

1

# QUẢN LÝ BỘ NHỚ

# QUẢN LÝ BỘ NHỚ

2

- ❑ Quản lý bộ nhớ là một trong những nhiệm vụ quan trọng và phức tạp nhất của hệ điều hành.
- ❑ Bộ phận quản lý bộ nhớ xem bộ nhớ chính như là một tài nguyên của hệ thống dùng để cấp phát và chia sẻ cho nhiều tiến trình đang ở trong trạng thái active.
- ❑ Các hệ điều hành đều mong muốn có nhiều các tiến trình trên bộ nhớ chính. Công cụ cơ bản của quản lý bộ nhớ là sự phân trang (paging) và sự phân đoạn (segmentation)

# Nhiệm vụ của quản lý bộ nhớ

3

- ❑ Trong các hệ thống đơn chương trình (uniprogramming), trên bộ nhớ chính ngoài hệ điều hành, chỉ có một chương trình đang thực hiện
- ❑ Trong các hệ thống đa chương (multiprogramming) trên bộ nhớ chính ngoài hệ điều hành, có thể có nhiều tiến trình đang hoạt động.
  - Trong hệ thống đa chương bộ phận quản lý bộ nhớ phải có nhiệm vụ đưa bất kỳ một tiến trình nào đó vào bộ nhớ khi nó có yêu cầu, kể cả khi trên bộ nhớ không còn không gian trống. Nó phải bảo vệ chính hệ điều hành và các tiến trình trên bộ nhớ tránh các trường hợp truy xuất bất hợp lệ xảy ra.

# Nhiệm vụ của quản lý bộ nhớ

4

*Bộ phận quản lý bộ nhớ phải thực hiện các nhiệm vụ sau đây:*

- **Sự tái định vị (Relocation):** Trong các hệ thống đa chương, không gian bộ nhớ chính thường được chia sẻ cho nhiều tiến trình khác nhau và yêu cầu bộ nhớ của các tiến trình luôn lớn hơn không gian bộ nhớ vật lý mà hệ thống có được.
- **Bảo vệ bộ nhớ (Protection):** Mỗi tiến trình phải được bảo vệ để chống lại sự truy xuất bất hợp lệ vô tình hay có chủ ý của các tiến trình khác.

# Nhiệm vụ của quản lý bộ nhớ

5

- **Chia sẻ bộ nhớ (Sharing):** Bất kỳ một chiến lược nào được cài đặt đều phải có tính mềm dẻo để cho phép nhiều tiến trình có thể truy cập đến cùng một địa chỉ trên bộ nhớ chính.  
VD: khi có nhiều tiến trình cùng thực hiện một chương trình thì việc cho phép mỗi tiến trình cùng truy cập đến một bản copy của chương trình sẽ thuận lợi hơn khi cho phép mỗi tiến trình truy cập đến một bản copy sở hữu riêng.

# Nhiệm vụ của quản lý bộ nhớ

6

- **Tổ chức bộ nhớ logic** (Logical organization): Bộ nhớ chính của hệ thống máy tính được tổ chức như là một dòng hoặc một mảng, không gian địa chỉ bao gồm một dãy có thứ tự các byte hoặc các word.
- **Tổ chức bộ nhớ vật lý** (Physical organization): Như chúng ta đã biết bộ nhớ máy tính được tổ chức theo 2 cấp:
  - Bộ nhớ chính cung cấp một tốc độ truy cập dữ liệu cao.
  - Bộ nhớ phụ có tốc độ truy xuất chậm và rẻ tiền hơn so với bộ nhớ chính.

# Kỹ thuật cấp phát bộ nhớ ( nạp chương trình vào bộ nhớ chính)

7

1. **Kỹ thuật phân vùng cố định (Fixed Partitioning)**
2. **Kỹ thuật phân vùng động (Dynamic Partitioning)**

# Kỹ thuật cấp phát bộ nhớ ( nạp chương trình vào bộ nhớ chính)

8

## 1. Kỹ thuật phân vùng cố định (Fixed Partitioning)

Trong kỹ thuật này không gian địa chỉ của bộ nhớ chính được chia thành 2 phần cố định, phần nằm ở vùng địa chỉ thấp dùng để chứa chính hệ điều hành, phần còn lại, tạm gọi là phần user program, là sẵn sàng cho việc sử dụng của các tiến trình khi các tiến trình được nạp vào bộ nhớ chính



# Kỹ thuật cấp phát bộ nhớ ( nạp chương trình vào bộ nhớ chính)

9

Có hai trở ngại trong việc sử dụng các phân vùng cố định với kích thước bằng nhau:

- Thứ nhất: khi kích thước của một chương trình là quá lớn so với kích thước của một partition thì người lập trình phải thiết kế chương trình theo cấu trúc overlay, theo đó chỉ những phần chia cần thiết của chương trình mới được nạp vào bộ nhớ chính khi khởi tạo chương trình, sau đó người lập trình phải nạp tiếp các modun cần thiết khác vào đúng partition của chương trình và sẽ ghi đè lên bất kỳ chương trình hoặc dữ liệu ở trong đó. Cấu trúc chương trình overlay tiết kiệm được bộ nhớ nhưng yêu cầu cao ở người lập trình

# Kỹ thuật cấp phát bộ nhớ (nạp chương trình vào bộ nhớ chính)

10

- Thứ hai: khi kích thước của một chương trình nhỏ hơn kích thước của một partition hoặc quá lớn so với kích thước của một partition nhưng không phải là bội số của kích thước một partition thì dễ xảy ra hiện tượng phân mảnh bên trong (internal fragmentation) bộ nhớ, gây lãng phí bộ nhớ.

Ví dụ: nếu có 3 không gian trống kích thước 30K nằm rải rác trên bộ nhớ, thì cũng sẽ không nạp được một modul chương trình có kích thước 12K, hiện tượng này được gọi là hiện tượng phân mảnh bên trong.

→ Cả hai vấn đề trên có thể được khắc phục bằng cách sử dụng các phân vùng có kích thước không bằng nhau.

# Kỹ thuật cấp phát bộ nhớ ( nạp chương trình vào bộ nhớ chính)

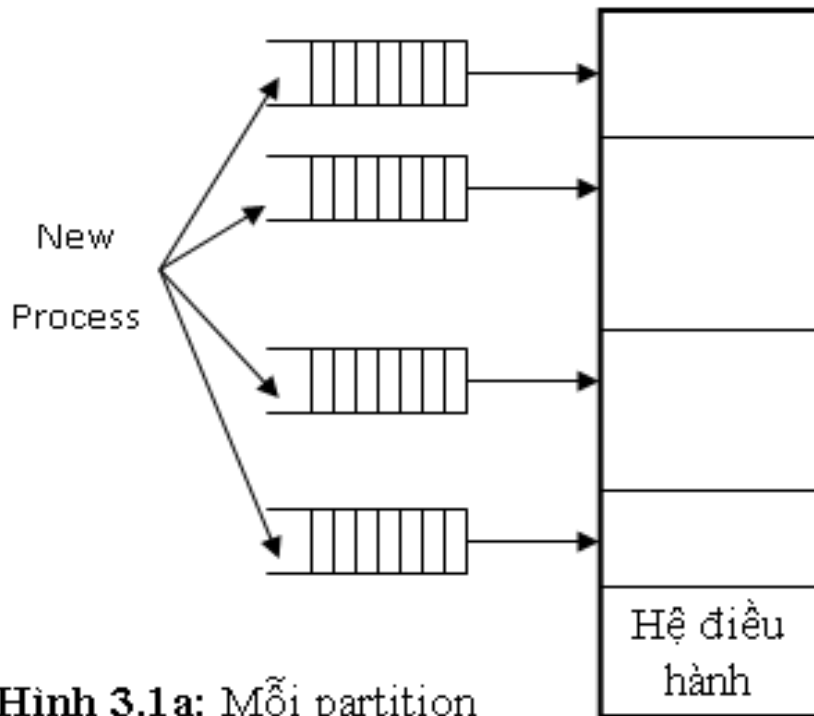
11

Với các partition có kích thước không bằng nhau thì có hai cách để lựa chọn khi đưa một tiến trình vào partition:

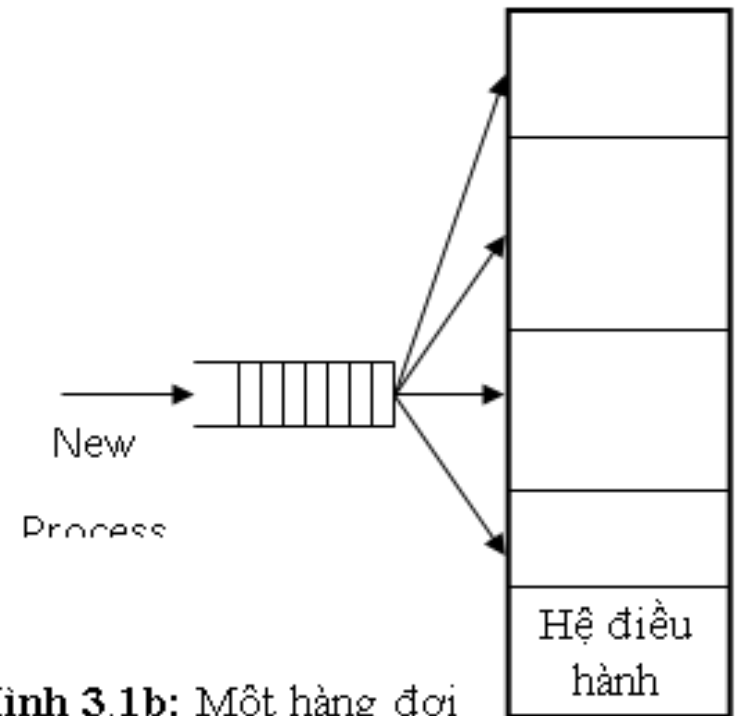
1. Mỗi phân vùng có một hàng đợi tương ứng, theo đó mỗi tiến trình khi cần được nạp vào bộ nhớ nó sẽ được đưa đến hàng đợi của phân vùng có kích thước vừa đủ để chứa nó, để vào/để đợi được vào phân vùng
2. Hệ thống dùng một hàng đợi chung cho tất cả các phân vùng, theo đó tất cả các tiến trình muốn được nạp vào phân vùng nhưng chưa được vào sẽ được đưa vào hàng đợi chung này

# Kỹ thuật cấp phát bộ nhớ ( nạp chương trình vào bộ nhớ chính)

12



**Hình 3.1a:** Mỗi partition có một hàng đợi riêng



**Hình 3.1b:** Một hàng đợi chung cho tất cả partition

→ Sự phân vùng cố định ít được sử dụng trong các hệ điều hành hiện nay.

# Kỹ thuật cấp phát bộ nhớ ( nạp chương trình vào bộ nhớ chính)

13

## 2. Kỹ thuật phân vùng động (Dynamic Partitioning)

Để khắc phục một vài hạn chế của kỹ thuật phân vùng cố định, kỹ thuật phân vùng động ra đời.

- Trong kỹ thuật phân vùng động, số lượng các phân vùng trên bộ nhớ và kích thước của mỗi phân vùng là có thể thay đổi.
- Khi có một tiến trình được nạp vào bộ nhớ nó được hệ điều hành cấp cho nó không gian vừa đủ để chứa tiến trình, phần còn lại để sẵn sàng cấp cho tiến trình khác sau này.

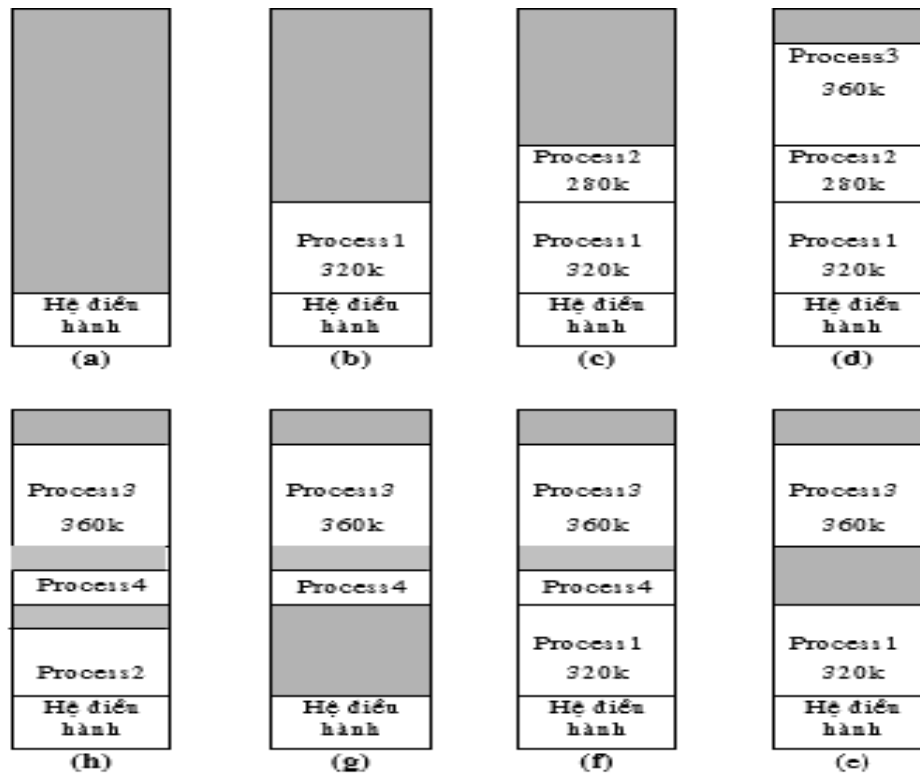
# Kỹ thuật cấp phát bộ nhớ ( nạp chương trình vào bộ nhớ chính)

14

- Khi một tiến trình kết thúc nó được đưa ra ngoài và phần không gian bộ nhớ mà tiến trình này trả lại cho hệ điều hành sẽ được hệ điều hành cấp cho tiến trình khác

# Kỹ thuật cấp phát bộ nhớ ( nạp chương trình vào bộ nhớ chính)

15

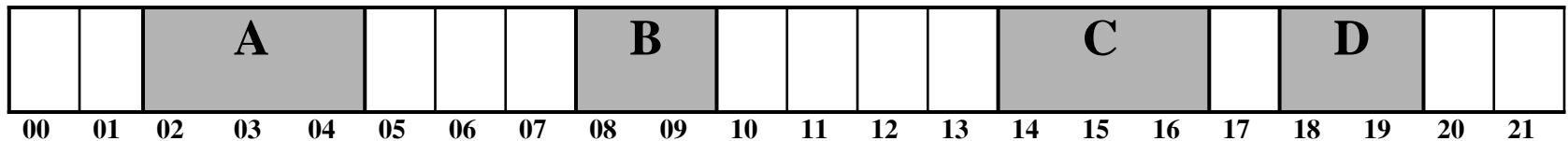


trên đây minh họa cho quá trình nạp/kết thúc các tiến trình theo thứ tự: nạp process1, nạp process2, nạp process3, kết thúc process2, nạp process4, kết thúc process1, nạp process2 vào lại, trong hệ thống phân vùng động

# Kỹ thuật cấp phát bộ nhớ ( nạp chương trình vào bộ nhớ chính)

16

Hệ điều hành sử dụng 2 cơ chế: Bản đồ bit và Danh sách liên kết. Trong cả 2 cơ chế này hệ điều hành đều chia không gian nhớ thành các đơn vị cấp phát có kích thước bằng nhau, các đơn vị cấp phát liên tiếp nhau tạo thành một khối nhớ (block), hệ điều hành cấp phát các block này cho các tiến trình khi nạp tiến trình vào bộ nhớ.



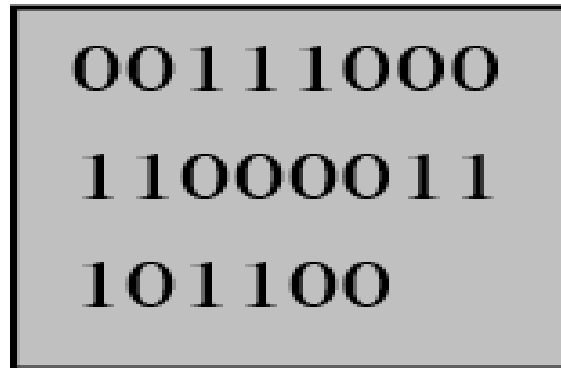
Một đoạn nhớ bao gồm 22 đơn vị cấp phát, tạo thành 9 block, trong đó có 4 block đã cấp phát (tô đậm, kí hiệu là P) cho các tiến trình: A, B, C, D và 5 block chưa được cấp phát (để trắng, kí hiệu là H)



# Kỹ thuật cấp phát bộ nhớ ( nạp chương trình vào bộ nhớ chính)

17

- Trong cơ chế bản đồ بیت: mỗi đơn vị cấp phát được đại diện bởi một بیت trong bản đồ بیت. Đơn vị cấp phát còn trống được đại diện bằng بیت 0, ngược lại đơn vị cấp phát được đại diện bằng بیت 1

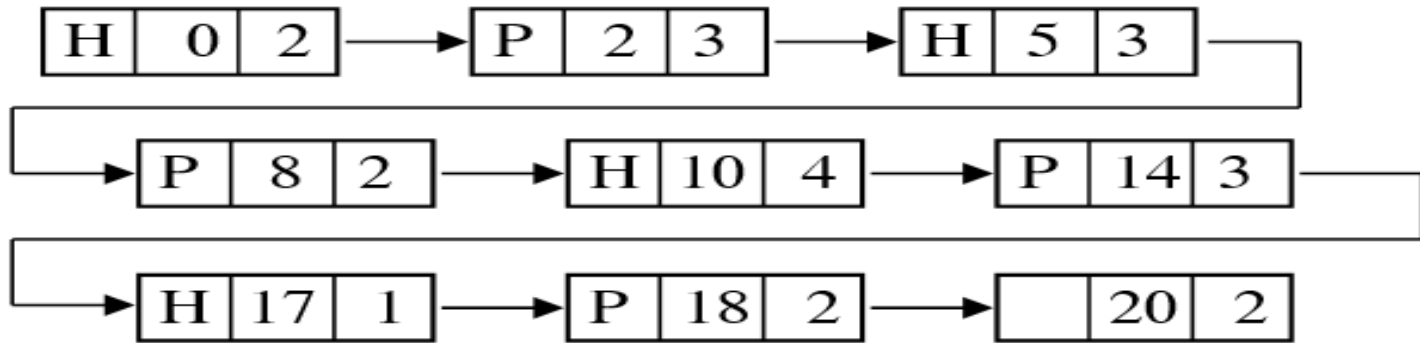


```
00111000
11000011
101100
```

quản lý các đơn vị cấp phát bằng bản đồ بیت.

# Kỹ thuật cấp phát bộ nhớ ( nạp chương trình vào bộ nhớ chính)

18



Trong cơ chế danh sách liên kết: Mỗi block trên bộ nhớ được đại diện bởi một phần tử trong danh sách liên kết, mỗi phần tử này gồm có 3 trường chính: trường thứ nhất cho biết khối nhớ đã cấp phát (P: process) hay đang còn trống (H: Hole), trường thứ hai cho biết thứ tự của đơn vị cấp phát đầu tiên trong block, trường thứ ba cho biết block gồm bao nhiêu đơn vị cấp phát

→ Như vậy khi cần nạp một tiến trình vào bộ nhớ thì hệ điều hành phải dựa vào bản đồ bit hoặc danh sách liên kết để tìm ra một block có kích thước đủ để nạp tiến trình.

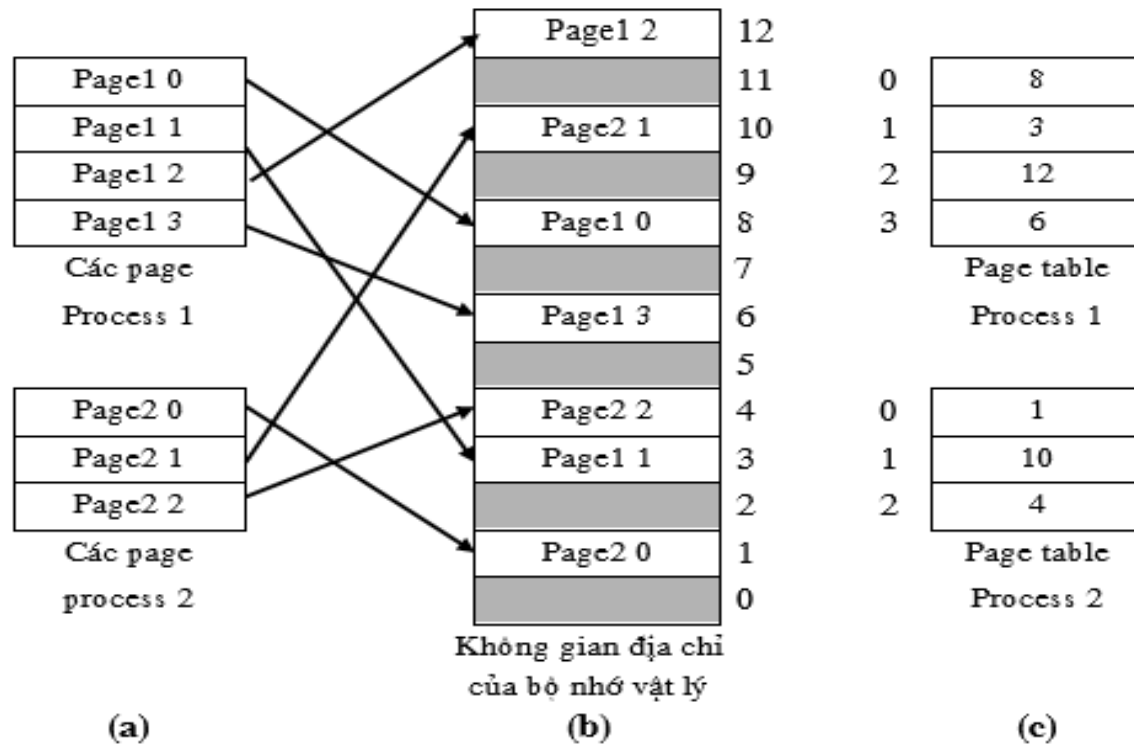
# Kỹ thuật phân trang đơn (Simple Paging)

19

Trong kỹ thuật này không gian địa chỉ bộ nhớ vật lý được chia thành các phần có kích thước cố định bằng nhau, được đánh số địa chỉ bắt đầu từ 0 và được gọi là các khung trang (page frame). Không gian địa chỉ của các tiến trình cũng được chia thành các phần có kích thước bằng nhau và bằng kích thước của một khung trang, được gọi là các trang (page) của tiến trình

# Kỹ thuật phân trang đơn (Simple Paging)

20



Các trang của 2 tiến trình process 1 và process 2 (a), được nạp vào bộ nhớ (b), và 2 bảng trang tương ứng của nó (c).

# Kỹ thuật phân trang đơn (Simple Paging)

21

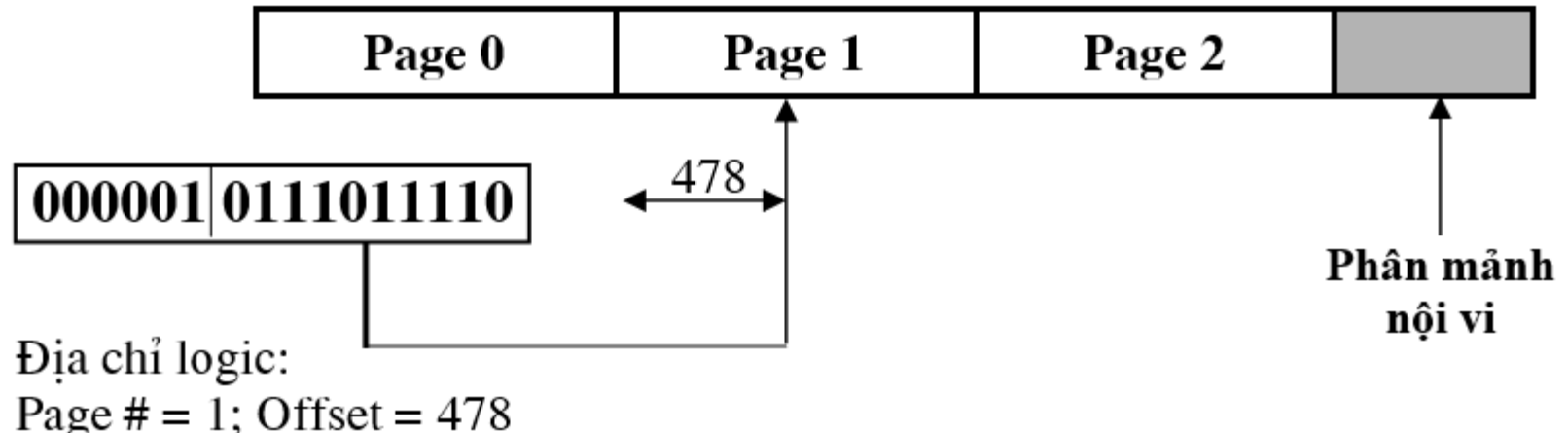
- ❑ Kích thước của mỗi trang hay khung trang do phần cứng quy định và thường là lũy thừa của 2, biến đổi từ 512 byte đến 8192 byte. Nếu kích thước của không gian địa chỉ là  $2^m$  và kích thước của trang là  $2^n$  thì  $m-n$  bit của địa chỉ logic là số hiệu trang (page) và  $n$  bit còn lại là địa chỉ tương đối trong trang (offset).

# Kỹ thuật phân trang đơn (Simple Paging)

22

Ví dụ: nếu địa chỉ logic gồm 16 bit, kích thước của mỗi trang là  $1K = 1024\text{byte}$  ( $2^{10}$ ), thì có 6 bit dành cho số hiệu trang, như vậy một chương trình có thể có tối đa  $2^6 = 64$  trang mỗi trang 1KB. Trong trường hợp này nếu CPU phát ra một giá trị địa chỉ 16 bit là:  $0000010111011110 = 1502$ , thì thành phần số hiệu trang là  $000001 = 1$ , thành phần offset là  $0111011110 = 478$ .

Hình minh hoạ:



# Kỹ thuật phân trang đơn (Simple Paging)

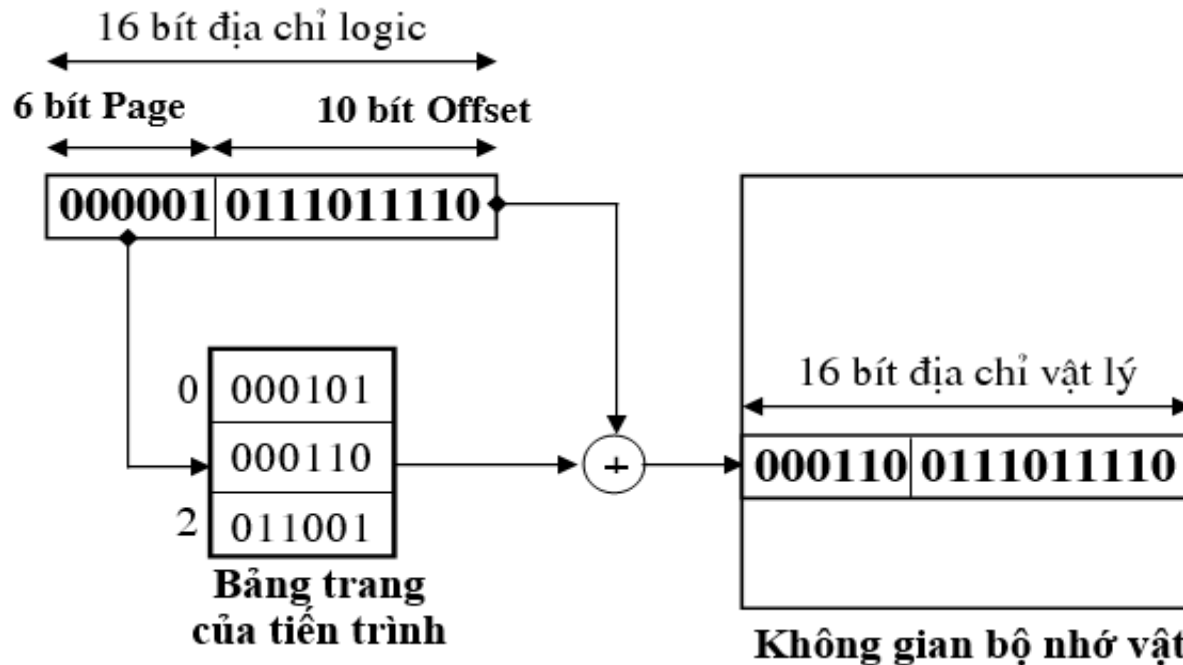
23

Việc chuyển từ địa chỉ logic sang địa chỉ vật lý được thực hiện theo các bước sau:

- Trích ra m-n bit trái nhất (thấp nhất) của địa chỉ logic để xác định số hiệu trang cần truy xuất.
- Sử dụng số hiệu trang ở trên để chỉ đến phần tử tương ứng trong bảng trang của tiến trình, để xác định khung trang tương ứng, ví dụ là k.
- Địa chỉ vật lý bắt đầu của khung trang là  $k \times 2^n$ , và địa chỉ vật lý của byte cần truy xuất là số hiệu trang cộng với giá trị offset. Địa chỉ vật lý không cần tính toán, nó dễ dàng có được bằng cách nối số hiệu khung trang với giá trị offset.

# Sơ đồ chuyển đổi địa chỉ logic (page) – vật lý

24



Trong sơ đồ ví dụ ở trên, chúng ta có địa chỉ logic là: 0000010111011110, với số hiệu trang là 1, offset là 478, giả định rằng trang này thường trú trong bộ nhớ chính tại khung trang 6 = 000110. Thì địa chỉ vật lý là khung trang số 6 và offset là 478 = 0001100111011110.



# Nhận xét về kỹ thuật phân trang:

25

- Có thể thấy sự phân trang được mô tả ở đây tương tự như sự phân vùng cố định. Sự khác nhau là với phân trang các phân vùng có kích thước nhỏ hơn, một chương trình có thể chiếm giữa nhiều hơn một phân vùng, và các phân vùng này có thể không liền kề với nhau.
- Kỹ thuật phân trang loại bỏ được hiện tượng phân mảnh ngoại vi.
- Khi cần truy xuất đến dữ liệu hay chỉ thị trên bộ nhớ thì hệ thống phải cần một lần truy xuất đến bảng trang

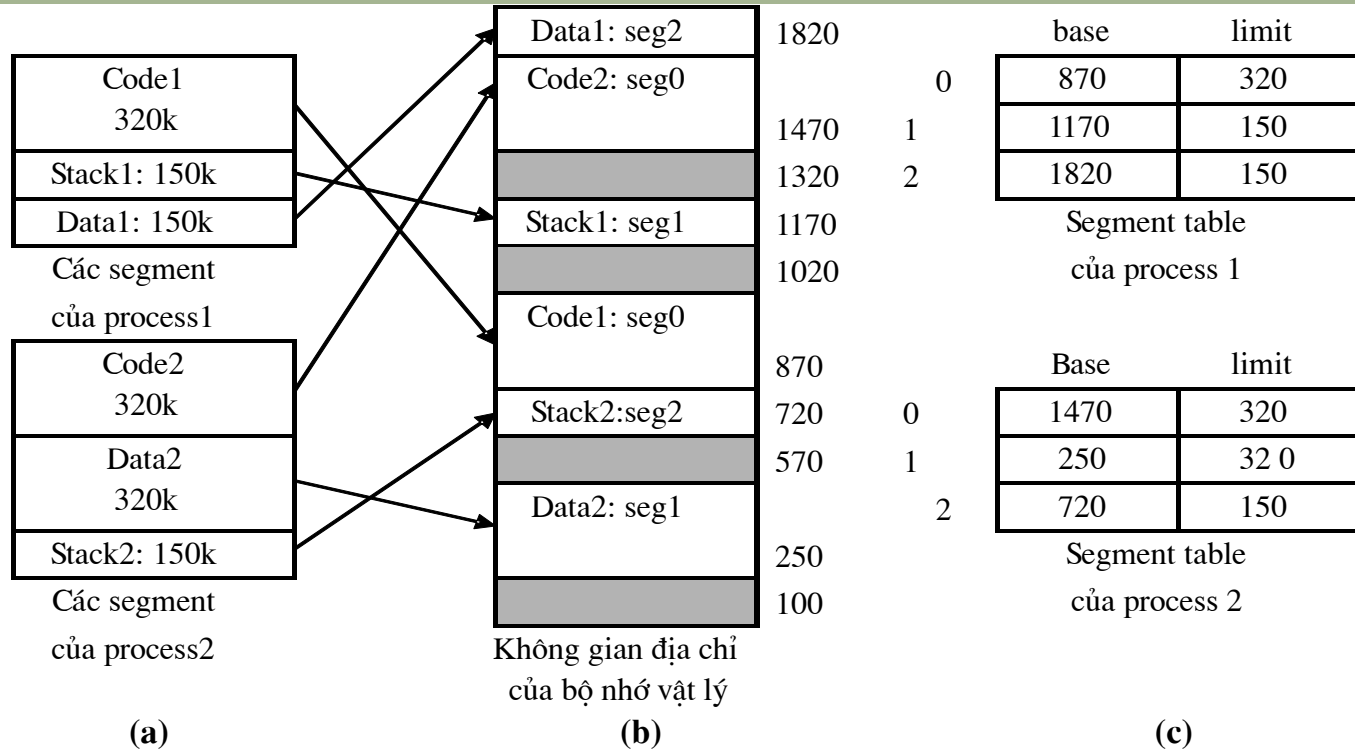
# Kỹ thuật phân đoạn đơn (Simple Segmentation)

26

- ❑ Trong kỹ thuật này không gian địa chỉ bộ nhớ vật lý được chia thành các phần cố định có kích thước không bằng nhau, được đánh số bắt đầu từ 0, được gọi là các phân đoạn (segment)
- ❑ Khi một tiến trình được nạp vào bộ nhớ thì tất cả các đoạn của nó sẽ được nạp vào các phân đoạn còn trống khác nhau trên bộ nhớ. Các phân đoạn này có thể không liên tiếp nhau. Xem hình

# Kỹ thuật phân đoạn đơn (Simple Segmentation)

27



Các đoạn của 2 tiến trình process 1 và process 2 (a), được nạp vào bộ nhớ (b), và 2 bảng đoạn tương ứng của nó (c).

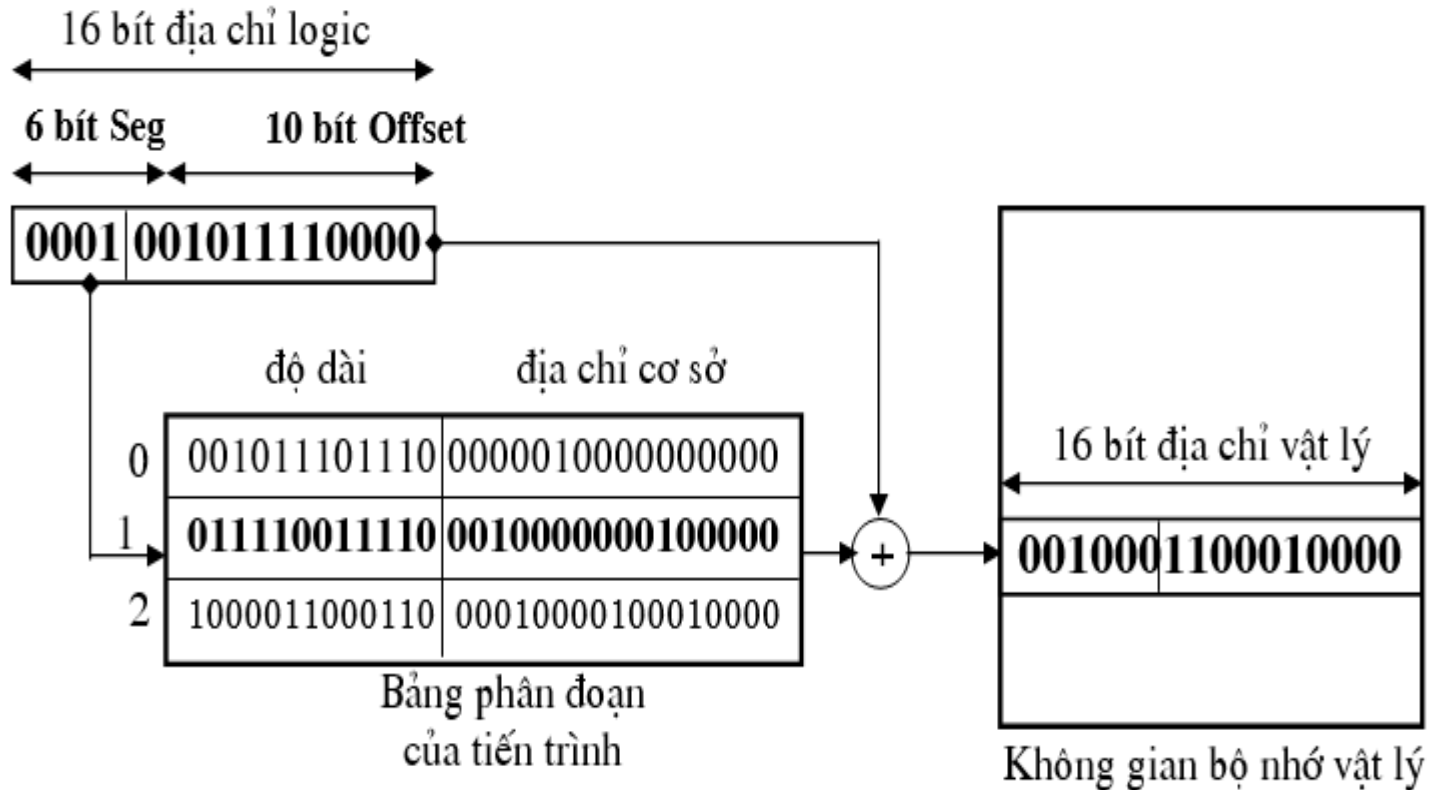
# Kỹ thuật phân đoạn đơn (Simple Segmentation)

28

Trong sơ đồ ví dụ sau đây, ta có địa chỉ logic là: 0001001011110000, với số hiệu segment là 1, offset là 752, giả định segment này thường trú trong bộ nhớ chính tại địa chỉ vật lý là 0010000000100000, thì địa chỉ vật lý tương ứng với địa chỉ logic ở trên là:  $0010000000100000 + 001011110000 = 0010001100010000$ .

# Sơ đồ chuyển đổi địa chỉ logic (segment) – vật lý

29



# Nhận xét về kỹ thuật phân đoạn

30

- ❑ Vì các segment có kích thước không bằng nhau nên sự phân đoạn tương tự như sự phân vùng động. Sự khác nhau là với sự phân đoạn một chương trình có thể chiếm giữ hơn một phân vùng, và các phân vùng này có thể không liền kề với nhau. Sự phân vùng loại trừ được sự phân mảnh nội vi, nhưng như sự phân vùng động nó vẫn xuất hiện hiện tượng phân mảnh ngoại vi
  - ❑ Kỹ thuật phân đoạn thể hiện được cấu trúc logic của chương trình, nhưng nó phải cấp phát các khối nhớ có kích thước khác nhau cho các phân đoạn của chương trình trên bộ nhớ vật lý, điều này phức tạp hơn nhiều so với việc cấp phát các khung trang
- Để dung hòa vấn đề này các hệ điều hành có thể kết hợp cả phân trang và phân đoạn.



# Bộ nhớ ảo

32

## ❑ Nhận xét

- Code của chương trình cần nạp vào bộ nhớ khi thực thi
  - Không phải tất cả các phần của một chương trình cần phải nạp vào bộ nhớ tại cùng một thời điểm
    - có những phần của chương trình ít khi sử dụng . VD: những xử lý lỗi ít khi xảy ra, những tính năng ít dùng , ...
    - Ngay cả khi toàn bộ chương trình đều cần dùng, thì không cần dùng toàn bộ cùng một lúc
- ⇒ **khả năng thực thi chương trình mà không cần nạp toàn bộ vào bộ nhớ**

Tham khảo :

[http://www.cse.hcmut.edu.vn/~sonsys/OS\\_CQA01/Lecture11.pdf](http://www.cse.hcmut.edu.vn/~sonsys/OS_CQA01/Lecture11.pdf)



# Bộ nhớ ảo

33

## ❑ Virtual memory:

- Cho phép nạp một phần chương trình cần thiết vào bộ nhớ để thực thi
- Tách rời bộ nhớ logical với bộ nhớ physical

## ❑ Ưu điểm :

- Kích thước của chương trình ( logical address space) có thể lớn hơn nhiều so với kích thước của bộ nhớ cấp phát cho chương trình (physical address space)
- Có nhiều chương trình chạy đồng thời hơn
- ...

# Bộ nhớ ảo

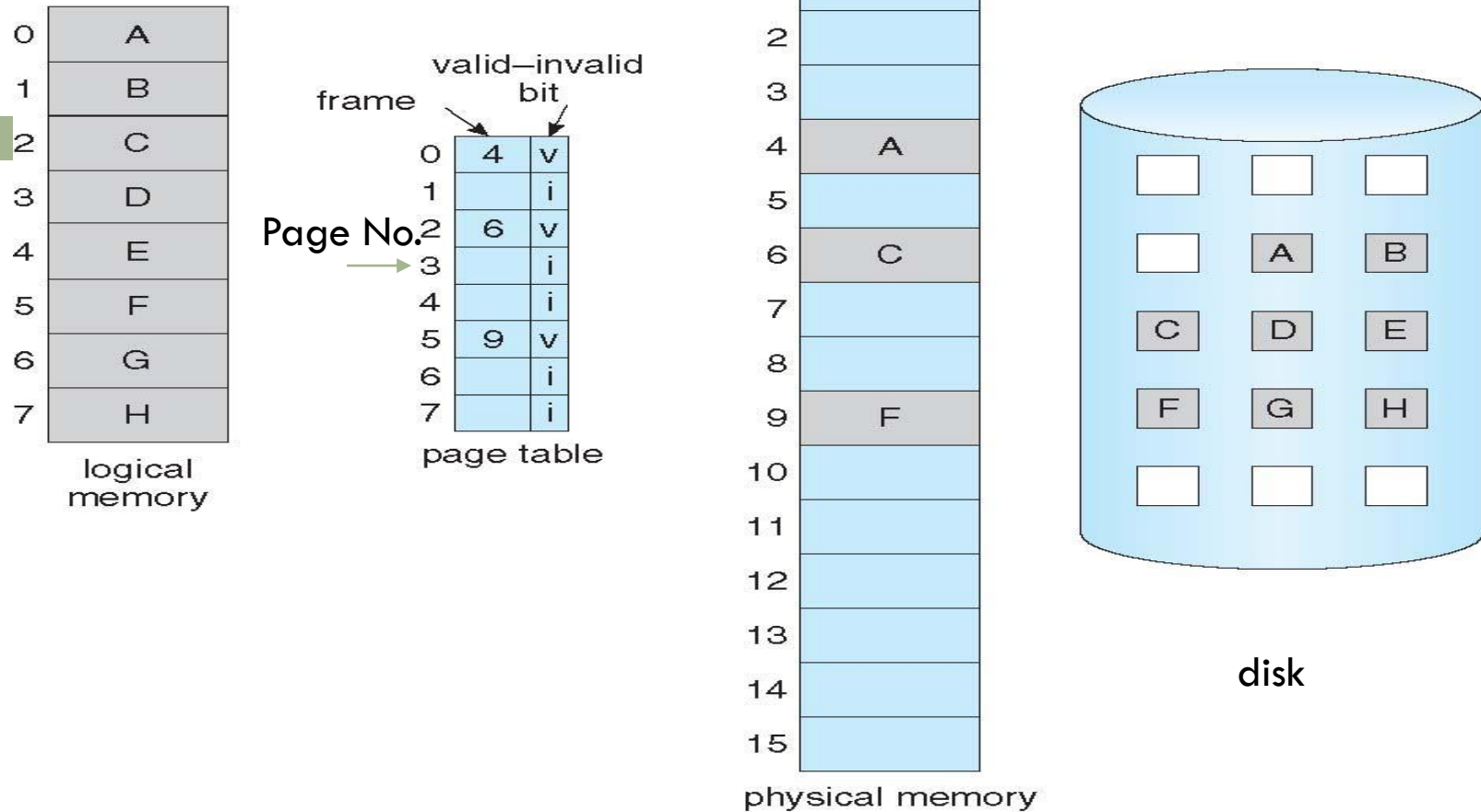
34

- ❑ **Virtual memory** : có 2 cơ chế hiện thực bộ nhớ ảo
  - Demand paging (*tìm hiểu cơ chế này*)
  - Demand segmentation

# Bộ nhớ ảo

35

- ❑ **Bộ nhớ ảo với cơ chế phân trang (Demand paging)**
  - Phân chia Không gian địa chỉ logic thành các page
  - Dùng bộ nhớ phụ (disk) để mở rộng bộ nhớ chính, lưu trữ các phần của process chưa được nạp (swap space)
  - Bổ sung bit cờ hiệu trong Page Table để nhận dạng tình trạng của một page đã được nạp vào bộ nhớ chính chưa
  - Cơ chế chuyển đổi giữa Bộ nhớ chính và Bộ nhớ phụ : **swapping**



Physical memory : chia thành các frame

Logical memory : chia thành các page

Page table : page nào được đặt vào frame nào => valid bit bật

Nếu page chưa được đặt vào frame => invalid bit bật

# Bộ nhớ ảo

37

## ❑ Hỗ trợ của OS

- ❑ **Page-replacement algorithm** : Chọn frame của process sẽ được thay thế trang nhớ
  - Mục tiêu: số lượng page fault nhỏ nhất
  - Được đánh giá bằng cách thực thi giải thuật đối với một chuỗi tham chiếu bộ nhớ (memory reference string) và xác định số lần xảy ra page fault
- **Frame-allocation algorithm** : Cấp phát cho process bao nhiêu frame?

# Bộ nhớ ảo

38

## ❑ Hỗ trợ của OS

- **Frame-allocation algorithm** : OS phải quyết định cấp cho mỗi process bao nhiêu frame.
  - Cấp ít frame => nhiều page fault
  - Cấp nhiều frame => giảm mức độ multiprogramming

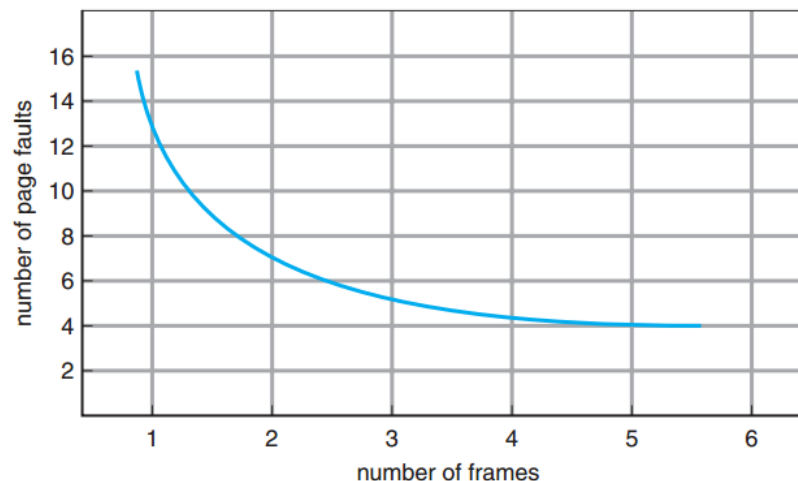
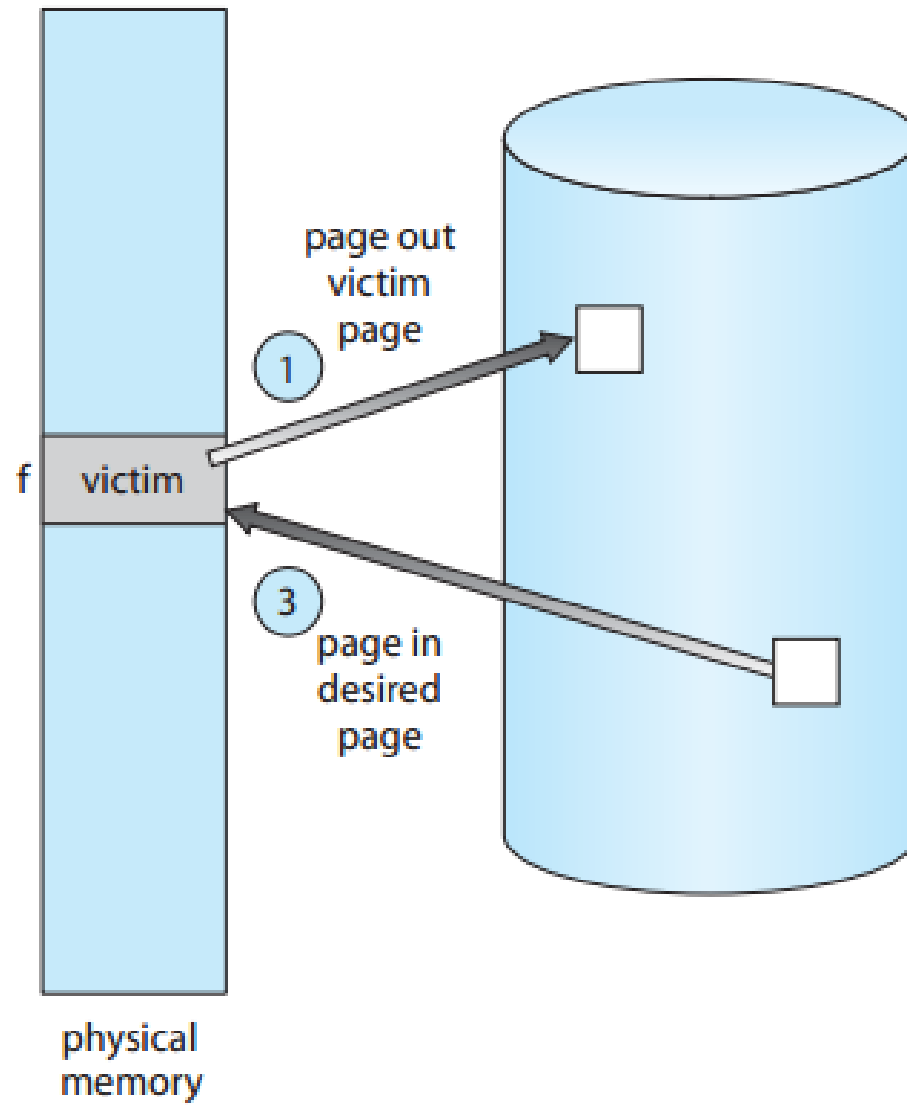
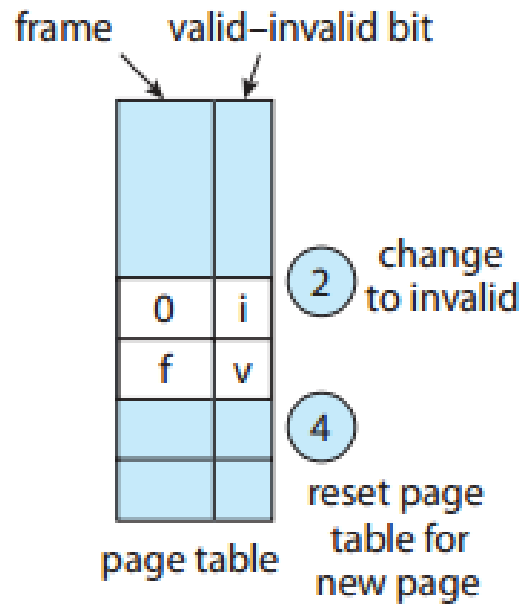


Figure 9.11 Graph of page faults versus number of frames.

# Lỗi trang và thay thế trang

39

- ❑ Truy suất đến một trang được đánh dấu bất hợp lệ (invalid) sẽ làm phát sinh lỗi trang, hdh sẽ thực hiện :
  1. Xác định vị trí trang muốn truy suất trên đĩa
  2. Tìm một khung trang trống trong bộ nhớ chính
    - a. Nếu tìm thấy -> đến bước 3
    - b. Nếu ko còn khung trang trống -> sử dụng thuật toán thay thế trang để chọn 1 nạn nhân
    - c. Ghi nội dung trang nạn nhân vào đĩa; cập nhật nội dung bảng trang, bảng khung trang
  3. Chuyển trang cần truy suất từ đĩa vào bộ nhớ chính; cập nhật nội dung bảng trang, bảng khung trang
  4. Tái kích hoạt tiến trình của người dùng





# Các thuật toán thay thế trang

41

- ❑ Mục tiêu : chọn trang “nạn nhân” là trang mà sau khi **thay thế sẽ gây ra ít lỗi trang nhất**
- ❑ Các thuật toán :
  - FIFO
    - Ghi nhận **thời điểm trang vào bnc => trang ở lâu nhất** được chọn
  - OPT (OPTimal)
    - Thay thế trang **sẽ lâu được sử dụng nhất trong tương lai** => ko khả thi
  - LRU (least recently used)
    - Ghi nhận thời điểm cuối cùng trang được truy cập => trang **lâu nhất chưa được truy suất** là trang được chọn

# Các thuật toán thay thế trang

42

## □ Bài tập 4

Xét chuỗi truy xuất bộ nhớ sau:

**1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3**

Giả sử **bộ nhớ vật lí có 4 khung trang**. Minh họa kết quả trình thay thế trang với các thuật toán thay thế sau:

a) FIFO

b) OPT

c) LRU

# Các thuật toán thay thế trang

43

□ FIFO : số lỗi trang 11

1 2 3 4 2 1 5 6 2 1 2 3 7 6 3  
\* \* \* \* \* \* \* \* \* \* \* \*

①	1	1	1	1	1	⑤	5	5	5	5	③	3	3	3
	②	2	2	2	2	2	⑥	6	6	6	6	⑦	7	7
		③	3	3	3	3	3	②	2	2	2	2	⑥	6
			④	4	4	4	4	4	①	1	1	1	1	1

- Sử dụng 4 khung trang
- Ban đầu cả 4 khung trang đều trống
- \* : có lỗi trang
- ① : trang được nạp từ đĩa vào bộ nhớ

# Các thuật toán thay thế trang

44

□ OPT : số lỗi trang 7

1 2 3 4 2 1 5 6 2 1 2 3 7 6 3  
\* \* \* \* \* \* \*

①	1	1	1	1	1	1	1	1	1	1	1	⑦	7	7
	②	2	2	2	2	2	2	2	2	2	2	2	2	2
		③	3	3	3	3	3	3	3	3	3	3	3	3
			④	4	4	⑤	⑥	6	6	6	6	6	6	6

# Các thuật toán thay thế trang

45

□ LRU : số lỗi trang 9

1 2 3 4 2 1 5 6 2 1 2 3 7 6 3  
\* \* \* \* \* \* \* \* \*

①	1	1	1	1	1	1	1	1	1	1	1	1	⑥	6
	②	2	2	2	2	2	2	2	2	2	2	2	2	2
		③	3	3	3	⑤	5	5	5	5	③	3	3	3
			④	4	4	4	⑥	6	6	6	6	⑦	7	7

# Các thuật toán thay thế trang

46

## ❑ Đánh giá

- Số lỗi trang :  $\text{FIFO} > \text{LRU} > \text{OPT}$

## ❑ Nhận xét :

- FIFO : dễ cài đặt
- OPT : Không khả thi
- LRU : cần hỗ trợ của phần cứng để lưu thông tin về thời gian page được tham chiếu
  - Ghi nhận nhãn thời gian
  - Hoặc dùng stack

# Các thuật toán thay thế trang

47

## ❑ Các thuật toán xấp xỉ LRU (*LRU-Approximation*)

- Thuật toán với các bit reference phụ trợ
- Thuật toán “cơ hội thứ hai” (Clock or second-chance)
- Thuật toán “cơ hội thứ hai” nâng cao (*Not Recently Used - NRU*)
- Các thuật toán thống kê

Tham khảo :

<http://blog.vnamct.com/2014/04/he-dieu-hanh-chuong-7-bo-nho-ao.html>

# Các thuật toán thay thế trang

48

**Bài 9.** Một máy tính có 4 khung trang. Thời điểm nạp, thời điểm truy cập cuối cùng, và các bit reference (R), modify (M) của mỗi trang trong bộ nhớ được cho trong bảng sau:

Trang	Nạp	Truy cập cuối	R	M
0	126	279	0	0
1	230	260	1	0
2	120	272	1	1
3	160	280	1	1

Trang nào sẽ được chọn thay thế theo:

- a) Thuật toán NRU
- b) Thuật toán FIFO
- c) Thuật toán LRU
- d) Thuật toán "cơ hội thứ 2"



# Các thuật toán thay thế trang

49

## □ Bài tập 5

Xem xét chuỗi tham chiếu trang sau:

**1,2,3,4,2,1,5,6,2,1,4,5,7,6,3,1**

Bao nhiêu page\_fault xuất hiện đối với giải thuật thay page FIFO, OPT, LRU  
(giả thuyết là được cấp 4 frame).

# Tóm tắt

50

- ❑ Các mô hình tổ chức bộ nhớ
  - Ưu/ nhược điểm
  - Cách chuyển đổi địa chỉ logic sang địa chỉ vật lý
- ❑ Bộ nhớ ảo
  - Khái niệm / ưu điểm
  - Các thuật toán thay thế trang

# Ôn tập quản lý process + quản lý bộ nhớ

51

- ❑ Thuật toán điều phối tiến trình  
(chương quản lý process )
- ❑ Chuyển đổi địa chỉ trong mô hình cấp phát bộ nhớ
- ❑ Khái niệm về bộ nhớ ảo
- ❑ Bài toán thay thế trang