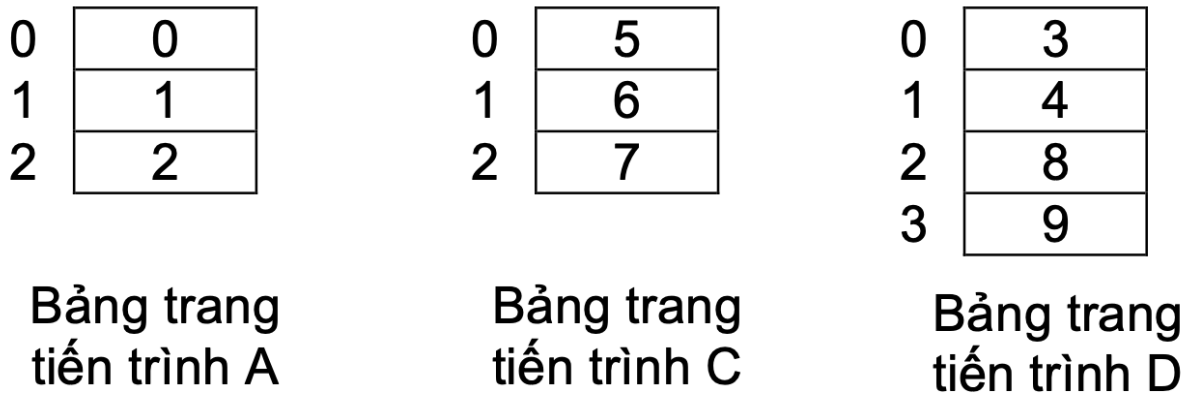


- Tiến trình được cấp các khung để chứa các trang của mình.
- Trang có thể chứa trong các khung nằm rải rác trong bộ nhớ

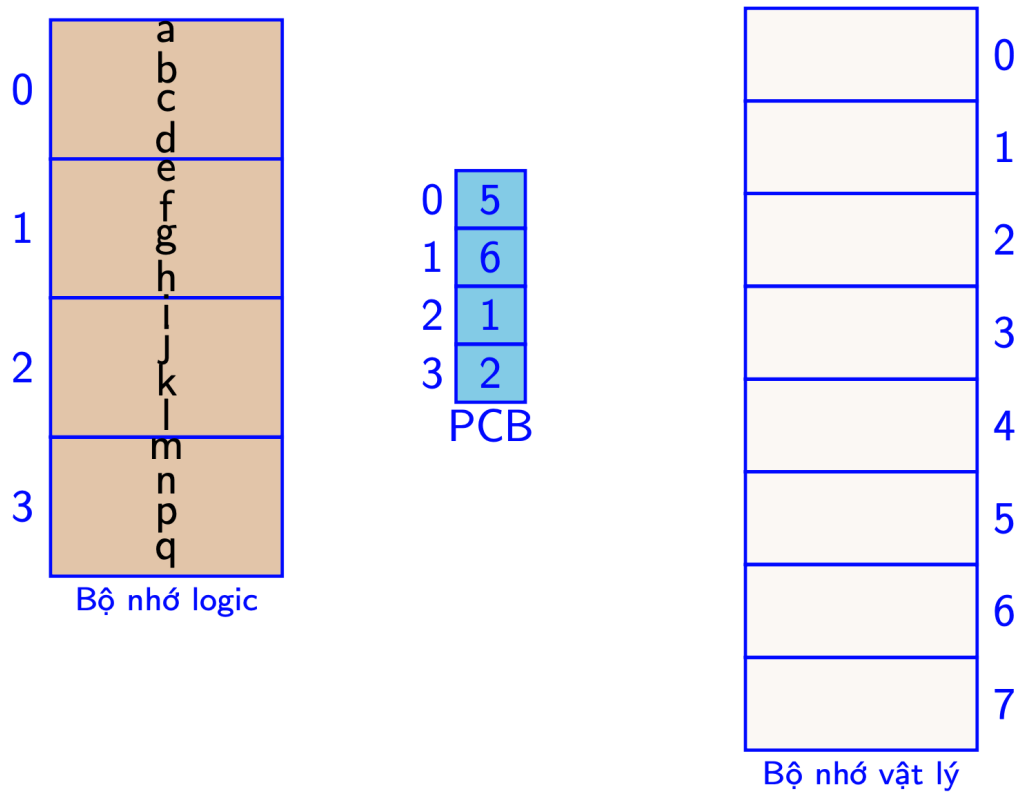
Ví dụ: Không gian nhớ cho các tiến trình A, B, C và D:

Số khung	Bộ nhớ
0	A.0
1	A.1
2	A.2
3	A.3
4	D.0
5	D.1
6	D.2
7	C.0
8	C.1
9	C.2
10	C.3
11	D.3
12	D.4
13	
14	

Bảng trang: HĐH quản lý việc cấp phát khung cho mỗi tiến trình bằng bảng trang (Page table): mỗi ô tương ứng với 1 trang và chứa số khung cấp cho trang đó.
Mỗi tiến trình có bảng trang riêng

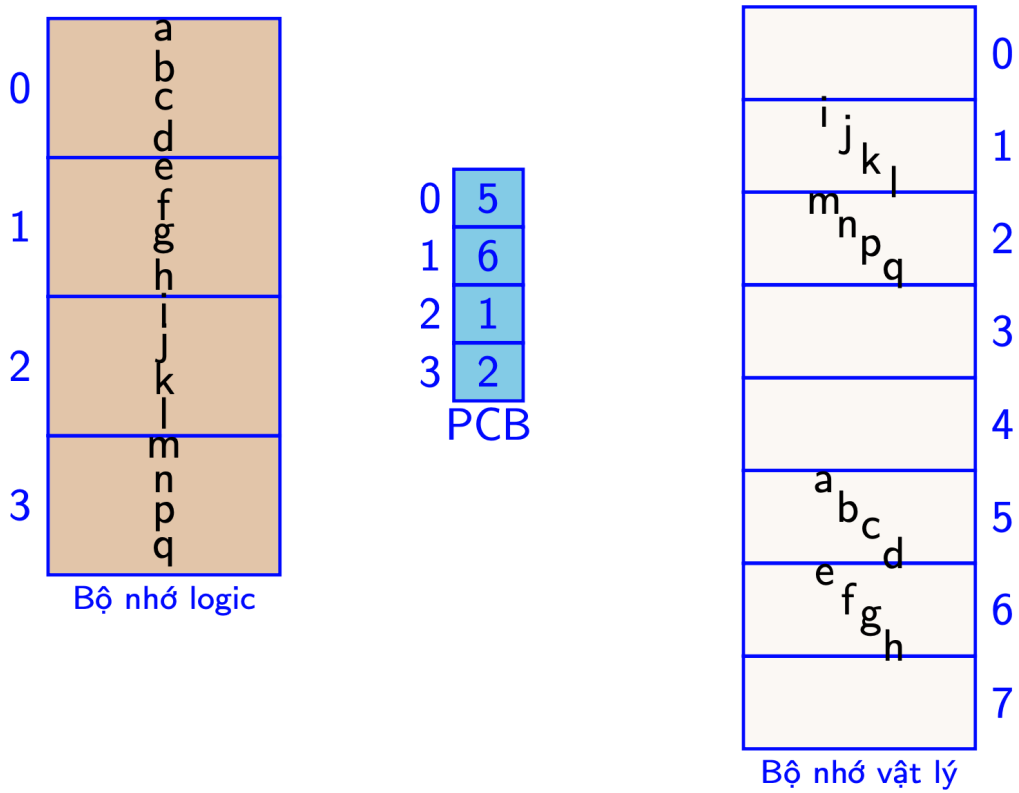


Ánh xạ địa chỉ địa chỉ logic khi phân trang sang địa chỉ vật lý.
Ví dụ: Địa chỉ của g trong trang 1 (Số thứ tự trang (p) : m = 1; Độ dịch trong trang (o): n = 2)

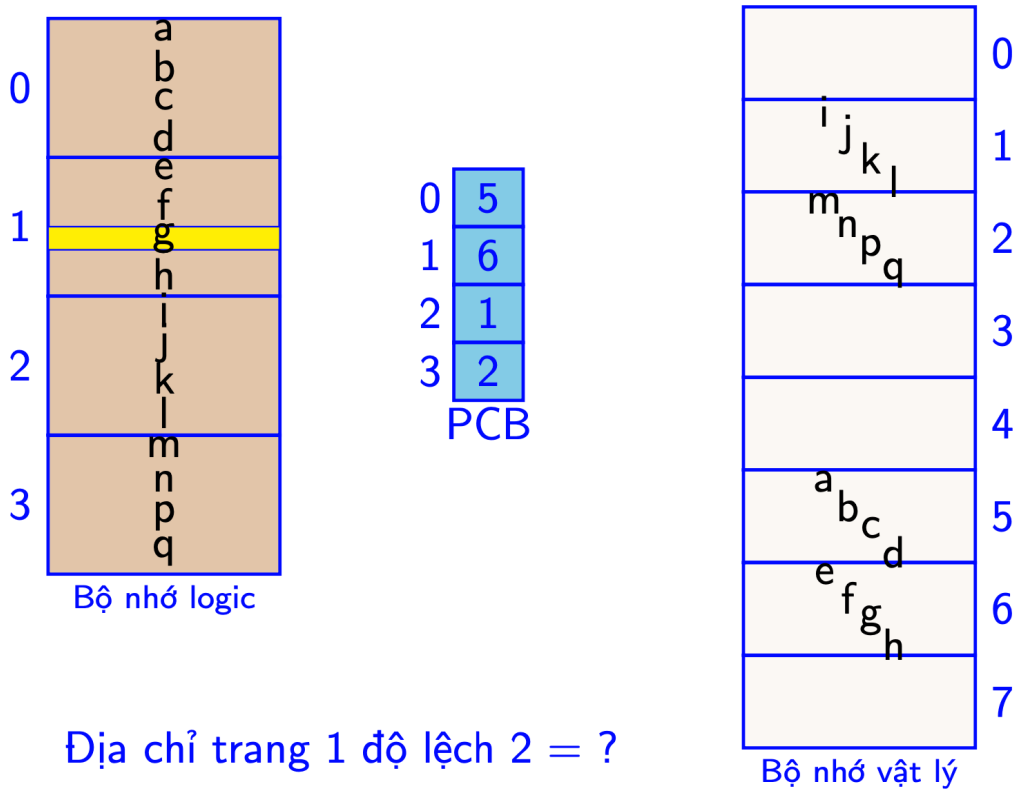


Ánh xạ địa chỉ ánh xạ địa chỉ logic khi phân trang sang địa chỉ vật lý.

Ví dụ: Địa chỉ của g trong trang 1 (Số thứ tự trang (p) : m = 1; Độ dịch trong trang (o): n = 2)

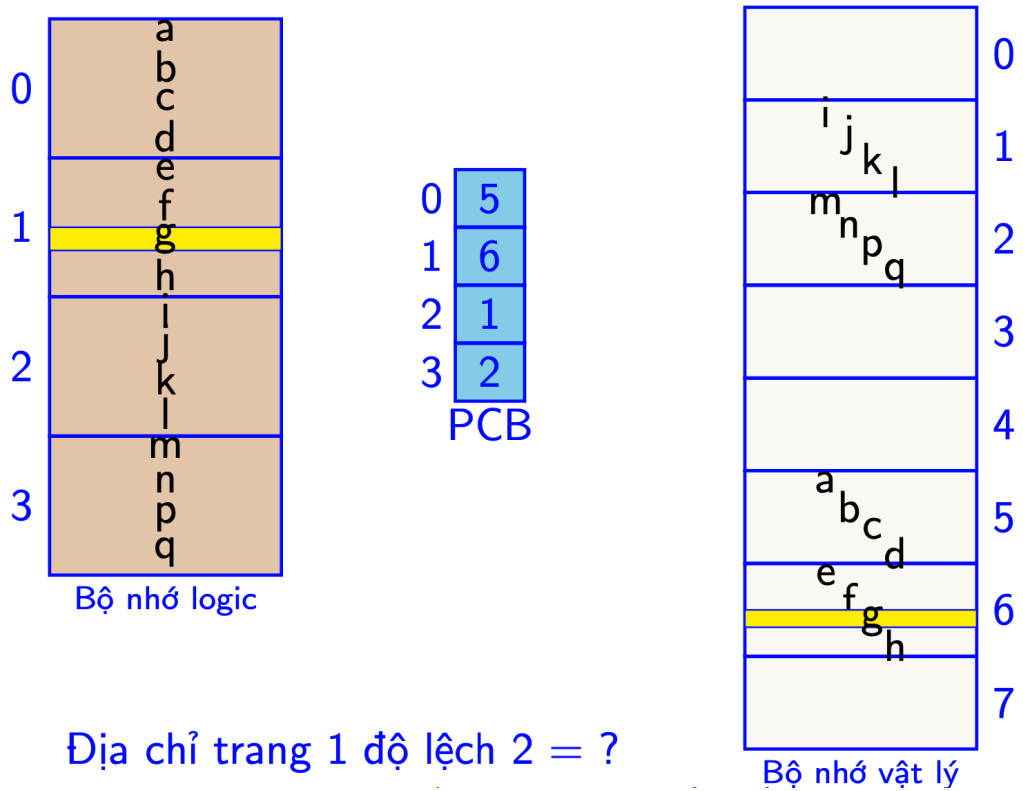


Ánh xạ địa chỉ địa chỉ logic khi phân trang sang địa chỉ vật lý.
Ví dụ: Địa chỉ của g trong trang 1 (Số thứ tự trang (p) : m = 1; Độ dịch trong trang (o): n = 2)



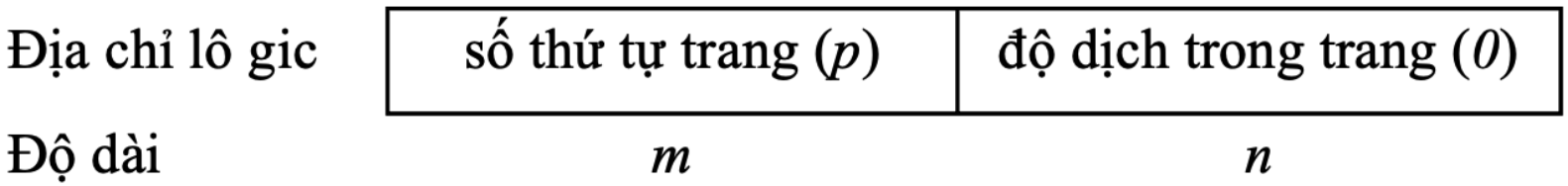
Địa chỉ trang 1 độ lệch 2 = ?

Ánh xạ địa chỉ địa chỉ logic khi phân trang sang địa chỉ vật lý.
Ví dụ: Địa chỉ của g trong trang 1 (Số thứ tự trang (p) : m = 1; Độ dịch trong trang (o): n = 2) = 6*4 + 2 = 26



Địa chỉ trang 1 độ lệch 2 = ?

- **Ánh xạ địa chỉ** gồm 2 phần:
  - Số thứ tự trang (p)
  - Độ dịch (địa chỉ lệch) của địa chỉ so với đầu trang (o)



- Nếu kích thước trang là 2^n. Biểu diễn địa chỉ logic dưới dạng địa chỉ có độ dài (m + n) bit
  - m bit cao: biểu diễn số thứ tự trang
  - n bit thấp: biểu diễn độ dịch trong trang nhớ

Ví dụ: Cho trang nhớ có kích thước 1024Byte, độ dài địa chỉ logic là 16 bit. Tính địa chỉ logic 1500: Tính số trang và độ dịch trong trang?

Ví dụ: Cho trang nhớ có kích thước 1024Byte, độ dài địa chỉ logic là 16 bit. Tính địa chỉ logic của địa chỉ vật lý 1500? Tính số trang và độ dịch trong trang?

Giải:

Cách 1:

m+n = 16 do không gian nhớ logic là 16 bit

Số lượng khung trong một trang là $1024 Bytes = 2^{10} Bytes$, nên cần 10 bit để biểu diễn cho khung trong trang

=> n = 10 ; m = 6

1500 = 000001*0111011100*

Vậy ta có $p = 000001_2 = 1; o = 0111011100_2 = 476$

Cách 2:

Phần $p = 1500/1024 = 1$

Phần $o = 1500 mod 1024 = 476$

Vậy ta có $p = 1; o = 476$

- Chuyển địa chỉ logic sang địa chỉ vật lý:
  - Lấy m bit cao của địa chỉ => được số thứ tự trang
  - Dựa vào bảng trang, tìm được số thứ tự khung vật lý (k)
  - Địa chỉ vật lý bắt đầu của khung là $k * 2^n$
  - **Địa chỉ vật lý của byte được tham chiếu = địa chỉ bắt đầu của khung + Địa chỉ lệch (độ dịch)**
- **Nhận xét** : Chỉ cần thêm số khung vào trước dãy bit biểu diễn độ lệch

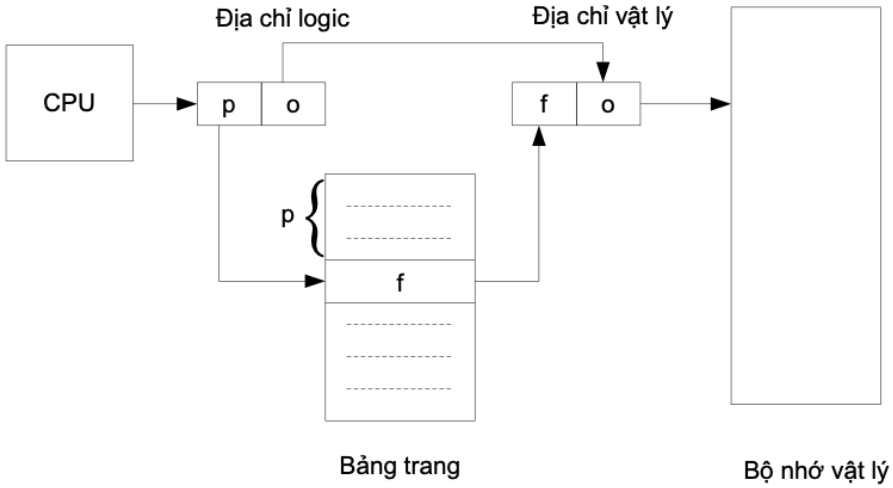
Ví dụ: Cho bảng trang như sau, với kích thước trang là 1KB. Hãy chuyển các địa chỉ logic sau sang địa chỉ vật lý:

- 1240
- 3580
- 1502

Page	Frame
0	3
1	2
2	6
3	4

- Hướng dẫn:
  - B1: Lấy (Địa chỉ logic) div (kích thước trang) = page => Xác định Frame
  - B2: Frame * (kích thước trang) + (địa chỉ logic) mod (kích thước trang) = địa chỉ vật lý

- Quá trình biến đổi từ địa chỉ logic sang địa chỉ vật lý được thực hiện bằng phần cứng
- Kích thước trang là lũy thừa của 2, nằm trong khoảng từ 512B đến 16MB
- Việc tách phần p và o trong địa chỉ logic được thực hiện dễ dàng bằng phép dịch bit



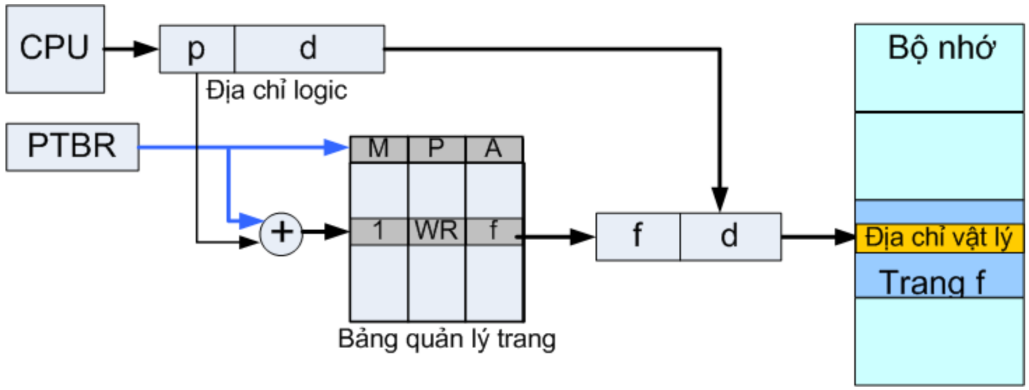
Ưu điểm của phân trang

- Không tồn tại hiện tượng phân đoạn ngoài: Phân mảnh trong khi phân trang có giá trị trung bình bằng nửa trang
- Hệ số song song cao và cho phép sử dụng chung trang
- Cơ chế ánh xạ địa chỉ hoàn toàn trong suốt đối với chương trình

Nhược điểm của phân trang

- Tồn tại hiện tượng phân đoạn trong
  - Luôn xuất hiện ở trang cuối cùng
  - Giảm kích thước trang cho phép tiết kiệm MEM?
    - Kích thước trang nhỏ => số lượng lớn, bảng trang to, khó quản lý
    - Không tiện cho việc trao đổi với đĩa do thực hiện theo từng khối lớn
- Đòi hỏi hỗ trợ của phần cứng cho chiến lược phân trang lớn
- Khi chương trình lớn, bảng quản lý trang nhiều phần tử

- Mỗi thao tác truy cập bộ nhớ đều đòi hỏi truy cập bảng phân trang => tổ chức bảng phân trang sao cho tốc độ truy cập là cao nhất



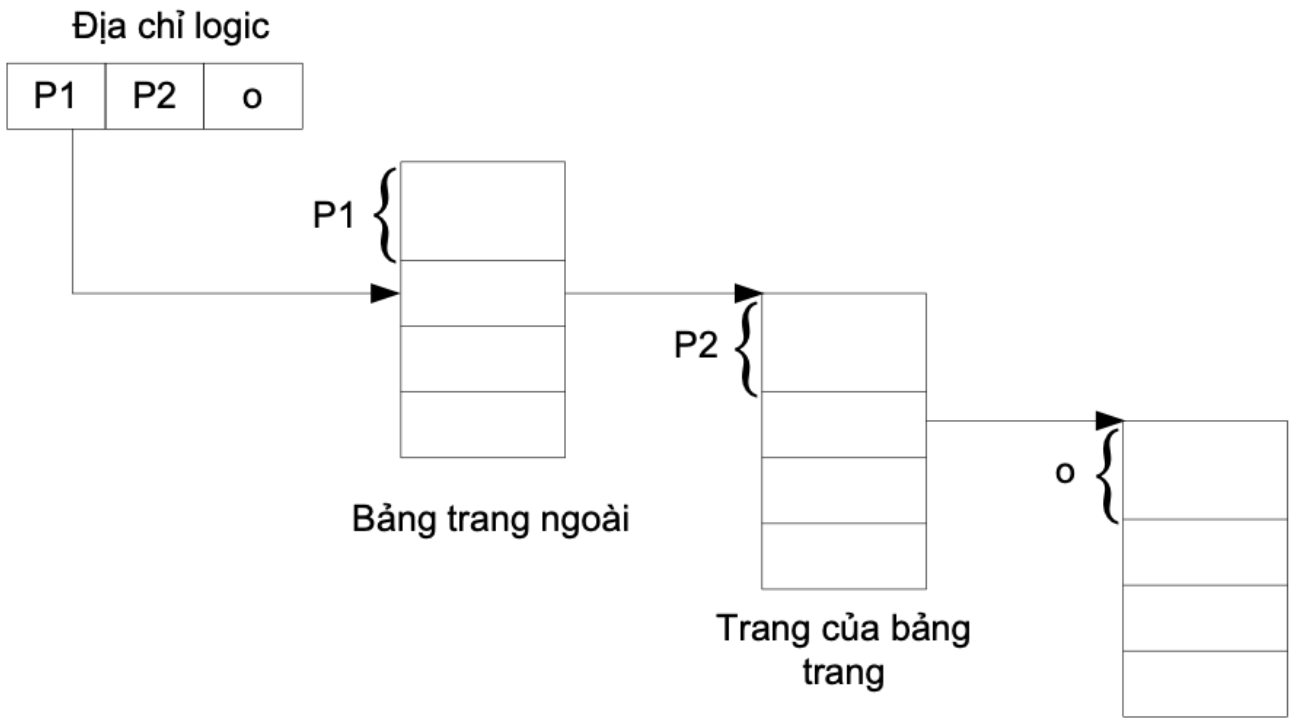
- Sử dụng tập hợp các thanh ghi làm bảng phân trang:
  - Tốc độ truy cập rất cao
  - Số lượng thanh ghi hạn chế => không áp dụng được
- Giữ các bảng trang trong MEM:
  - Vị trí mỗi bảng được trỏ bởi thanh ghi cơ sở bảng trang PTBR (Page Table Base Register)
  - Thời gian để truy cập bảng => sử dụng bộ nhớ cache tốc độ cao

Bảng trang nhiều mức:

- Các hệ thống tính toán hiện đại cho phép sử dụng không gian địa chỉ logic lớn ($2^{32} -> 2^{64}$) => Số lượng tragn cần quản lý tăng dẫn đến kích thước bảng trang tăng
- **Nguyên tắc**: *Bảng quản lý trang được phân trang*
  - Chia bảng trang thành những phần nhỏ hơn
  - Tổ chức bảng trang nhiều mức: Khoản mục của bảng mức trên chỉ tới bảng trang khác
- **Ví dụ về bảng 2 mức**:
  - Máy tính có không gian địa chỉ logic là 4GB ($2^{32}B$), kích thước trang nhớ là 4KB ($2^{12}B$):
    - Số thứ tự trang $2^{32}/2^{12} = 2^{20} => m = 20bit$
    - Độ dịch trong trang $2^{12} => n = 12bit$
- Bảng trang được phân trang. Số hiệu trang được chia thành 2 phần:

- P1: 10 bit cho phép định vị khoản mục trong bảng mức trên
- P2: 10 bit định vị khoản mục trong bảng mức dưới
- O: 12 bit, chứa độ dịch trong trang

- Địa chỉ truy nhập có dạng P_1, P_2, o



Tổng kết

Chương 3 Quản lý bộ nhớ

- Phân chương bộ nhớ
- Phân trang bộ nhớ

Chương 3 Quản lý bộ nhớ

- Phân đoạn bộ nhớ
- Bộ nhớ ảo

Câu hỏi và bài tập

1. **Câu 1**: Cho bộ nhớ có kích thước 1MB. Sử dụng phương pháp kề cận để cấp phát cho các tiến trình lần lượt với kích thước như sau : A : 120KB, B : 210KB, C : 150KB, D : 40KB.
2. **Câu 2**: Không gian nhớ logic gồm 512 trang, mỗi trang có kích thước 1024B. Bộ nhớ vật lý gồm 64 khung. Cho biết địa chỉ logic dài bao nhiêu bit, phần số thứ tự trang và phần độ dịch trong trang có độ dài lần lượt là bao nhiêu bit. Địa chỉ vật lý dài bao nhiêu bit.
3. **Câu 3**: Hệ thống phân trang, kích thước trang 1024B. Bảng phân trang hiện thời như hình, hãy tính địa chỉ vật lý cho các địa chỉ lôgic:

3	0
2	4
1	
0	1

   3.1 1020

   3.2 2060