Phần 3: Quản lý bộ nhớ

Trường Đại học Xây Dựng
Khoa Công nghệ Thông tin

Các vấn đề trình bày

- **3.1 Các khái niệm cơ bản**
- **3.2 Quản lý bộ nhớ thật**
- **3.3 Quản lý bộ nhớ ảo**
- **3.4 Phân trang**
- **3.5 Phân đoạn**
- **3.6 Quản lý bộ nhớ ảo phân trang và phân đoạn**

3.1 Các khái niệm cơ bản

- **Một trong các hoạt động cơ bản của máy tính là lưu trữ dữ liệu dạng nhị phân. Các dữ liệu này là các chương trình hoặc số liệu mà Vi xử lý đưa ra hoặc đọc vào tuỳ theo yêu cầu. Bộ nhớ là các thiết bị để thực hiện nhiệm vụ lưu trữ dữ liệu của máy vi tính.**
- **Mỗi ô nhớ được xác định bởi một địa chỉ. Thông thường mỗi ô nhớ có dung lượng là 1 byte. Các byte được ghép thành từ. Những máy 16 bit số liệu thì tổ chức 2 byte/từ, còn các máy 32 bit số liệu thì độ dài từ gấp đôi (4 byte/từ).**
- **Bộ nhớ là một dãy các ô nhớ liên tiếp nhau**

Các khái niệm cơ bản

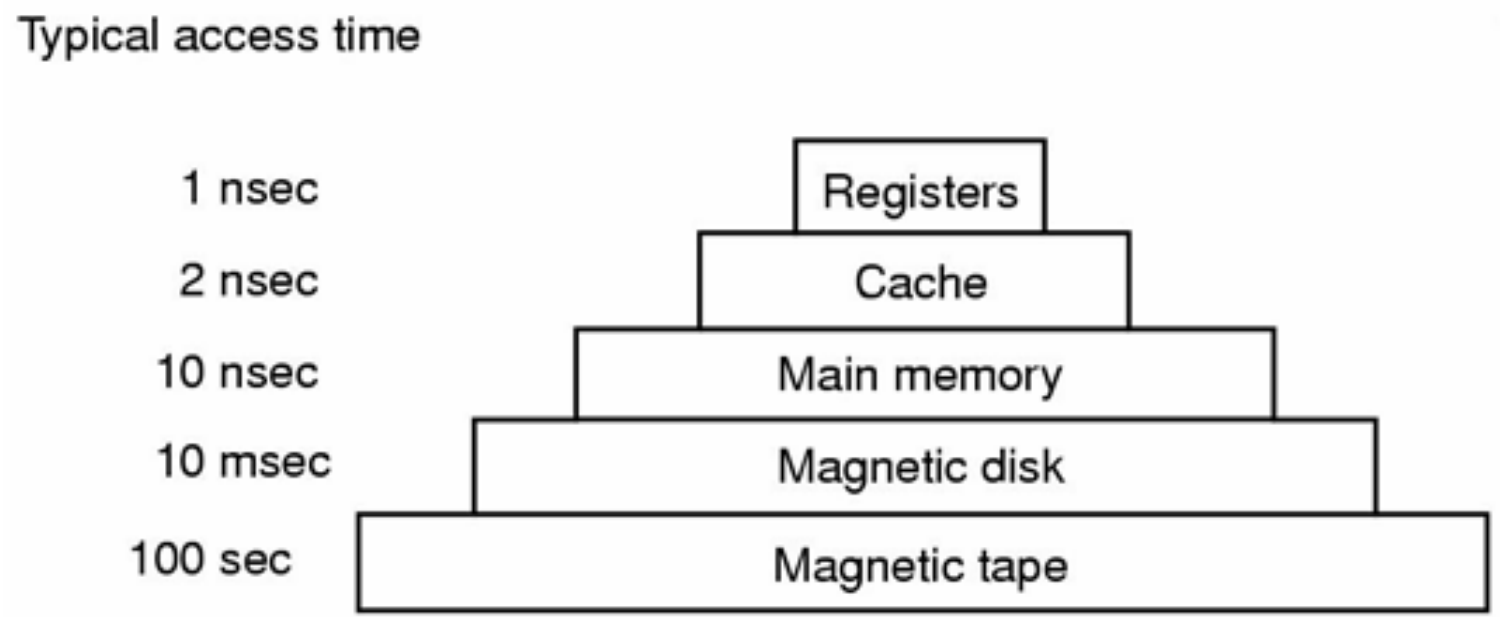
- **Bộ nhớ là thiết bị lưu trữ duy nhất mà qua đó CPU có thể thực hiện trao đổi thông tin với môi trường bên ngoài. Do vậy nhu cầu tổ chức quản lý bộ nhớ là một trong những nhiệm vụ cơ bản của hệ điều hành.**
- **Chương trình = Tập các câu lệnh (chỉ thị máy) + dữ liệu**
- **Hệ điều hành có trách nhiệm cung cấp không gian nhớ cho các tiến trình khi có yêu cầu.**

Các khái niệm cơ bản

- **Địa chỉ vật lý – địa chỉ tuyệt đối: Là địa chỉ thực trong bộ nhớ, được cấp phát cho các biến khi thực hiện chương trình.**
- **Địa chỉ logic - địa chỉ tương đối:**
  - ***Là vị trí nhớ độc lập với cấu trúc, tổ chức vật lý của bộ nhớ.***
  - ***Là địa chỉ do hệ thống tạo ra và được cấp phát cho các biến khi dịch chương trình.***
- **Bộ nhớ logic: Là tập hợp tất cả địa chỉ logic phát sinh sau khi dịch chương trình.**
- **Bộ nhớ vật lý: Là tập hợp tất cả các địa chỉ vật lý tương ứng với các địa chỉ logic khi thực hiện chương trình.**
- **Chú ý: Chương trình của người sử dụng chỉ thao tác trên các địa chỉ logic chứ không thao tác trên các địa chỉ vật lý. Địa chỉ vật lý chỉ được xác định khi thực hiện truy xuất chương trình.**

Các khái niệm cơ bản

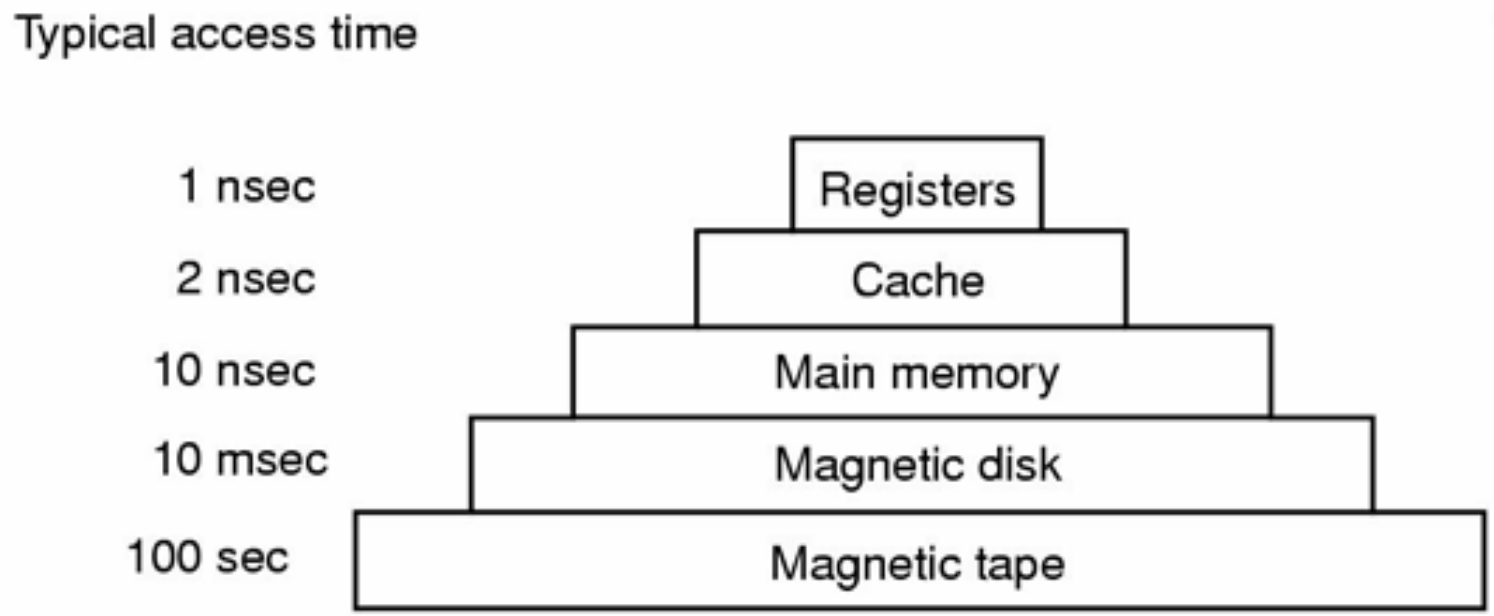
- **Máy tính thường sử dụng các loại bộ nhớ chính:**



Bộ nhớ máy tính

Các khái niệm cơ bản

- **Máy tính thường sử dụng các loại bộ nhớ chính:**



Bộ nhớ máy tính

Các khái niệm cơ bản

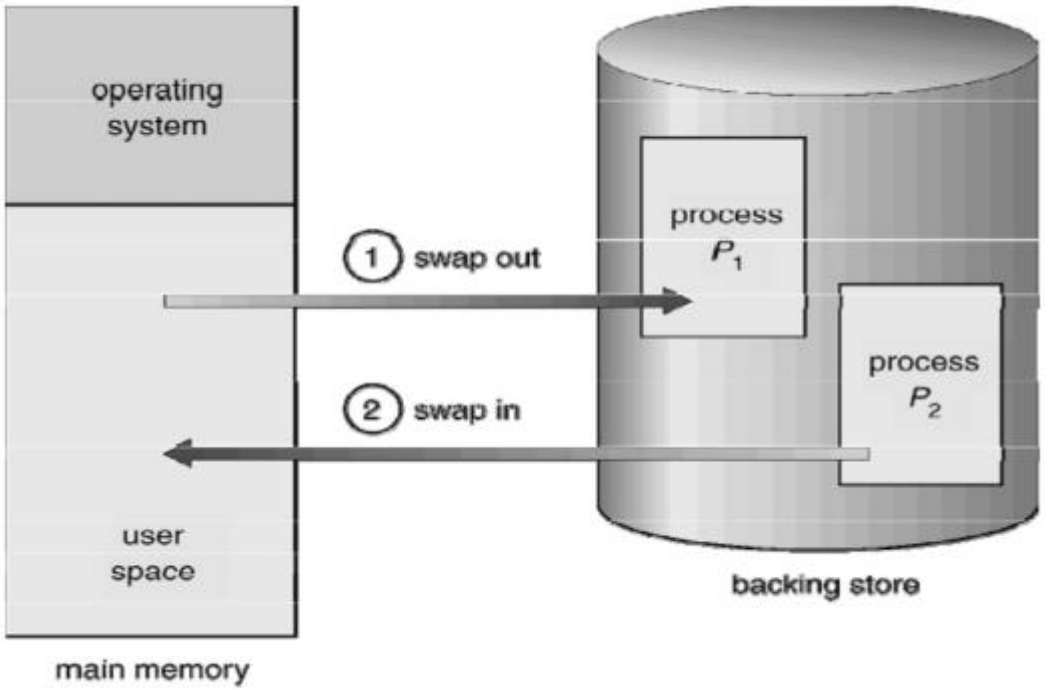
- **Bộ nhớ máy tính:**
  - ***Cache: Giá thành cao, dung lượng hạn chế, tốc độ truy cập nhanh. Thường nằm ngay trong bộ vi xử lý lưu trữ bản sao các dữ liệu thường hay sử dụng để truy nhập nhanh hơn.***
  - ***Bộ nhớ chính RAM: Giá thành, dung lượng, tốc độ trung bình. Lưu trữ các dữ liệu và lệnh chương trình đang được sử dụng.***
  - ***Đĩa cứng: Dung lượng rất lớn, chậm, giá rẻ.***
- **Việc quản lý bộ nhớ phải tận dụng tối đa các ưu điểm của từng loại bộ nhớ máy tính để đảm bảo cung cấp cho người lập trình một không gian bộ nhớ tốt nhất, xử lý dữ liệu nhanh và hiệu quả.**

Các khái niệm cơ bản

- **Cơ chế overlay**
  - **Tại mỗi thời điểm, chỉ giữ lại trong bộ nhớ những lệnh hoặc dữ liệu cần thiết, giải phóng các lệnh/dữ liệu chưa hoặc không cần dùng đến**
  - **Cơ chế này rất hữu dụng khi kích thước một process lớn hơn không gian bộ nhớ cấp cho process đó.**
  - **Cơ chế này được điều khiển bởi người sử dụng (thông qua sự hỗ trợ của các thư viện lập trình) chứ không cần sự hỗ trợ của hệ điều hành**

Các khái niệm cơ bản

- Cơ chế swapping
  - Một process có thể tạm thời bị swap ra khỏi bộ nhớ chính và lưu trên một hệ thống lưu trữ phụ. Sau đó, process có thể được nạp lại vào bộ nhớ để tiếp tục quá trình thực thi.

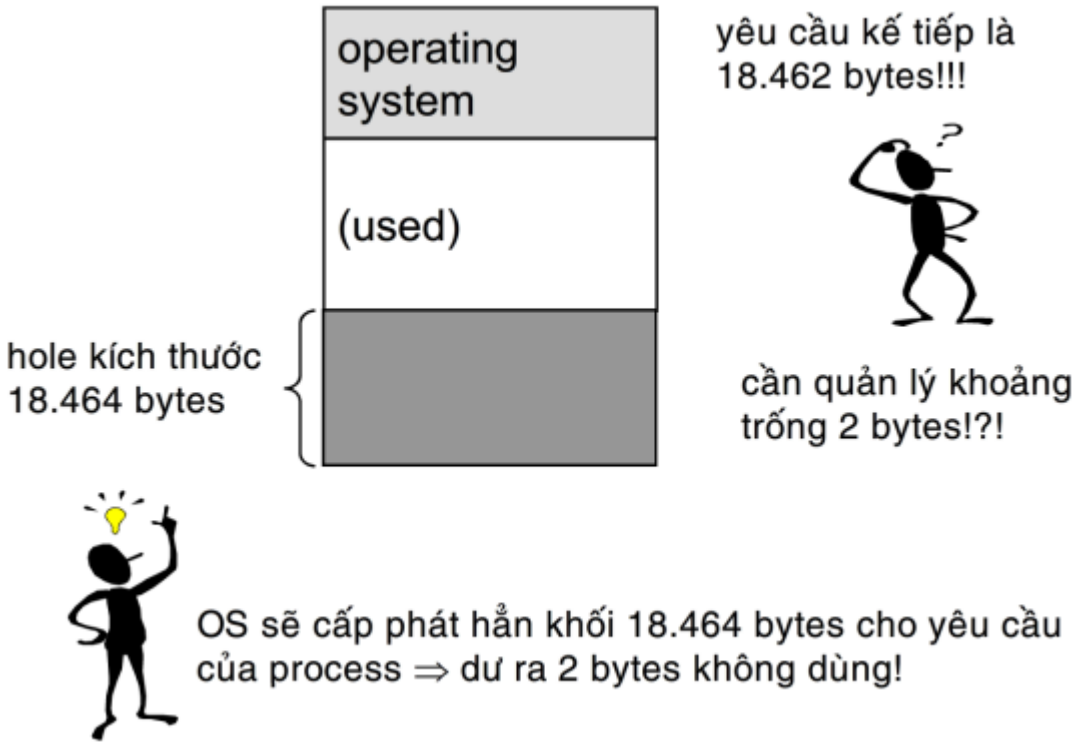


Các khái niệm cơ bản

- **Phân mảnh ngoại (extennal fragmentation)**
  - ***Kích thước không gian bộ nhớ còn trống đủ để thỏa mãn một yêu cầu cấp phát, tuy nhiên không gian nhớ này không liên tục => phải dùng cơ chế kết nối (compaction)***
- **Phân mảnh nội (internal fragmentation)**
  - ***Kích thước vùng nhớ được cấp phát lớn hơn vùng nhớ yêu cầu. Ví dụ: cấp một khoảng trống 16.464 bytes cho một process yêu cầu 16.462 bytes***
  - ***Hiện tượng phân mảnh nội thường xảy ra khi bộ nhớ thực (physical memory) được chia thành các khối có kích thước cố định (fixed – sized block) và các process được cấp phát theo đơn vị khối***

Các khái niệm cơ bản

- **Phân mảnh nội (internal fragmentation)**

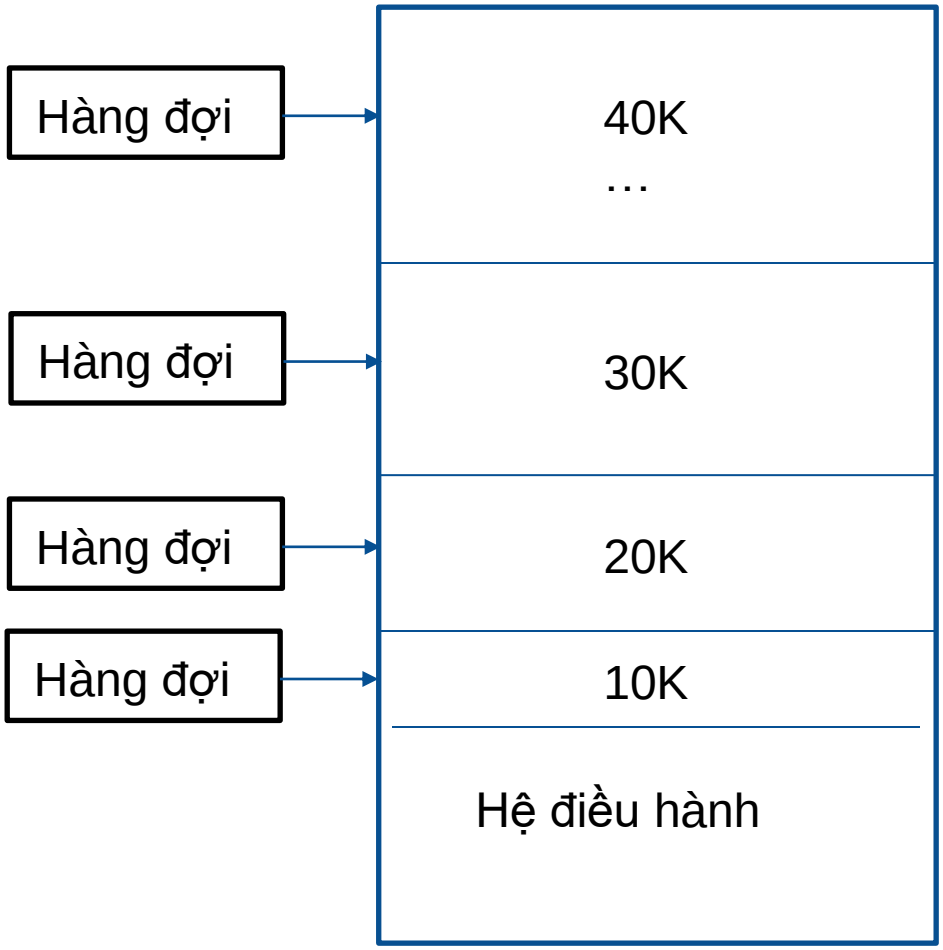


3.2 Quản lý bộ nhớ thật

- **Một Process phải được nạp hoàn toàn vào bộ nhớ thì mới được thực thi.**
- **Khi máy tính có dung lượng Ram tương đối lớn (512M~ G), còn các phần mềm có dung lượng nhỏ ta có thể sử dụng 1 trong 3 kỹ thuật quản lý bộ nhớ sau đây.**
  - ***Kỹ thuật phân vùng tĩnh nhiều hàng chờ độc lập (Modeling Multiprograming)***
  - ***Kỹ thuật phân vùng tĩnh một hàng chờ***
  - ***Kỹ thuật phân vùng động***

3.2.1 Kỹ thuật phân vùng tĩnh nhiều hàng chờ

- **HĐH được load vào vùng nhớ thấp của Ram**
- **Phần trống còn lại của Ram được chia thành nhiều phân vùng kích thước tăng dần (10K-20K-30K…)**
- **Mỗi phân vùng có 1 hàng chờ tương ứng.**
- **Khi chạy ứng dụng HĐH sẽ chọn phân vùng có kích thước bé nhất nhưng >= kích thước của ứng dụng để sắp xếp vào hàng chờ tương ứng**
- **HĐH sẽ phục vụ các tiến trình trong hàng chờ theo thuật toán FIFS**

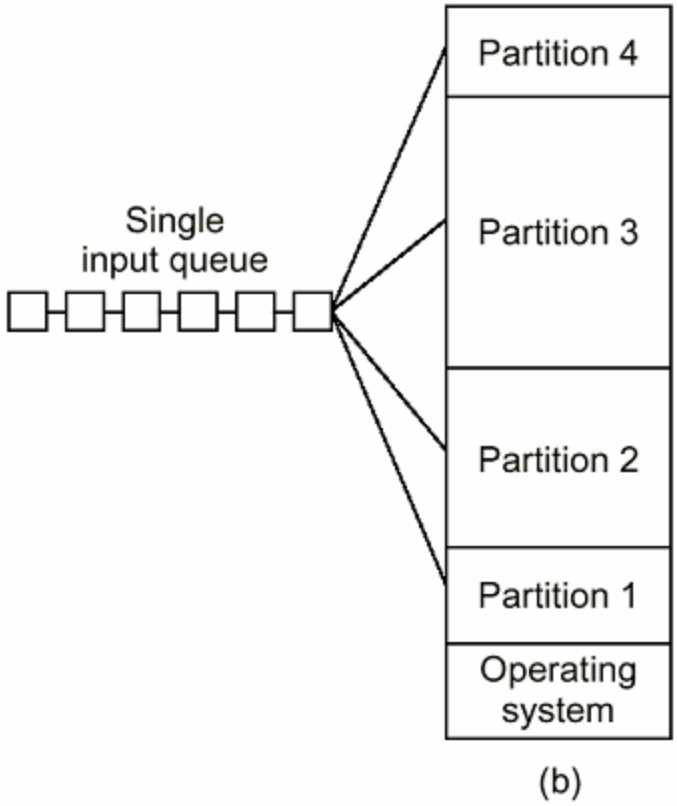


Kỹ thuật phân vùng tĩnh nhiều hàng chờ

- **Việc sử dụng kỹ thuật phân vùng tĩnh nhiều hàng chờ có 2 khuyết điểm chính:**
  - ***Thứ nhất, kích thước các phân vùng tĩnh là cố định và thường không khớp với các kích thước của các ứng dụng. Do đó phần bộ nhớ không sử dụng ở các phân vùng sẽ bị lãng phí → Hiện tượng phân mảnh nội vi.***
  - ***Thứ hai, Khi máy tính chạy nhiều chương trình nhỏ. Các chương trình sẽ xếp hàng để chờ ở phân vùng có kích thước nhỏ trong khi phân vùng có kích thước lớn không được sử dụng.***

3.2.2 Kỹ thuật phân vùng tĩnh một hàng chờ duy nhất

- HĐH được load vào vùng nhớ thấp của Ram
- Phần trống còn lại của Ram được chia thành nhiều phân vùng kích thước tăng dần (10K-20K-30K…)
- Chỉ có 1 hàng chờ duy nhất cho các ứng dụng
- Khi 1 phân vùng rỗng, HĐH sẽ chọn 1 tiến trình trong hàng chờ để phục vụ.

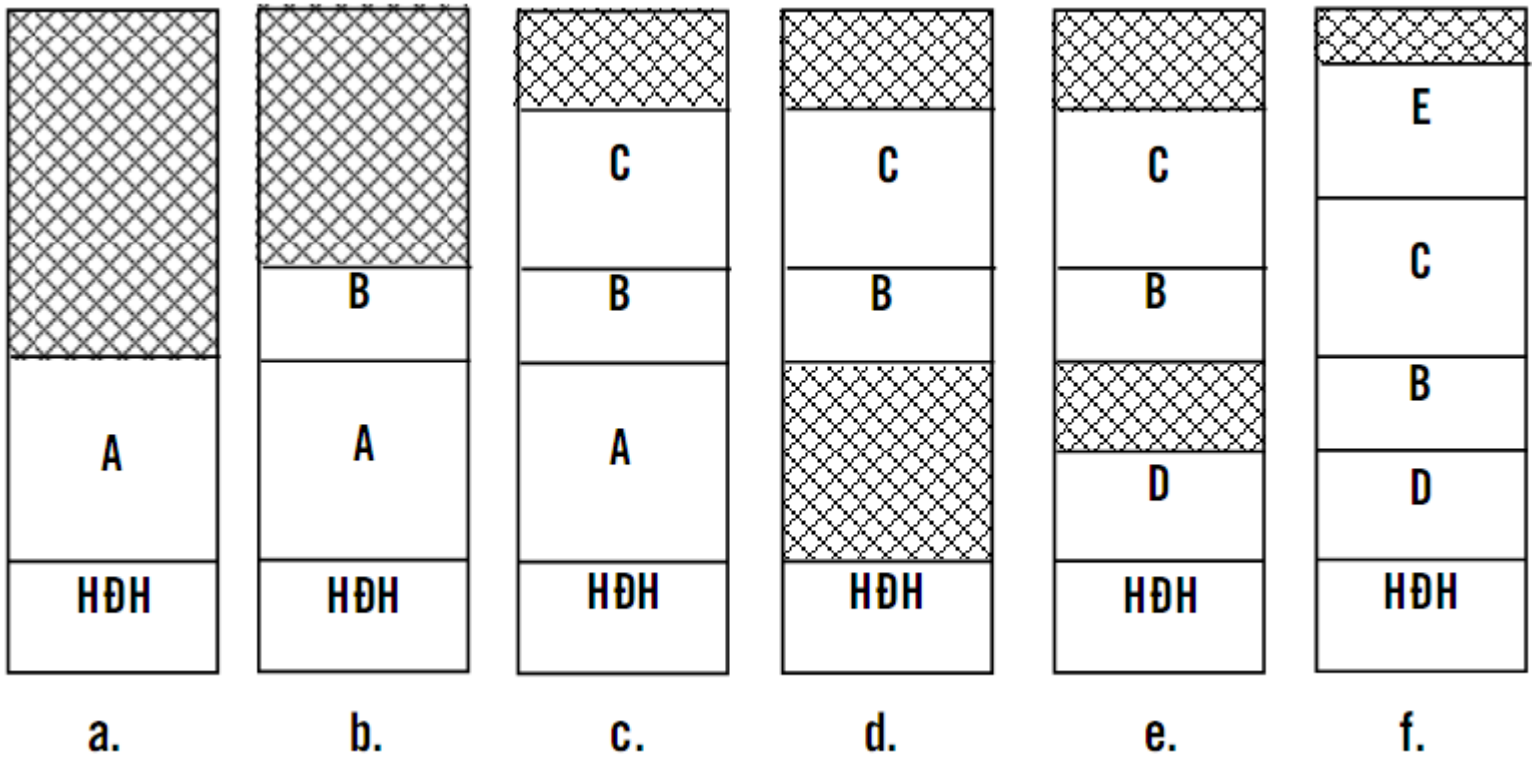


Các chiến lược cấp phát bộ nhớ

- **First-Fit: Cấp khối nhớ đầu tiên thỏa mãn**
- **Best-Fit: Cấp phát khối nhớ bé nhất đủ để đáp ứng cho tiến trình. Phải duyệt toàn bộ danh sách**
- **Worst-Fit: Cấp phát khối nhớ lớn nhất đáp ứng được tiến trình. Phải duyệt toàn bộ danh sách**
- ***Như vậy: First-Fit và Best-Fit tốt hơn Worst-Fit về vấn đề tốc độ và tận dụng bộ nhớ***

3.2.3 Kỹ thuật phân vùng động

- **Kỹ thuật phân vùng động**



Phân vùng động

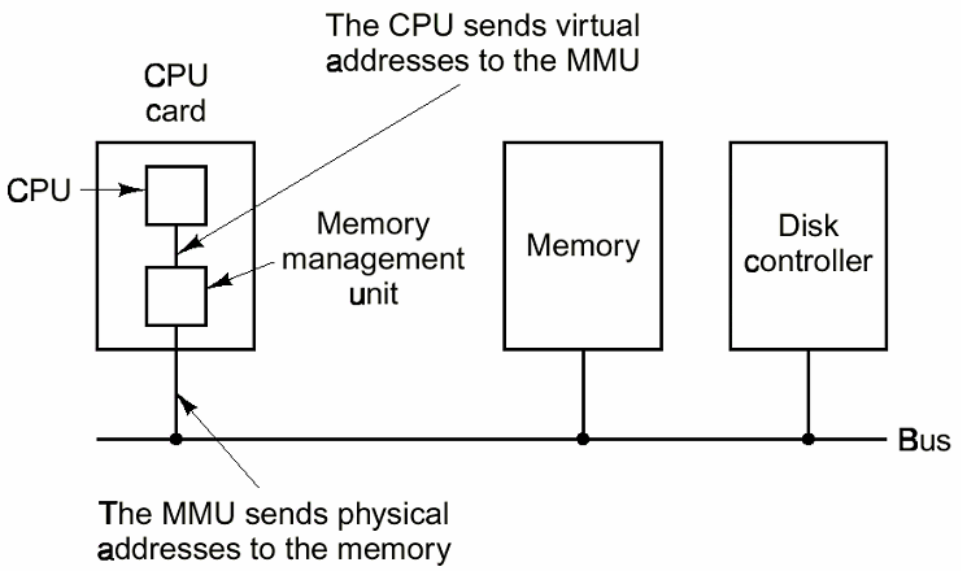
- **Hình a :** ***HĐH được load vào bộ nhớ thấp của RAM. Khi ứng dụng A cần chạy, HĐH tạo phân vùng động vừa đúng kích thước của phần mềm A và nạp A vào phân vùng vừa tạo. Phần bộ nhớ trống còn lại sẽ được sử dụng cho các chương trình tiếp theo.***
- **Hình d:** ***Chương trình A kết thúc và trả lại phân vùng của mình.***
- **Hình e:** ***Chương trình D có kích thước < kích thước chương trình A cần chạy. HĐH tạo phân vùng ngay trên phân vùng A vừa trả lại để cho D chạy.***
- **Hình f:** ***Chương trình E có kích thước lớn, HĐH không tìm được phân vùng phù hợp trong khi bộ nhớ trống đủ để đáp ứng tiến trình → đây gọi là hiện tượng phân mảnh ngoại vi. Khi đó nó dời các phân vùng đang chạy ứng dụng lại liền nhau để tạo vùng nhớ trống duy nhất và tạo phân vùng cho E chạy.***

3.3 Quản lý bộ nhớ ảo

- **Các phương pháp quản lý bộ nhớ đã nghiên cứu đều có chung ý tưởng là nạp toàn bộ file phần mềm vào bộ nhớ trước khi chạy ứng dụng. Do đó, khi các máy tính phải chạy đồng thời nhiều ứng dụng có kích thước lớn thì hệ điều hành sẽ không thể đáp ứng được.**
- **Để khắc phục tình trạng đó người ta sử dụng phương pháp quản lý bộ nhớ ảo với ý tưởng: không nạp hết toàn bộ file phần mềm vào bộ nhớ trước khi chạy mà chỉ khi chạy đến lệnh nào và cần truy xuất dữ liệu nào thì hệ điều hành mới nạp phần dữ liệu đó vào bộ nhớ. Sau khi thực hiện xong giải phóng phần dữ liệu đó để chứa code và dữ liệu khác.**

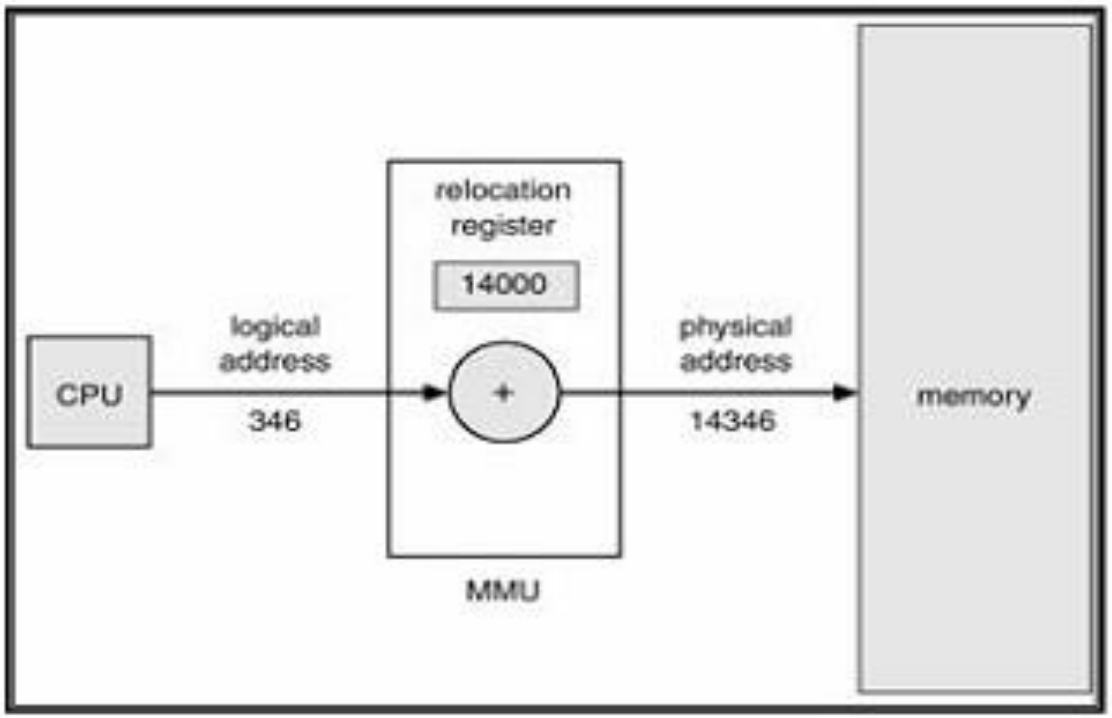
Quản lý bộ nhớ ảo

- **Việc quản lý bộ nhớ ảo như vậy sẽ gây ra vấn đề về tốc độ thực hiện chương trình do phải chờ việc giải phóng/nhập dữ liệu vào bộ nhớ. Để khắc phục người ta thực hiện quản lý bộ nhớ ảo bằng phần cứng. Đơn vị quản lý bộ nhớ ảo là MMU (Memory Management Unit).**



Quản lý bộ nhớ

- **Trong MMU có thanh ghi Relocation(định vị lại) sử dụng để tính toán địa chỉ vật lý từ địa chỉ ảo của 1 tiến trình của người sử dụng**



Quản lý bộ nhớ

Có 3 phương pháp quản lý bộ nhớ ảo khác nhau:

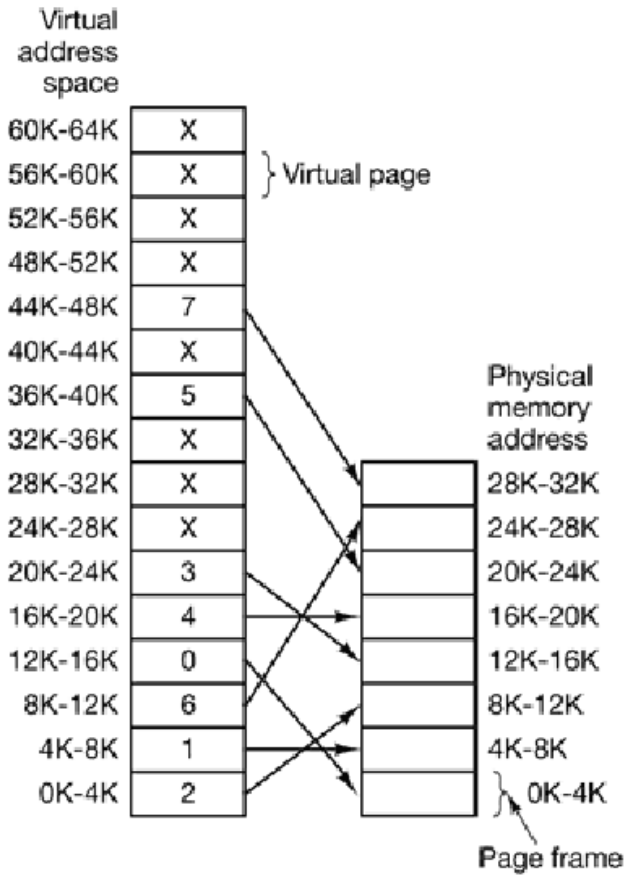
- **Quản lý bộ nhớ ảo theo phân trang (Paging)**
- **Quản lý bộ nhớ ảo theo phân đoạn (Segmentation)**
- **Quản lý bộ nhớ ảo theo phân trang và phân đoạn (Paging and Segmentation)**

3.4 Phân trang (Paging)

- **Nguyên lý:**
  - **Hệ điều hành chia bộ nhớ ảo chương trình thành nhiều đơn vị có kích thước bằng nhau, mỗi đơn vị này là 1 trang. Kích thước trang ảo = 2^i (256, 516, 1K, 2K,…) (Page)**
  - **Bộ nhớ Ram cũng được chia thành nhiều đơn vị quản lý, mỗi đơn vị được gọi là trang thật (frame) và có kích thước bằng trang ảo.**
  - **Trang thật là nơi chứa trang ảo khi cần thiết.Tại 1 thời điểm 1 trang thật chứa duy nhất 1 trang ảo, nhưng theo thời gian nó có thể chứa nhiều trang ảo khác nhau.**
  - **Hệ điều hành quản lý việc ánh xạ trang ảo và trang thật qua bảng trang (PCB – Page Control block)**

Phân trang

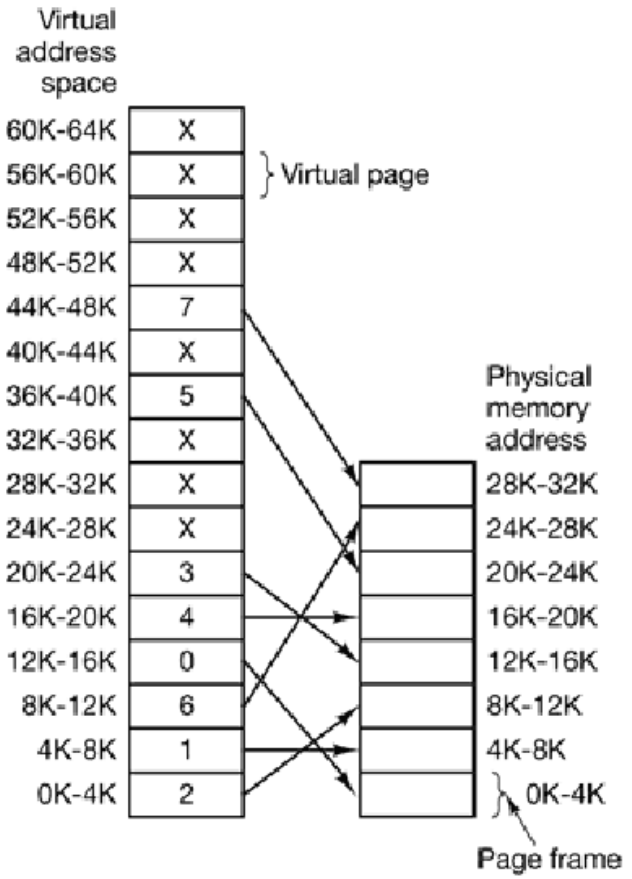
- **Ví dụ ánh xạ giữa địa chỉ ảo và địa chỉ thật.**
  - **Máy tính RAM 32KB được chia thành 8 trang thật, mỗi trang kích thước 4KB.**
  - **Chương trình kích thước 64KB được chi thành 16 trang ảo, kích thước 4KB**
  - **Hiện 8 trang ảo đang chiếm hết Ram**



Phân trang

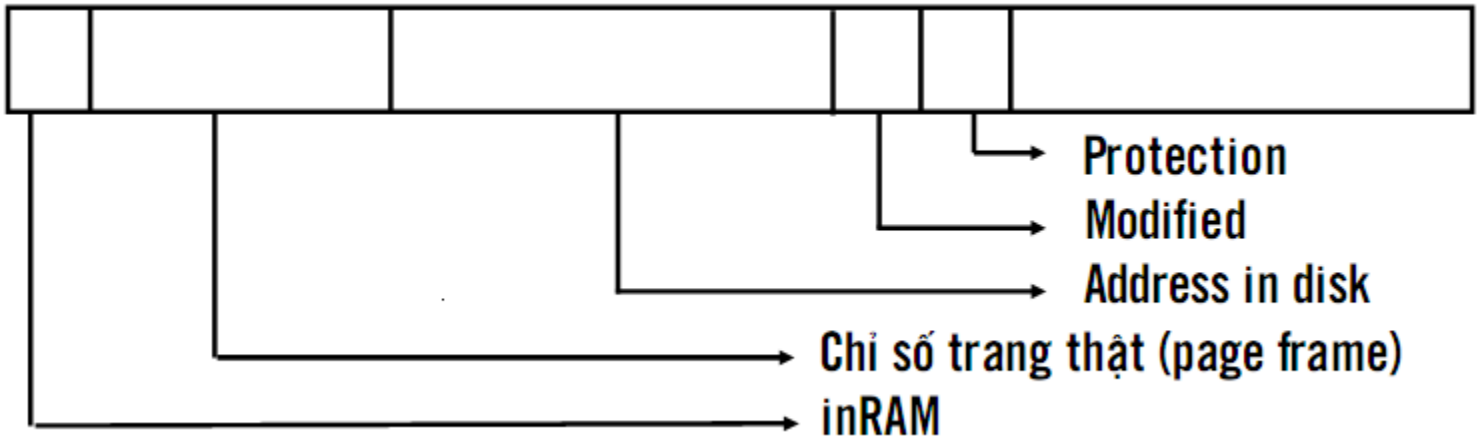
Ta có bảng trang như sau

Logical	Physical
3	0
1	1
0	2
5	3
4	4
9	5
2	6
11	7



Phân trang

- **Để quản lý việc ánh xạ trang ảo sang trang thật hệ điều hành sử dụng một bảng đặc tả trang ảo (bảng trang-page table), bảng trang có số phần tử bằng số trang ảo của chương trình tương ứng, mỗi phần tử của bảng có cấu trúc như sau:**



Phân trang

- **Chuyển đổi từ địa chỉ ảo sang địa chỉ thật:**
  - **Mỗi địa chỉ được sinh ra bởi CPU được chia thành 2 phần: Page number p (số trang) và Page offset d (số thứ tự trong trang).**
  - **Không gian ảo kích thước 2^m , kích thước trang ảo 2^n**
  - **Địa chỉ ảo có m bit, sử dụng m-n bit cao làm số hiệu trang và n bit thấp làm offset**

Phân trang

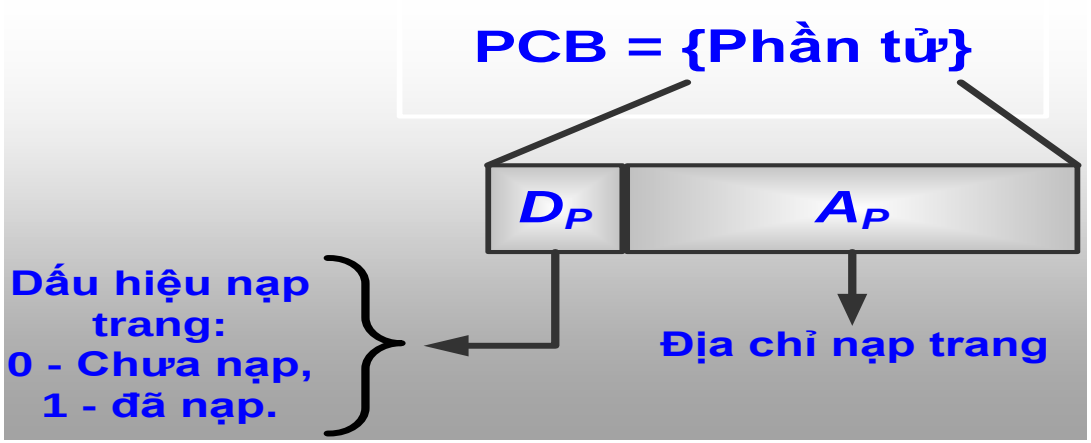
- **Ví dụ:**
  - **Giả sử kích thước luận lý là 32 bytes, kích thước trang thật là 4 bytes => có 8 trang đánh số từ 0->7.**
  - **Kích thước page number là 5-2=3 bit, kích thước page offset là 2bit**

Page number	Page offset
0	A
	B
	C
	D
…..	…..
7	A
	B
	C
	d

(m=5, n=2, p=8, d=4)

Phân trang

- **Khi chương trình được nạp vào bộ nhớ, hệ thống xây dựng 1 bảng quản lý trang PCB. Địa chỉ của bảng PCB được nạp vào thanh ghi Rp. Mỗi phần tử của PCB gồm 2 phần:**
  - ***D: cho biết trang đã được nạp vào bộ nhớ hay chưa.***
  - ***Ap: là địa chỉ trang vật lý chứa trang logic đang xét. Nếu D=0, Ap có thể chứa thông tin cần thiết để tìm trang ở bộ nhớ ngoài.***

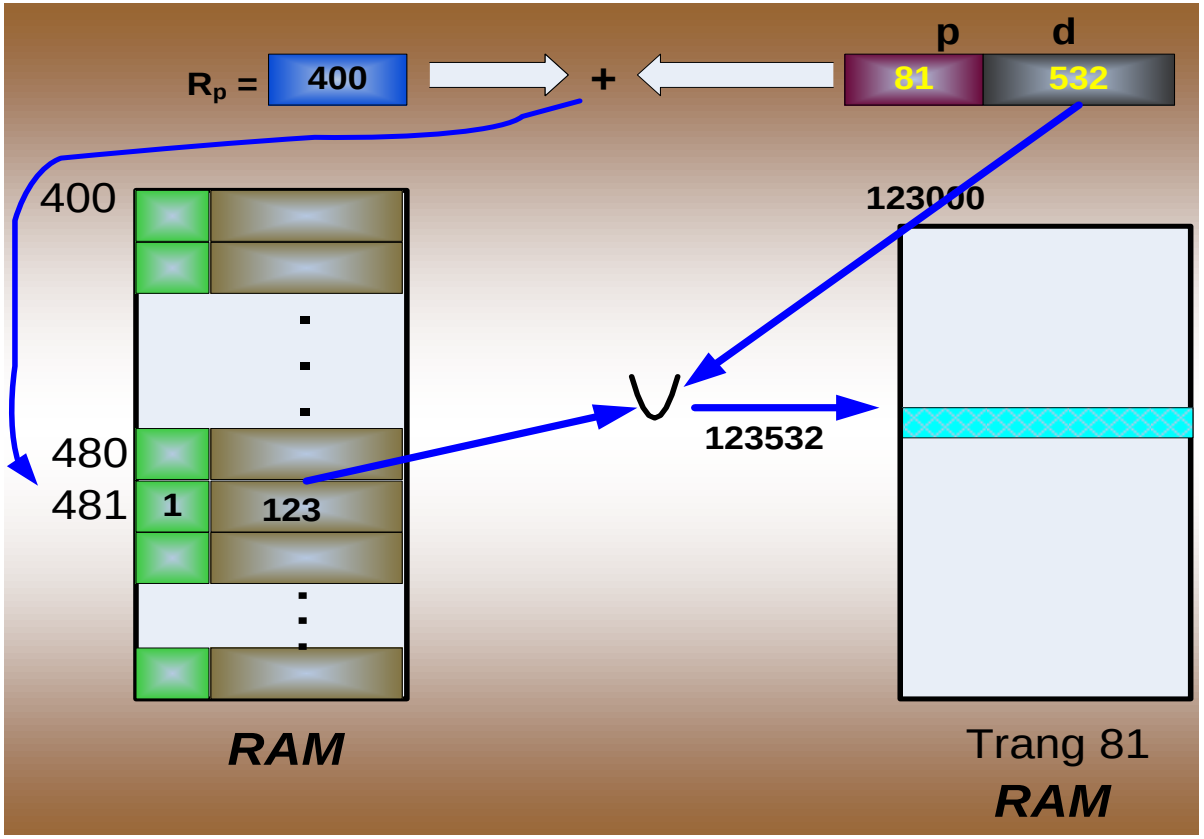


Phân trang

- **Để truy nhập dữ liệu hệ thống thực hiện như sau:**
  - **B1: lấy nội dung thanh ghi Rp + p để truy nhập tới phần tử thứ p trong PCB. Nếu D=0 thì hệ thống nạp trang vào bộ nhớ. Khi đó D=1 và Ap sẽ chứa địa chỉ trang trong bộ nhớ vật lý.**
  - **B2: Hệ thống lấy địa chỉ trang Ap ghép với d tạo ra địa chỉ vật lý của dữ liệu đã đưa vào đó và truy nhập tới địa chỉ vừa tính được để truy xuất dữ liệu.**

Phân trang

- **Ví dụ:**



Phân trang

- **Xét một không gian địa chỉ có 8 trang, mỗi trang có kích thước 1KB, ánh xạ vào bộ nhớ vật lý có 32 khung trang. Hỏi:**
  a) Địa chỉ logic và vật lý gồm bao nhiêu bit?
  b) Bảng trang có bao nhiêu mục?

8 trang → m-n=3

1KB → n=10

→m=13 bit để đánh logic

→15 bit để đánh vật lý

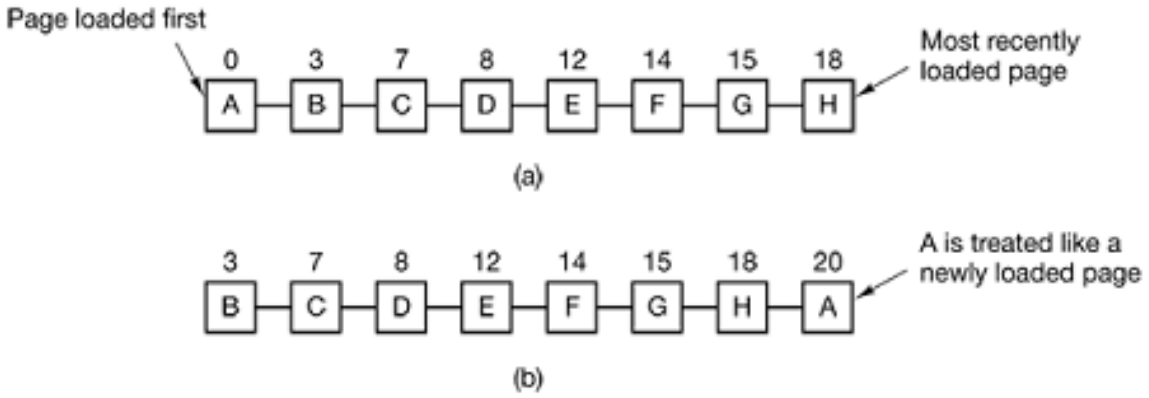
8 trang → 8 mục

Các phương pháp giải phóng trang thật

- **Như đã biết, khi cần nạp trang ảo máy sẽ tìm một trang thật trống. Tuy nhiên ít khi tìm được do số lượng trang thật hạn chế. Vì vậy máy phải tìm một trang thật nào đó và giải phóng nó để nạp trang ảo mới. Có nhiều phương pháp giải phóng trang thật khác nhau phụ thuộc vào mục tiêu xây dựng hệ điều hành.**
- **Dưới đây chúng ta nghiên cứu các phương pháp giải phóng trang thật cơ bản: FIFO, the second chance page replacement algorithm(cơ hội lần 2), Clock, NRU (not recently used), LRU(last recently used).**

FIFO

- **Dùng 1 danh sách chứa các trang thật đã dùng theo thứ tự trang nào mới nhất sẽ đưa xuống cuối danh sách, các trang dùng cũ nhất sẽ đứng đầu danh sách. Khi cần trang thật HĐH sẽ giải phóng các trang ở đầu danh sách trước.**



FIFO

- **Như đã biết HĐH thường được nạp vào Ram đầu tiên do đó Phương pháp này rất dễ giải phóng các trang chưa module hệ điều hành.**
- **Để khắc phục nhược điểm này ta sử dụng phương pháp cơ hội lần 2 với bit R (0/1) để xác định các trang dùng cũ nhất và không được truy xuất trong thời gian gần.**

Cơ hội lần 2

- **Biến R (0/1) được sử dụng như sau:**
  - ***Định kỳ R được xóa về 0***
  - ***Khi được truy xuất R được set lên 1***
- **Khi cần giải phóng trang, hệ điều hành tìm trang đầu danh sách và kiểm tra biến R của nó.**
  - ***Nếu R=1 thì tha nó, set R=0 và tìm đến phần tử tiếp theo.***
  - ***Nếu R=0 thì giải phóng***
- **Như vậy trang được giải phóng là trang không được truy xuất gần đây và cũ nhất.**

LRU (Last Recently Used)

- **Thay thế trang có lần sử dụng cuối cùng, cách thời điểm đổi trang lâu nhất.**
- **Để thực hiện phương pháp này, hệ điều hành ghi lại mốc thời gian mỗi lần trang được truy xuất. Mỗi khi cần giải phóng trang, ta chọn trang có mốc thời gian nhỏ nhất (xa thời điểm chọn nhất).**

NRU (Not Recently Used)

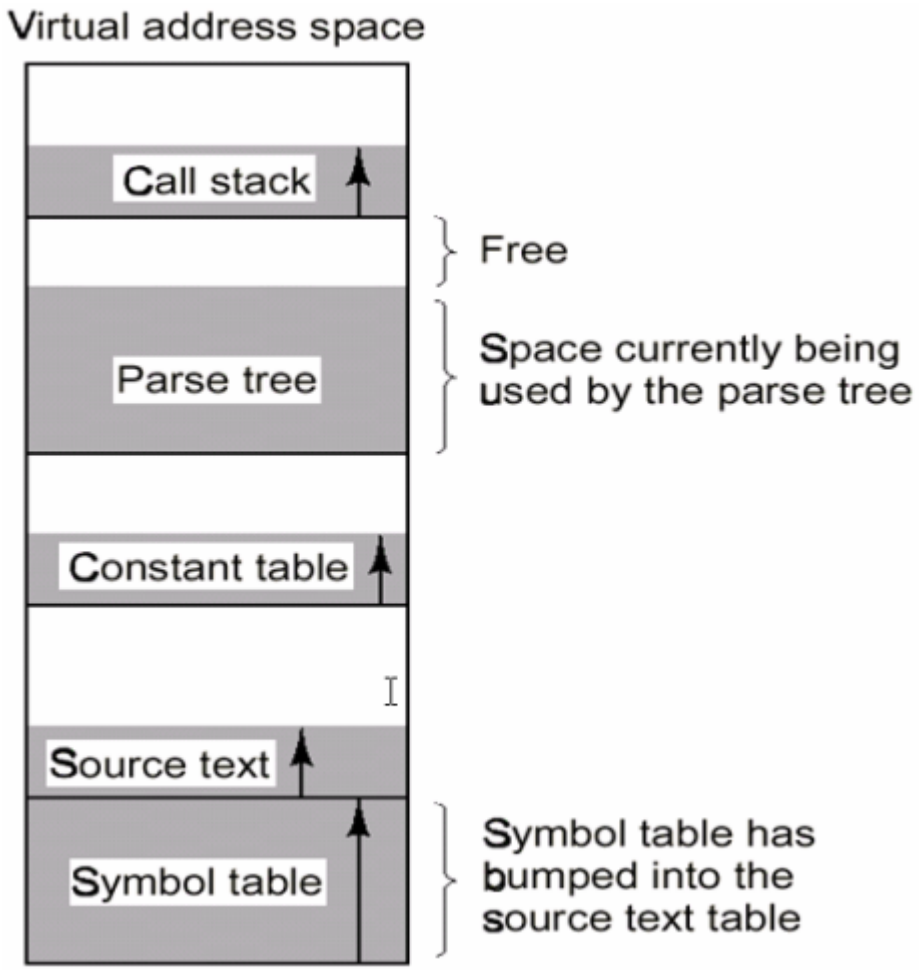
- **Sử dụng 2 bit trạng thái R và M như sau**
  - ***R (0/1): Định kỳ set về 0, mỗi khi trang bị truy xuất thì set lên 1.***
  - ***M (0/1): Mỗi lần nạp trang ảo, M set về 0. Mỗi lần thay đổi nội dung M set về 1.***
- **Như vậy 1 trang ảo có các sự kiện sau:**
  - ***S0: R=0, M=0: Trong thời gian gần, trang chưa bị truy xuất và chưa thay đổi nội dung***
  - ***S1: R=0, M=1: Trong thời gian gần, trang chưa bị truy xuất nhưng đã thay đổi nội dung***
  - ***S2: R=1, M=0: Trong thời gian gần, trang bị truy xuất nhưng không thay đổi nội dung***
  - ***S3: R=1, M=1: Trong thời gian gần, trang bị truy xuất và thay đổi nội dung***

NRU (Not Recently Used)

- **Như vậy thứ tự S0->S3 là thứ tự các trang đã được truy xuất và thay đổi nội dung từ lâu nhất đến mới nhất.**
- **Khi tìm trang để thay thế HĐH chọn theo thứ tự ưu tiên từ S0->S3**

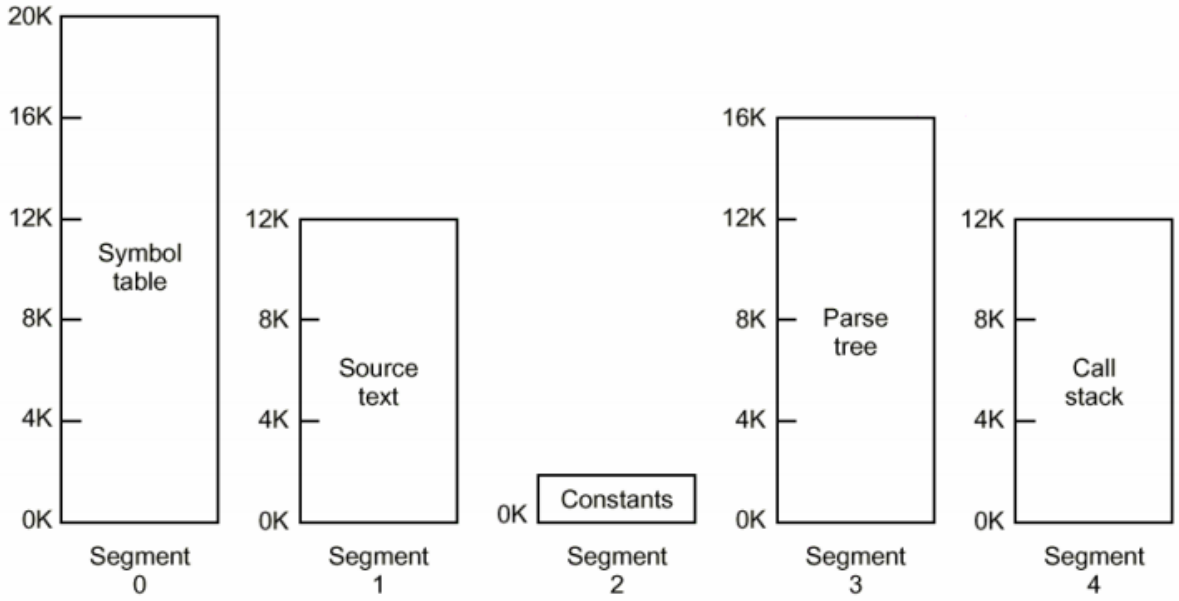
3.5. Phân đoạn (Segmentation)

- **Trong kỹ thuật phân trang, mặc dù không gian ảo mà chương trình có thể truy xuất có kích thức rất lớn (4GB), nhưng nó là không gian phẳng nên 1 số chương trình lớn vẫn có thể gặp phiền hà sau đây: chương trình tự chia không gian ảo của mình ra thành nhiều partition khác nhau để chứa những thông tin độc lập cần xử lý, trong quá trình chạy, nếu 1 trong các partition không đủ chỗ chứa thông tin thì chương trình sẽ bị dừng đột ngột.**



Phân đoạn (Segmentation)

- **Do đó, thay vì cấp phát cho chương trình 1 không gian phẳng duy nhất, nếu hệ thống cấp phát cho chương trình các không gian bộ nhớ độc lập có kích thước thay đổi động theo nhu cầu (miễn sao tổng kích thước của chúng bị hạn chế trên nào đó) như hình sau thì chương trình sẽ không gặp vấn đề tràn bộ nhớ:**

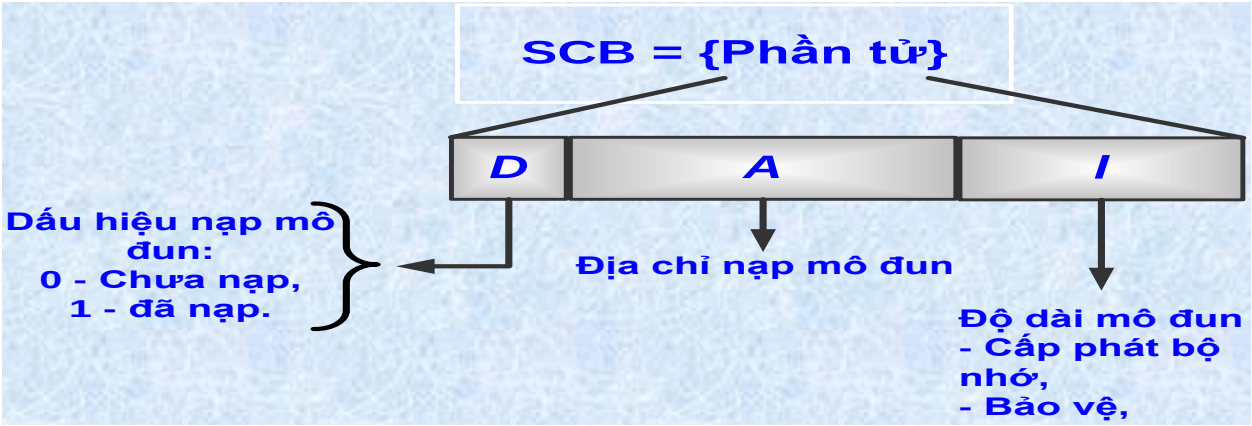


Phân đoạn (Segmentation)

- **Nguyên lý hoạt động:**
  - **Khi chạy, chương trình được phép truy xuất dữ liệu trong nhiều không gian khác nhau, mỗi không gian được gọi là segment. Mỗi segment có kích thước thay đổi được theo thời gian, ô nhớ đầu tiên của mỗi segment luôn bắt đầu từ 0.**
  - **Bộ nhớ RAM có kích thước nhỏ nào đó. Các segment của chương trình thường nằm trên đĩa cứng, khi cần thiết segment sẽ được nạp vào 1 vùng thích hợp trong RAM.**
  - **Tại từng thời điểm, 1 vùng nhớ RAM thật chứa tối đa 1 segment ảo, nhưng theo thời gian nó có thể chứa nhiều segment ảo khác nhau**
  - **Khi ứng dụng truy xuất 1 ô nhớ, nó xác định địa chỉ ô nhớ dạng phân cấp: segment+offset.**

Phân đoạn (Segmentation)

- **Bộ nhớ quản lý các segment này bởi SCB.**
- **Mỗi phần tử trong SCB đặc trưng bởi 3 trường tin:**
  - ***D: cho biết module đã nạp vào bộ nhớ hay chưa(D=0 chưa nạp, D=1 đã nạp)***
  - ***A: Địa chỉ của vùng nhớ sẽ định vị module***
  - ***L: Kích thước của module***



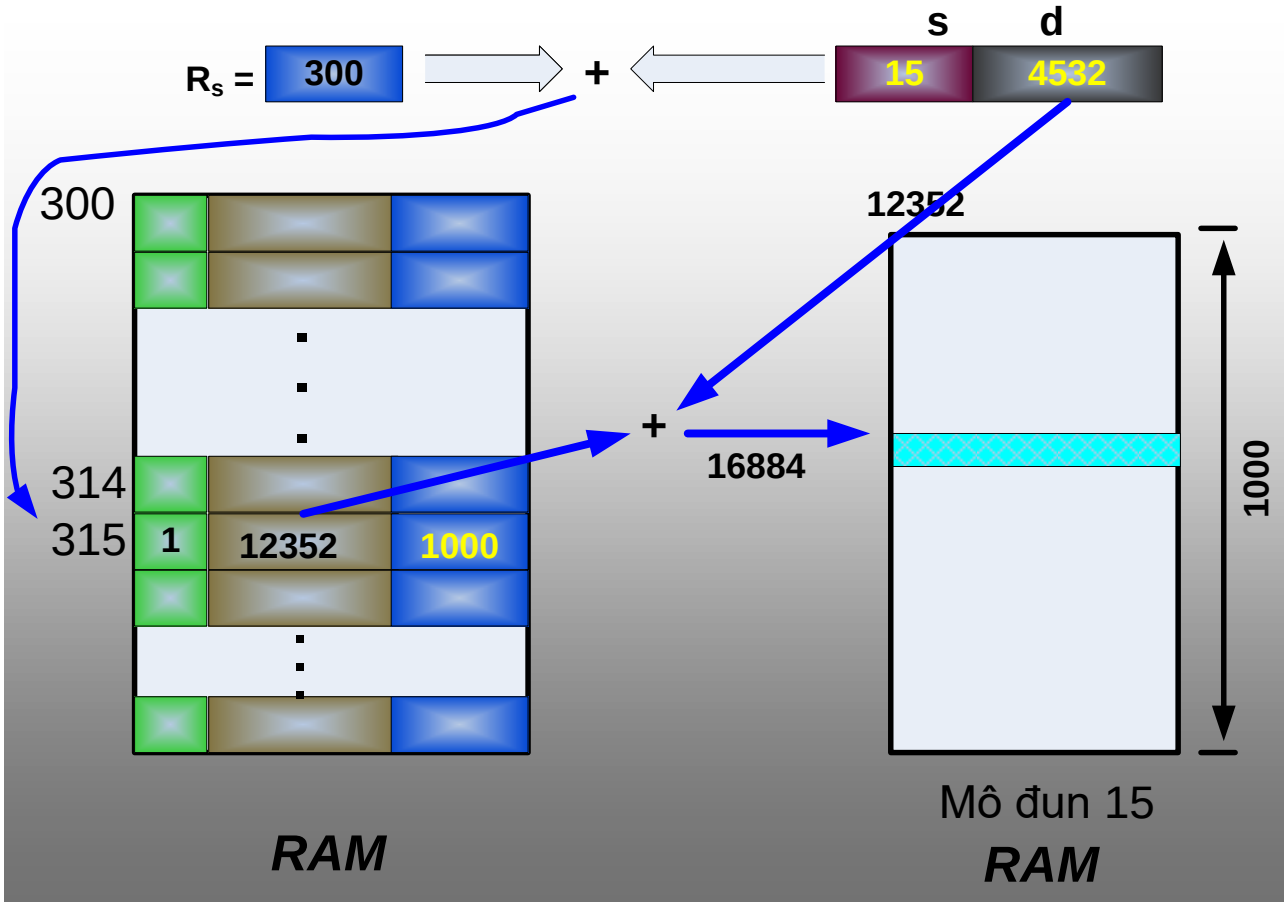
Phân đoạn (Segmentation)

- **SCB được xây dựng ngay khi biên dịch chương trình. Ban đầu chỉ có trường D và L có giá trị.**
- **Khi thực hiện, SCB được nạp vào bộ nhớ, địa chỉ đầu được đưa vào thanh ghi đoạn Rs.**
- **Địa chỉ truy cập dữ liệu được biểu diễn bởi cặp (s,d). Trong đó s là số hiệu module cần truy cập(ví dụ module thứ 2), d là địa chỉ tương đối tính từ đầu Segment.**
- **Để truy cập dữ liệu cần 2 bước:**
  - ***B1: Lấy nội dung thanh ghi Rs+s->phần tử thứ s trong SCB***
    - **Nếu D=0->nạp CT vào bộ nhớ, xin cấp phát không gian nhớ theo kích thước L. Trường địa chỉ A được chỉ tới module thứ s.**
    - **Nếu D=1, hệ thống thực hiện bước tiếp theo.**
  - ***B2: Truy nhập tới bộ nhớ theo địa chỉ A+d để truy xuất dữ liệu.***

Phân đoạn (Segmentation)

- **Ví dụ: module thứ 15 của chương trình (s=15) có địa chỉ tương đối d=4532, độ dài module L=1000, Địa chỉ nạp module A=12352, module đã được nạp vào bộ nhớ, nội dung thanh ghi Rs là 300. Để truy nhập tới module hệ thống thực hiện như sau:**
  - ***Lấy 300+15->phần tử thứ 315 trong SCB.***
  - ***Truy nhập đến địa chỉ 12352+4532=16884 để truy xuất dữ liệu.***

Phân đoạn (Segmentation)



Phân đoạn (Segmentation)

- **Ưu điểm: Không đòi hỏi công cụ tổ chức đặc biệt, có thể áp dụng trên mọi hệ thống.**
- **Nhược điểm: Hiệu quả sử dụng bộ nhớ phụ thuộc cấu trúc chương trình của người sử dụng.**

3.6. Quản lý bộ nhớ ảo kết hợp phân trang và phân đoạn

- **Quy trình đổi địa chỉ ảo sang địa chỉ thật có khuyết điểm trong trường hợp quản lý segment có kích thước lớn: ta khó/không tìm được vùng RAM trống chứa nó. Vì lý do này, trong thực tế, người ta phải kết hợp 2 phương pháp quản lý phân trang và phân đoạn lại, đây là phương pháp mạnh nhất hiện nay. Ý tưởng là hệ thống quản lý mỗi segment phần mềm như là 1 không gian ảo gồm nhiều trang ảo, mỗi lần chương trình truy xuất ô nhớ nằm trong trang ảo nào của segment nào, hệ thống sẽ tìm cách nạp nó vào RAM.**

Quản lý bộ nhớ ảo kết hợp phân trang và phân đoạn

- **Ý tưởng: Hệ thống quản lý mỗi segment phần mềm như 1 không gian ảo gồm nhiều trang ảo, mỗi lần chương trình truy xuất ô nhớ nằm trong trang ảo nào của segment nào, hệ thống sẽ tìm cách nạp nó vào Ram.**
- **Quy đổi địa chỉ ảo sang địa chỉ thật:**
  - ***1: Từ địa chỉ mà chương trình truy xuất gồm 2 phần: Segment (s) và offset. Hệ thống tách offset thành 2 phần page(p) và offset 1.***
  - ***2: Truy xuất record quản lý segment trong bảng đặc tả segment. Nếu D=1, thì bản đặc tả cho segment đã có trong Ram. Nếu không thì tìm cách nạp nó vào Ram.***
  - ***3. Truy xuất record quản lý trang ảo P trong bảng đặc tả trang. Nếu D=1 thì địa chỉ thật là***

Quản lý bộ nhớ ảo kết hợp phân trang và phân đoạn

- **3. Truy xuất record quản lý trang ảo P trong bảng đặc tả trang. Nếu D=1 thì địa chỉ thật là**

Page frame (p)	Offset

- **4. Nếu D=0, hệ thống sẽ tìm 1 trang rỗng K, nếu không có nó phải tìm cách loại bỏ 1 trang ít gây phiền hà nhất. Sau đó nạp trang ảo vào trang thật K.**
- **5. Hiệu chỉnh D=1 và page frame =K rồi quay lại bước 3.**