Chương 6

QUẢN LÝ BỘ NHỚ

Quản lý bộ nhớ

- Cơ sở
- Tải chồng (overlay)
- Tráo đổi (swapping)
- Phân phối bộ nhớ liên tục
- Phân trang (paging)
- Phân đoạn (segmentation)
- Phân đoạn kết hợp với phân trang (Segmentation with Paging)

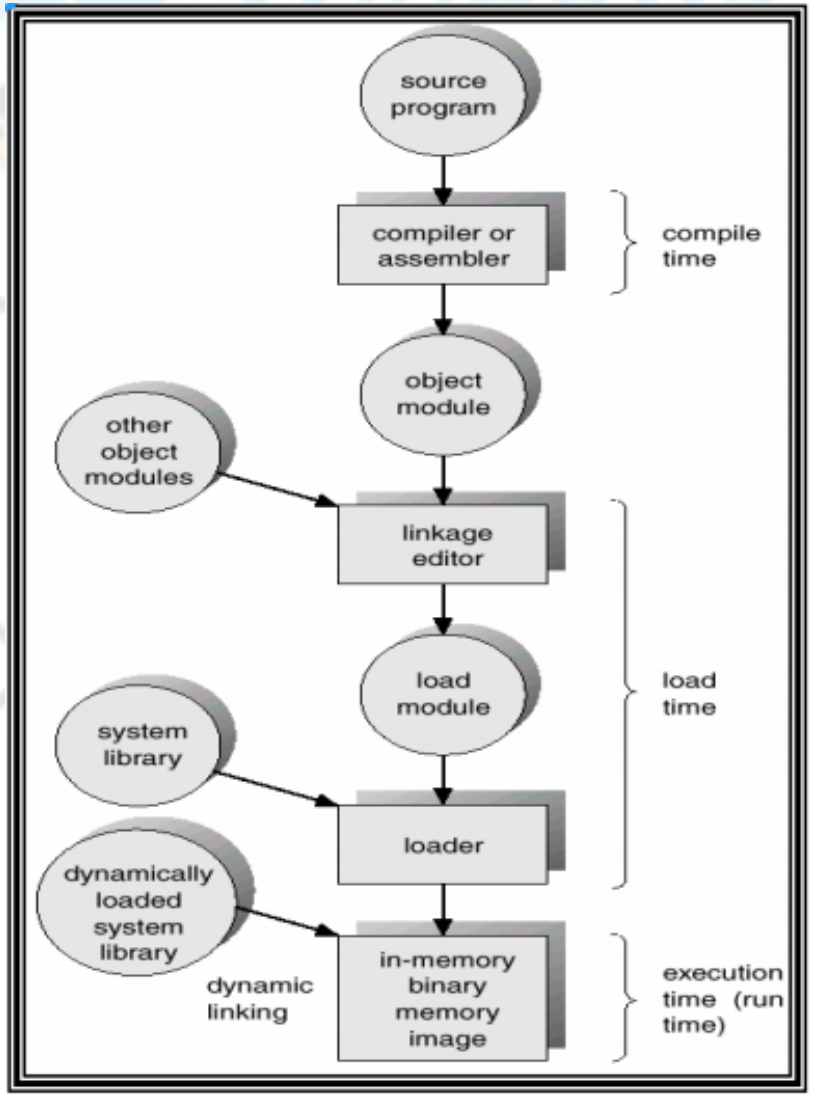
Cơ sở

- Chương trình muốn thực thi cần phải được tải vào bộ nhớ và đặt trong một tiến trình
- Hàng đợi vào (Input Queue)
  - Tập các chương trình trên đĩa, đang đợi tải vào bộ nhớ để thực hiện
- Các chương trình người dùng muốn được thực thi cần phải qua một số bước trong đó có bước gán địa chỉ cho các câu lệnh/ dữ liệu.

Gán bộ nhớ cho các câu lệnh và dữ liệu

- Việc gán địa chỉ cho các câu lệnh và dữ liệu được thực hiện tại các thời điểm
  - Biên dịch (compile time) – nếu vị trí trong bộ nhớ đã được biết trước – sinh ra mã tuyệt đối (absolute code); cần phải được biên dịch lại nếu vị trí bắt đầu bị thay đổi
  - Lúc tải (loading time) – phải sinh ra mã có thể định vị lại (relocatable code) – nếu vị trí trong bộ nhớ không được biết trước
    - Mã có thể định vị lại “14 bytes kể từ đầu module”
  - Lúc thực thi (run time)– gán địa chỉ được trì hoãn cho đến khi thực thi nếu tiến trình có thể thay đổi, từ đoạn bộ nhớ này đến đoạn bộ nhớ khác trong khi thực thi.
    - Yêu cầu phần cứng hỗ trợ cho các ánh xạ địa chỉ (thanh ghi cơ sở, thanh ghi giới hạn)

Các bước xử lý của tiến trình người dùng



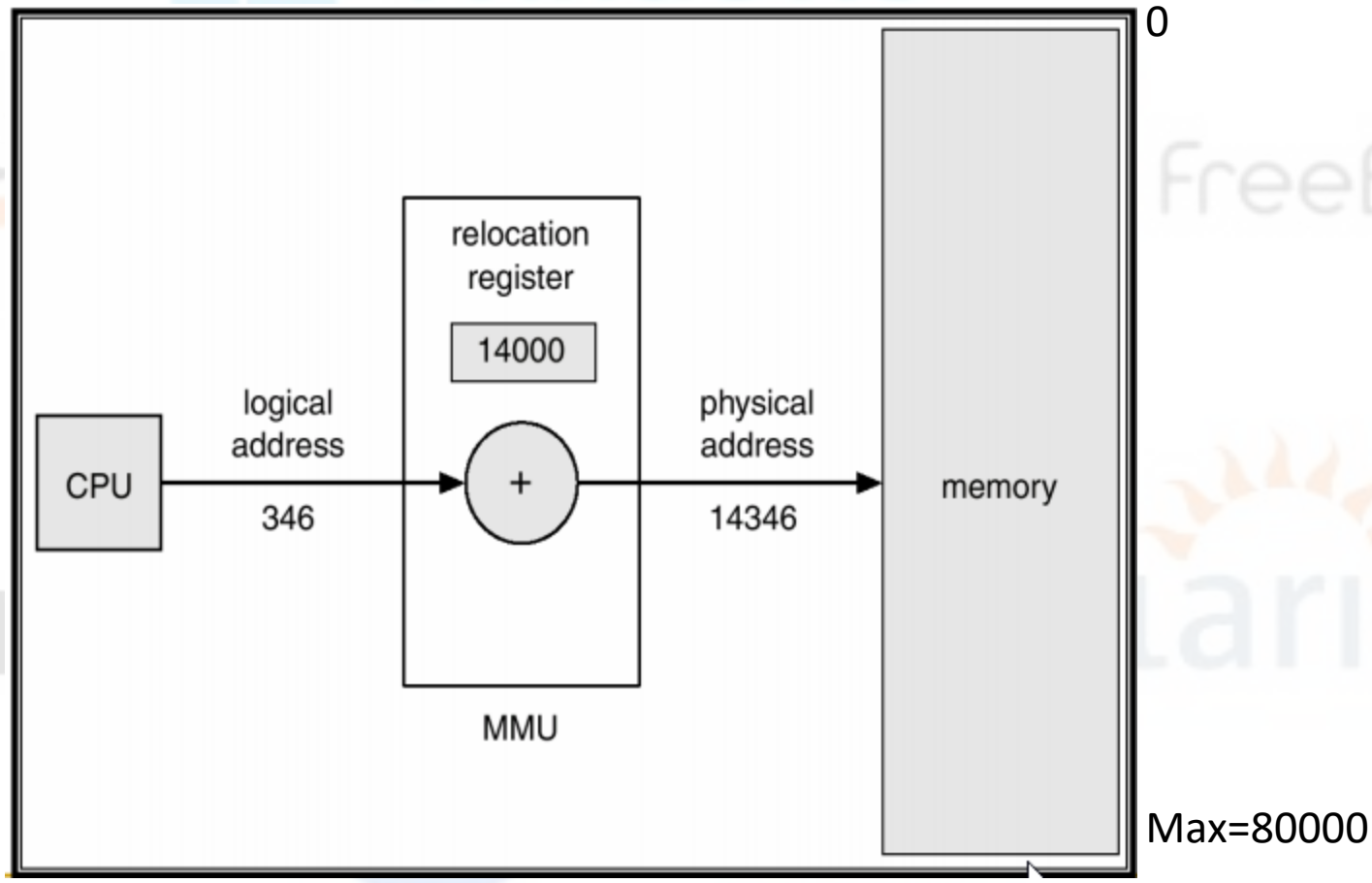
Không gian địa chỉ vật lý và không gian địa chỉ luận lý

- Khái niệm không gian địa chỉ luận lý gắn với không gian địa chỉ vật lý là trung tâm của các kĩ thuật quản lý bộ nhớ
  - Các địa chỉ luận lý – được CPU sinh ra; còn được gọi là địa chỉ ảo
  - Địa chỉ vật lý – địa chỉ thật trong bộ nhớ, được đơn vị quản lý bộ nhớ thấy
- Như nhau trong lược đồ gán địa chỉ lúc biên dịch, tải
- Khác nhau trong lược đồ gán địa chỉ lúc thực thi

Đơn vị quản lý bộ nhớ (MMU)

- Thiết bị phần cứng thực hiện việc ánh xạ địa chỉ ảo đến địa chỉ vật lý
- Ví dụ về 1 lược đồ MMU đơn giản
  - Giá trị thanh ghi định vị lại (relocation register) được cộng vào cho mỗi địa chỉ được sinh ra bởi tiến trình người dùng tại thời điểm nó tải vào bộ nhớ.
- Chương trình người dùng làm việc với các địa chỉ luận lý; nó không bao giờ thấy địa chỉ vật lý

Gán địa chỉ động với thanh ghi relocation



Tải động vào bộ nhớ

- Các phương thức không được tải vào bộ nhớ cho tới khi nó được gọi
- Tận dụng không gian bộ nhớ tốt hơn
  - Phương thức không được sử dụng sẽ không bao giờ được tải
- Hữu ích khi cần lượng mã lớn để xử lý các trường hợp không thường xuyên
- Không cần phải có sự hỗ trợ đặc biệt của hệ điều hành trong thiết kế chương trình

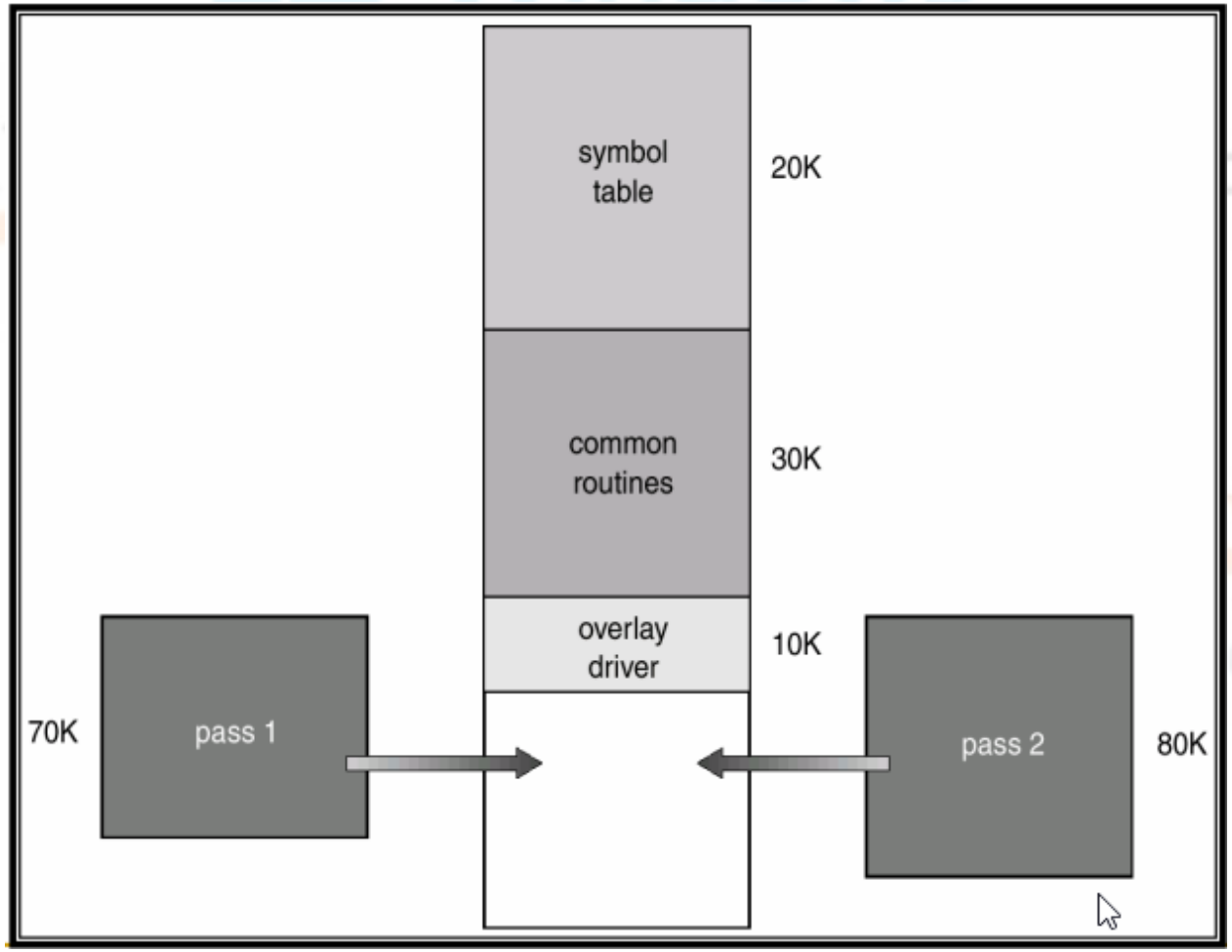
Liên kết động

- Việc liên kết sẽ bị trì hoãn đến thời gian thực thi
- Các đoạn mã nhỏ, gọi là stub, được sử dụng để xác định thủ tục thư viện trong vùng bộ nhớ thích hợp.
- Stub được thay thế bởi địa chỉ vật lý của routine (thủ tục) và thực thi routine
- Hệ điều hành cần phải kiểm tra xem liệu phương thức có nằm trong địa chỉ bộ nhớ của tiến trình
- Liên kết động rất hữu hiệu cho các thư viện

Overlay

- Chỉ giữ trong bộ nhớ những câu lệnh và dữ liệu cần trong bất cứ thời điểm nào
- Cần khi tiến trình lớn hơn kích cỡ bộ nhớ được cấp cho nó
- Được thực thi bởi người dùng, không cần sự hỗ trợ đặc biệt từ hệ điều hành, thiết kế lập trình của cấu trúc overlay tương đối phức tạp

Tải chồng (Overlay)



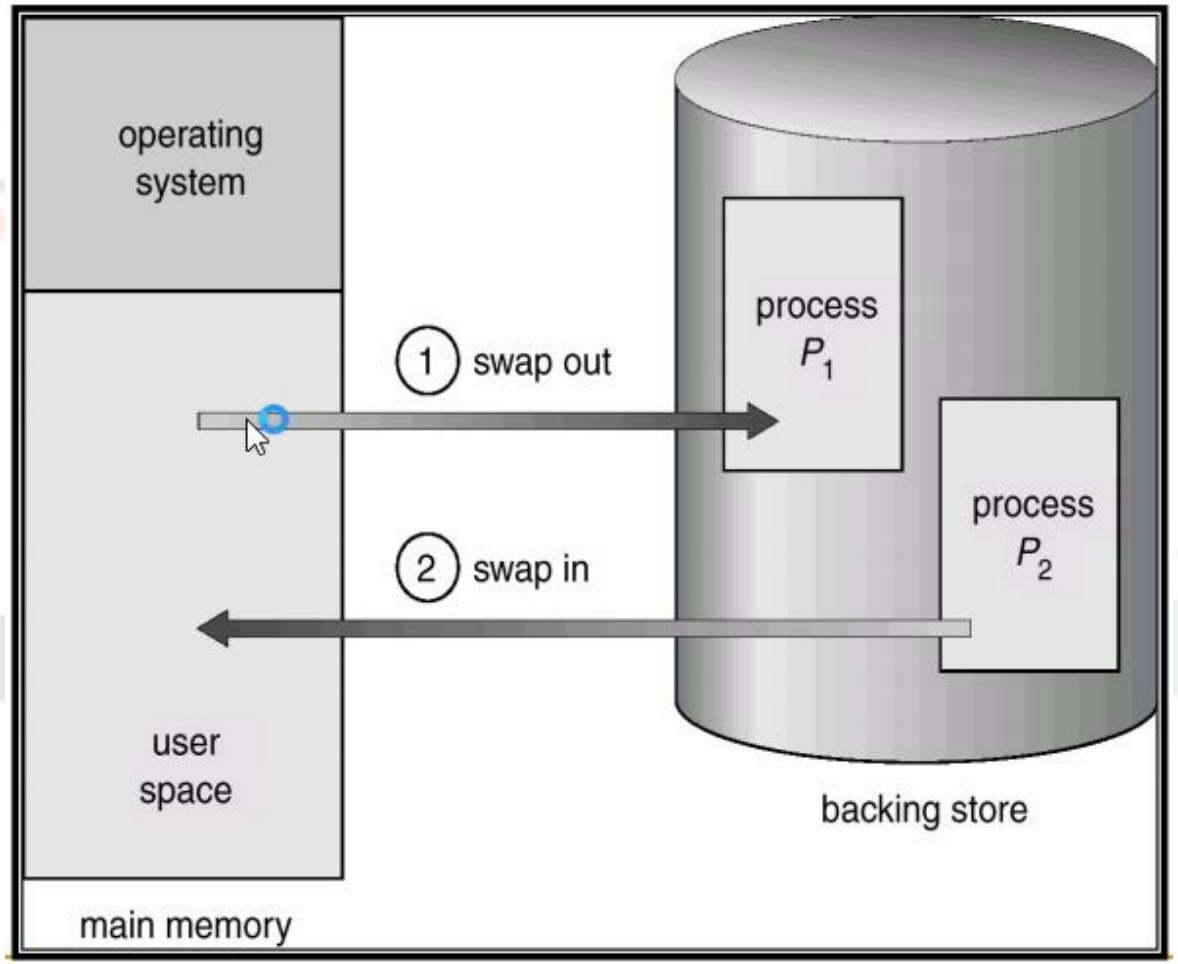
Tráo đổi (Swapping)

- Một tiến trình có thể bị swapped tạm ra bộ lưu trữ nền, sau đó được mang trở lại bộ nhớ để thực thi tiếp
- Bộ lưu trữ nền – đĩa tốc độ nhanh, đủ lớn để lưu trữ phiên bản của tất cả ảnh bộ nhớ cho tất cả người dùng; phải cung cấp khả năng truy cập trực tiếp đến các ảnh bộ nhớ này.
- Roll out, roll in – biến thể swapping được sử dụng trong thuật toán lập lịch có ưu tiên; tiến trình có độ ưu tiên thấp nhất bị swap ra cho phép tiến trình có độ ưu tiên cao nhất được tải vào và thực thi.

...Swapping

- Một trong những giai đoạn quan trọng trong thời gian swap là thời gian chuyển đổi ngữ cảnh
  - Tổng thời gian chuyển giao tỉ lệ với tổng dung lượng bộ nhớ bị swap.
- Ta có thể thấy nhiều phiên bản biến thể trên rất nhiều hệ thống, ví dụ UNIX, Linux, Windows.

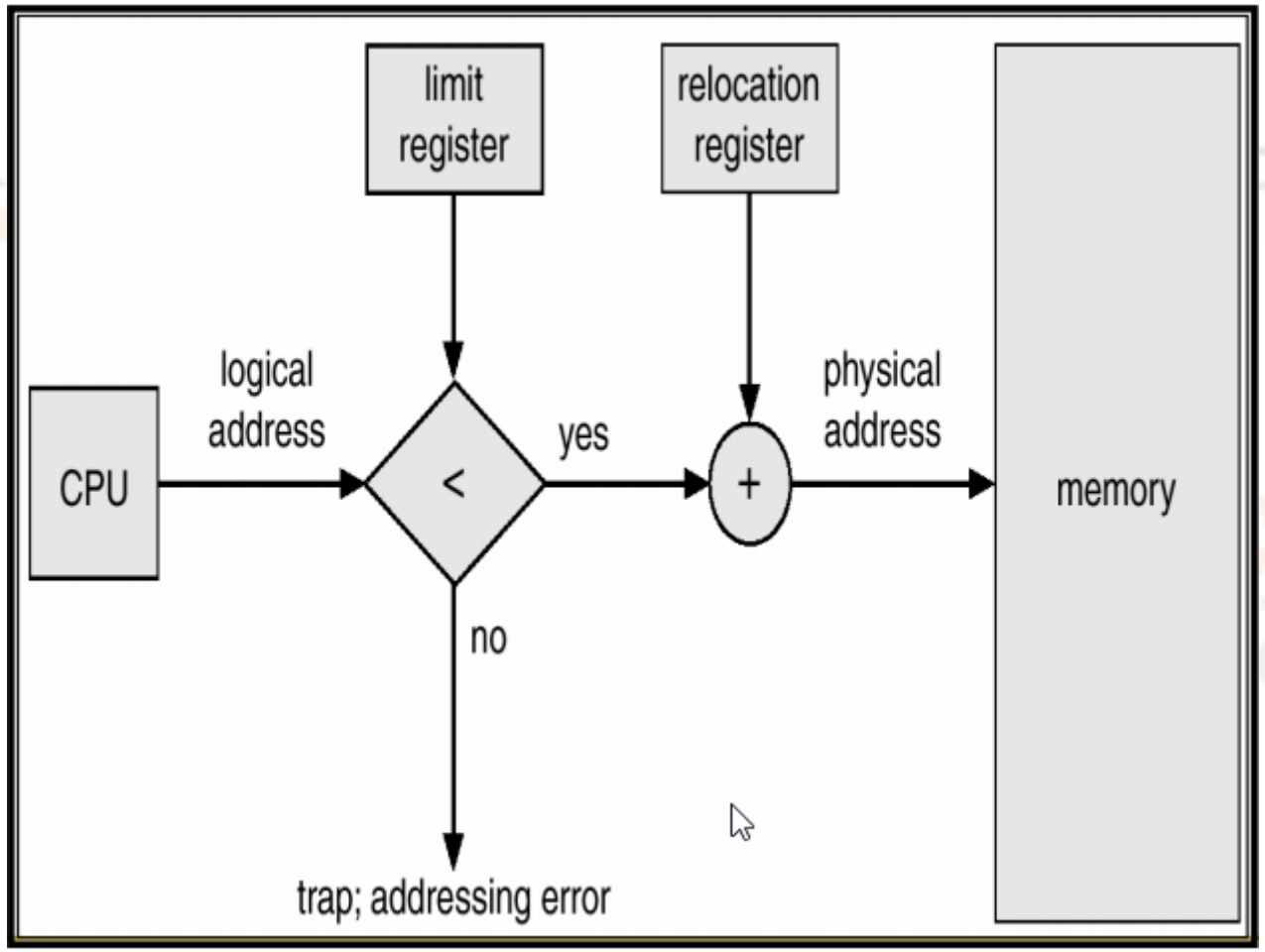
Lược đồ Swapping



Phân phối bộ nhớ liên tục

- Bộ nhớ chính thường được chia thành hai phần
  - Phần lưu trú hệ điều hành, thường được tổ chức trong vùng bộ nhớ thấp (địa chỉ thấp) với vector ngắt.
  - Các tiến trình người dùng, thường được tổ chức trong vùng bộ nhớ cao.
- Bảo vệ
  - Lược đồ thanh ghi định vị lại (relocation register) cho việc bảo vệ các tiến trình người dùng.
  - Thanh ghi định vị lại chứa giá trị của địa chỉ vật lý nhỏ nhất; thanh ghi giới hạn chứa các giá trị từ miền địa chỉ luận lý – các địa chỉ luận lý phải có giá trị nhỏ hơn giá trị của thanh ghi giới hạn.

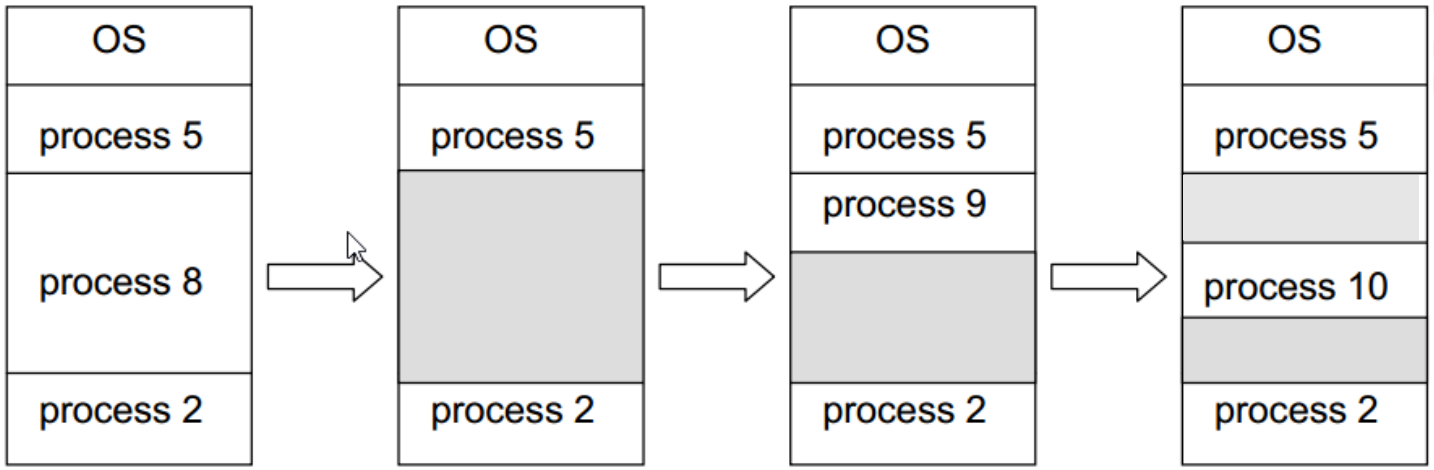
Hỗ trợ phần cứng cho các thanh ghi relocation và thanh ghi limit



... Phân phối liên tục

- Phân phối đa phân đoạn
  - Lỗ hổng – khối bộ nhớ rỗi; các lỗ hổng với những kích cỡ khác nhau nằm rải rác trong bộ nhớ.
    - Khi một tiến trình cần tải vào bộ nhớ, nó được phân phối vùng bộ nhớ từ lỗ hổng đủ lớn chứa nó.
  - Hệ điều hành quản lý thông tin về:

a) các phân đoạn đã được phân phối b) Các phân đoạn rỗi (lỗ hổng)



Bài toán phân phối bộ nhớ động

- Làm thế nào để phân phối tiến trình có kích cỡ n vào một danh sách các lỗ hổng còn rỗi
  - First-fit: tìm lỗ hổng đầu tiên đủ lớn
  - Best-fit: tìm lỗ hổng bé nhất, đủ lớn
    - Tìm kiếm trên toàn bộ danh sách các lỗ hổng (trừ phi các lỗ hổng được sắp xếp theo kích cỡ)
    - Sinh ra phần thừa nhỏ nhất
  - Worst-fit: tìm lỗ hổng lớn nhất
    - Cũng phải tìm kiếm
    - Sinh ra phần thừa lớn nhất
- First-fit và best-fit tốt hơn chiến lược worst-fit trên quan điểm tốc độ và sự tận dụng bộ nhớ

Sự phân mảnh

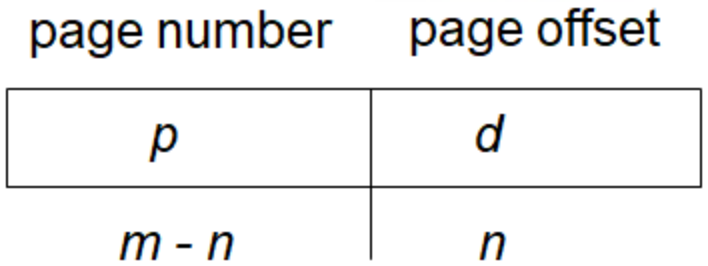
- **Phân mảnh ngoài** – tổng không gian bộ nhớ có thể đáp ứng yêu cầu, nhưng không liên tục
- **Phân mảnh trong** – bộ nhớ được phân phối có thể lớn hơn một chút so với yêu cầu; sự khác biệt về kích cỡ này là bên trong một phân đoạn, và không được sử dụng
- Làm giảm phân mảnh ngoài bằng kết khối
  - Xáo các nội dung bộ nhớ để đặt tất cả vùng bộ nhớ rỗi cạnh nhau tạo thành một khối lớn
  - Kết khối chỉ thích hợp khi việc phân đoạn lại là động và được thực hiện tại lúc thực thi

Phân trang (paging)

- Cho phép không gian địa chỉ luận lý của một tiến trình có thể không liên tục;
- Chia bộ nhớ vật lý thành các khối có kích cỡ cố định gọi là khung (frame) (kích cỡ là lũy thừa của 2, khoảng từ 512 bytes đến 8192 bytes).
- Chia bộ nhớ luận lý thành các khối cùng kích cỡ, gọi là trang (page).
- Lưu lại tất cả các frame rỗi.
- Để thực thi một chương trình có n page, cần tìm n frame rỗi và tải chương trình vào.
- Thiết lập một bảng trang (page table) để chuyển đổi địa chỉ luận lý thành địa chỉ vật lý.
- Có sự phân mảnh trong

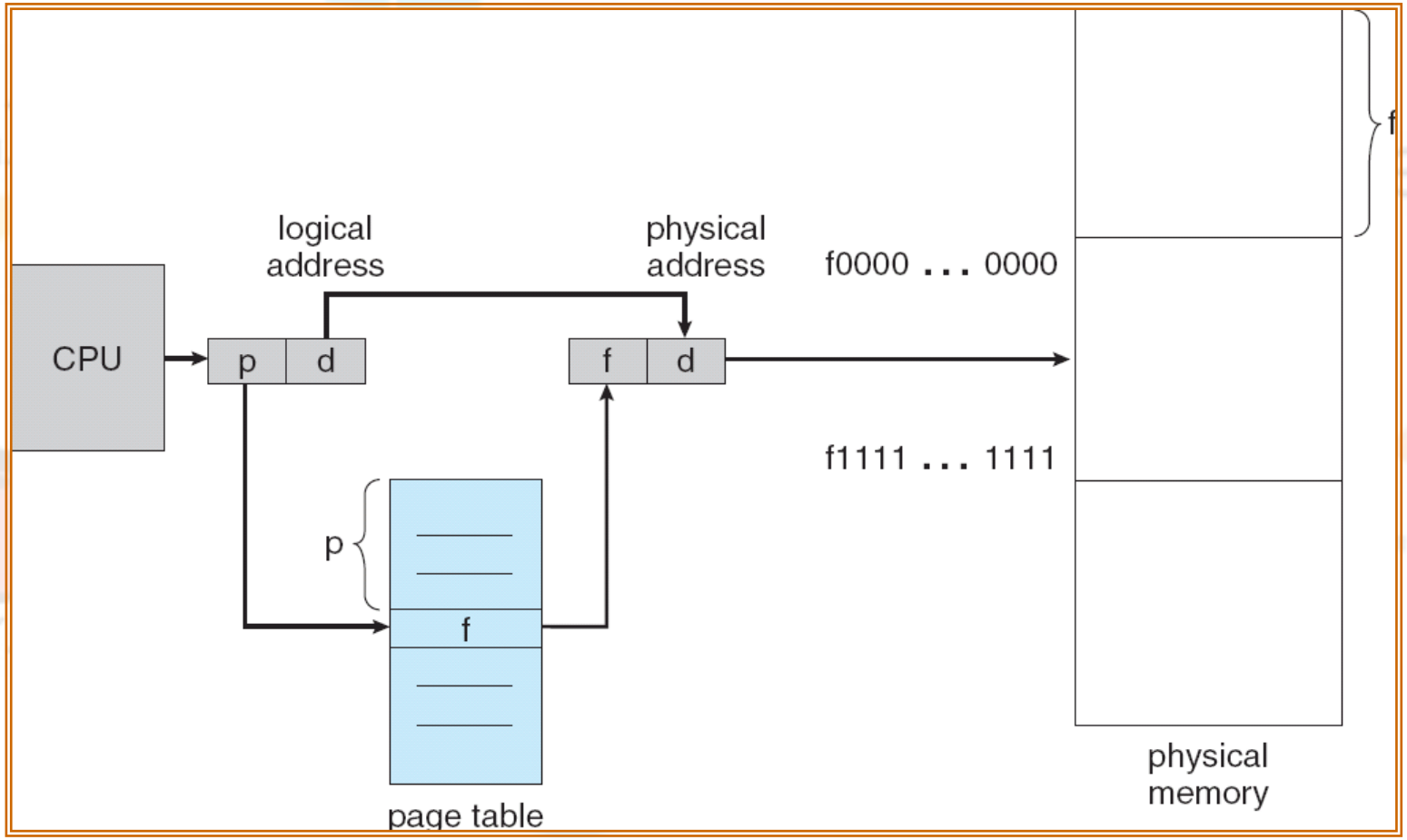
Lược đồ dịch địa chỉ

- Địa chỉ sinh bởi CPU được chia thành hai phần:
  - Page number (p) – được sử dụng như là một chỉ số trong bảng trang, chứa địa chỉ cơ sở (bắt đầu) của mỗi trang trong bộ nhớ vật lý
  - Page offset (d) – được kết hợp với địa chỉ cơ sở để xác định địa chỉ vật lý được gửi cho đơn vị bộ nhớ. (offset – phần bù, có thể coi như độ lệch so với đầu trang)

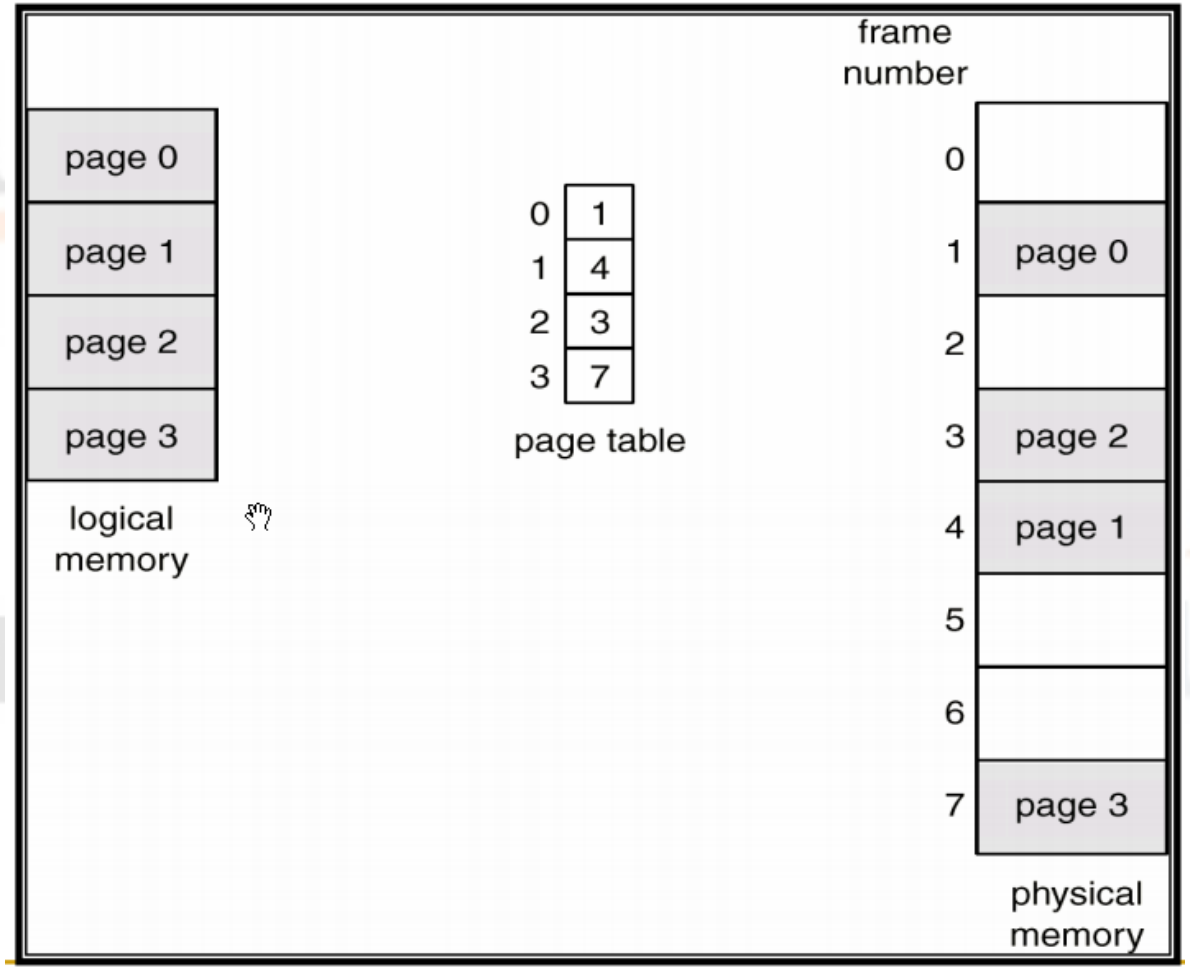


  - Cho không gian địa chỉ luận lý kích thước 2^m và kích thước trang 2^n (số trang = $2^m/2^n$)
    - Ví dụ: không gian địa chỉ 20MB = 2^{20}B, mỗi trang 512B = 2^9B, số trang là $2^{20}/2^9=2^{11}$, số hiệu trang từ 0 đến 2^{11}-1, d nhận giá trị từ 0 đến 511

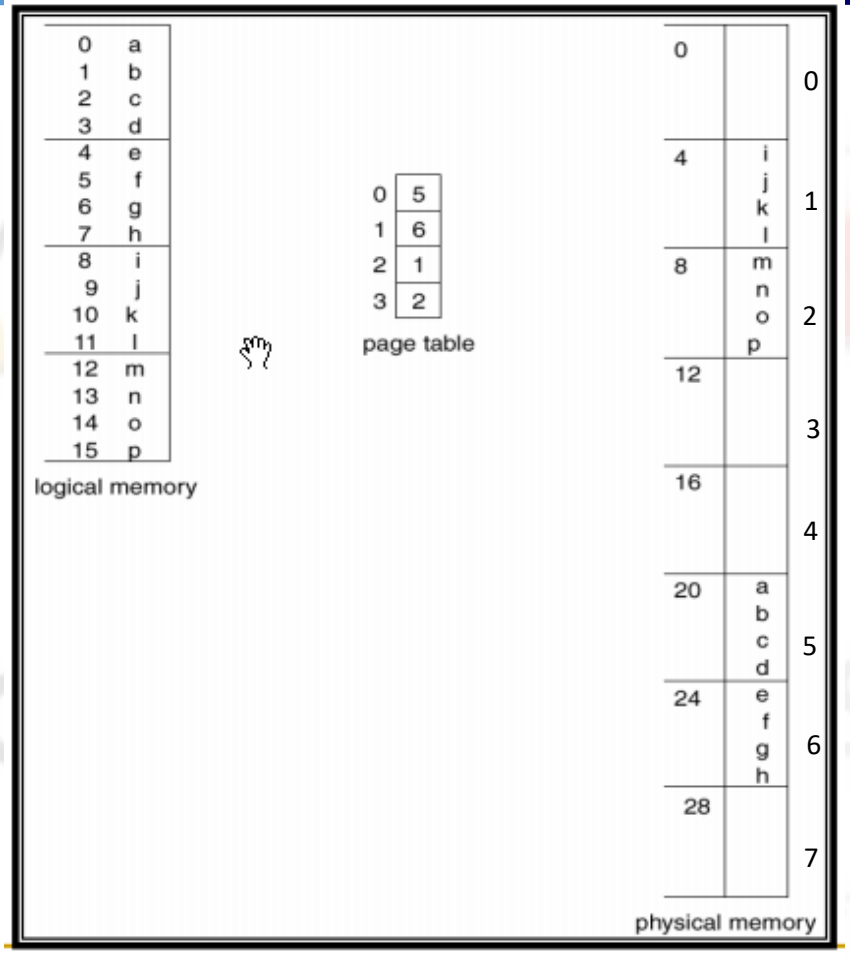
Phần cứng phân trang



Ví dụ về phân trang



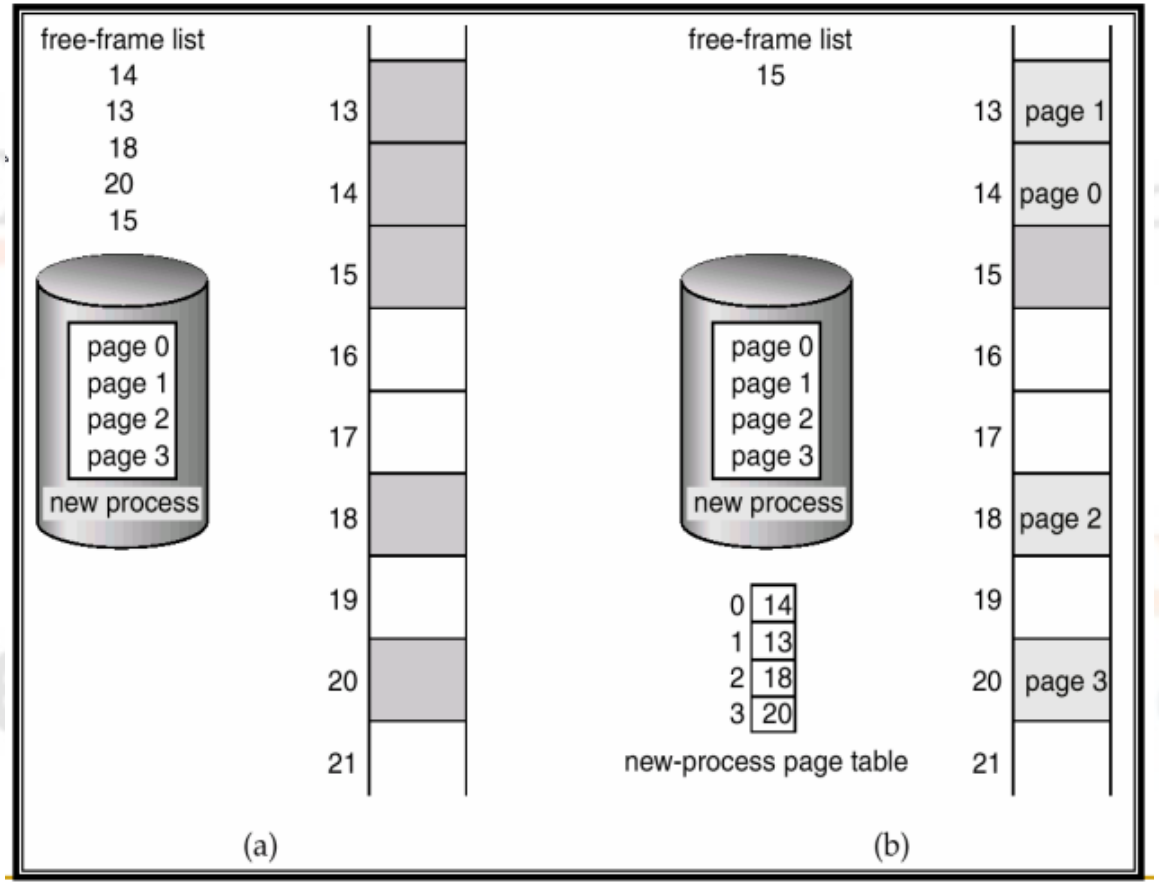
...Ví dụ về phân trang



Địa chỉ vật lý=số hiệu frame*framesize+độ lệch trong trang
Ví dụ kí tự *h* nằm trong page 1, tra bảng trang thì page 1 được cấp frame 6, độ lệch của h trong page 1 là 3
Địa chỉ vật lý của h = 6*4+3 =27

Địa chỉ luận lý=số hiệu page*pagesize+độ lệch trong khung
Ví dụ kí tự *a* nằm trong frame 5, tra bảng trang thì frame 5 cấp cho page 0, độ lệch của a trong frame 5 là 0
Địa chỉ luận lý của a = 0*4+0 = 0

Các frame rỗi



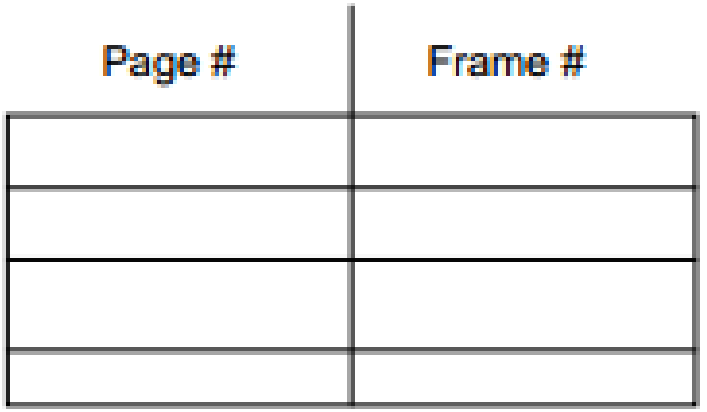
Trước khi phân phối Sau khi phân phối

Cài đặt bảng trang (page table)

- Bảng trang được lưu trữ trong bộ nhớ trong.
- Thanh ghi cơ sở bảng-trang (PTBR) trỏ đến bảng trang.
- Thanh ghi độ dài bảng trang (PTLR) chỉ kích cỡ của bảng trang.
- Trong lược đồ này, mọi truy cập đến dữ liệu và câu lệnh đòi hỏi hai lần truy cập bộ nhớ:
  - Một lần cho bảng trang và một lần cho dữ liệu/ câu lệnh.
- Vấn đề hai lần truy cập bộ nhớ này có thể được giải quyết bằng cách sử dụng một cache phần cứng tra cứu nhanh gọi là bộ nhớ kết hợp hay bộ đệm dịch (TLBs)

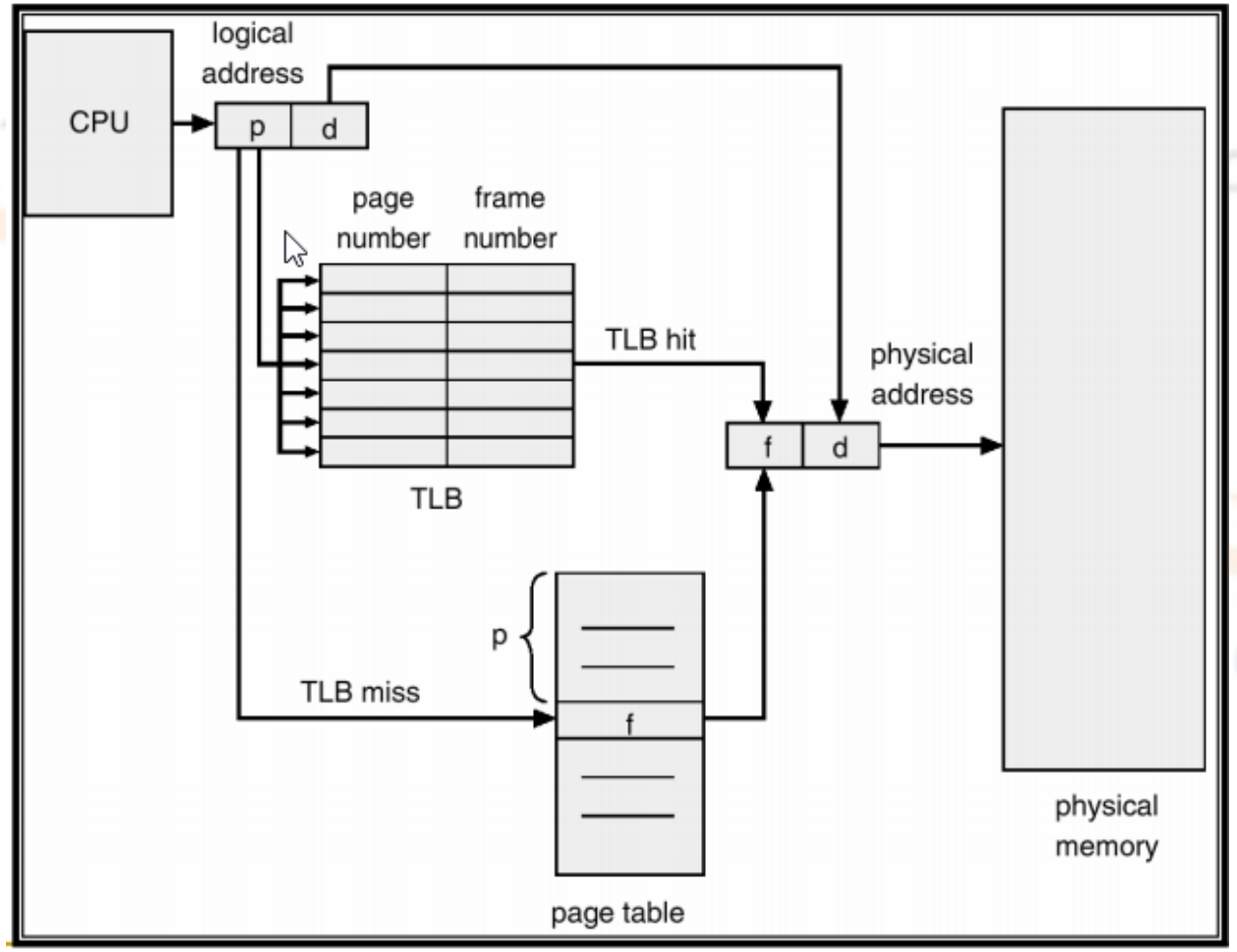
Bộ nhớ kết hợp

- Bộ nhớ kết hợp – tìm kiếm song song



- Dịch địa chỉ (p, d)
  - Nếu p là thanh ghi kết hợp, lấy frame # ra
  - Nếu không, lấy frame # từ bảng trang trong bộ nhớ

Phần cứng cho phân trang, TLB



Thời gian truy cập hiệu quả

- Tra cứu kết hợp = ε thời gian đơn vị
- Giả sử thời gian chu kỳ bộ nhớ là 1 micro giây
- Tỉ lệ trúng (hit ratio) – phần trăm thời gian mà một page number được tìm thấy trong các thanh ghi kết hợp; tỉ lệ liên quan đến số các thanh ghi kết hợp.
- Hit ratio = α
- **Thời gian truy cập hiệu quả** (EAT)

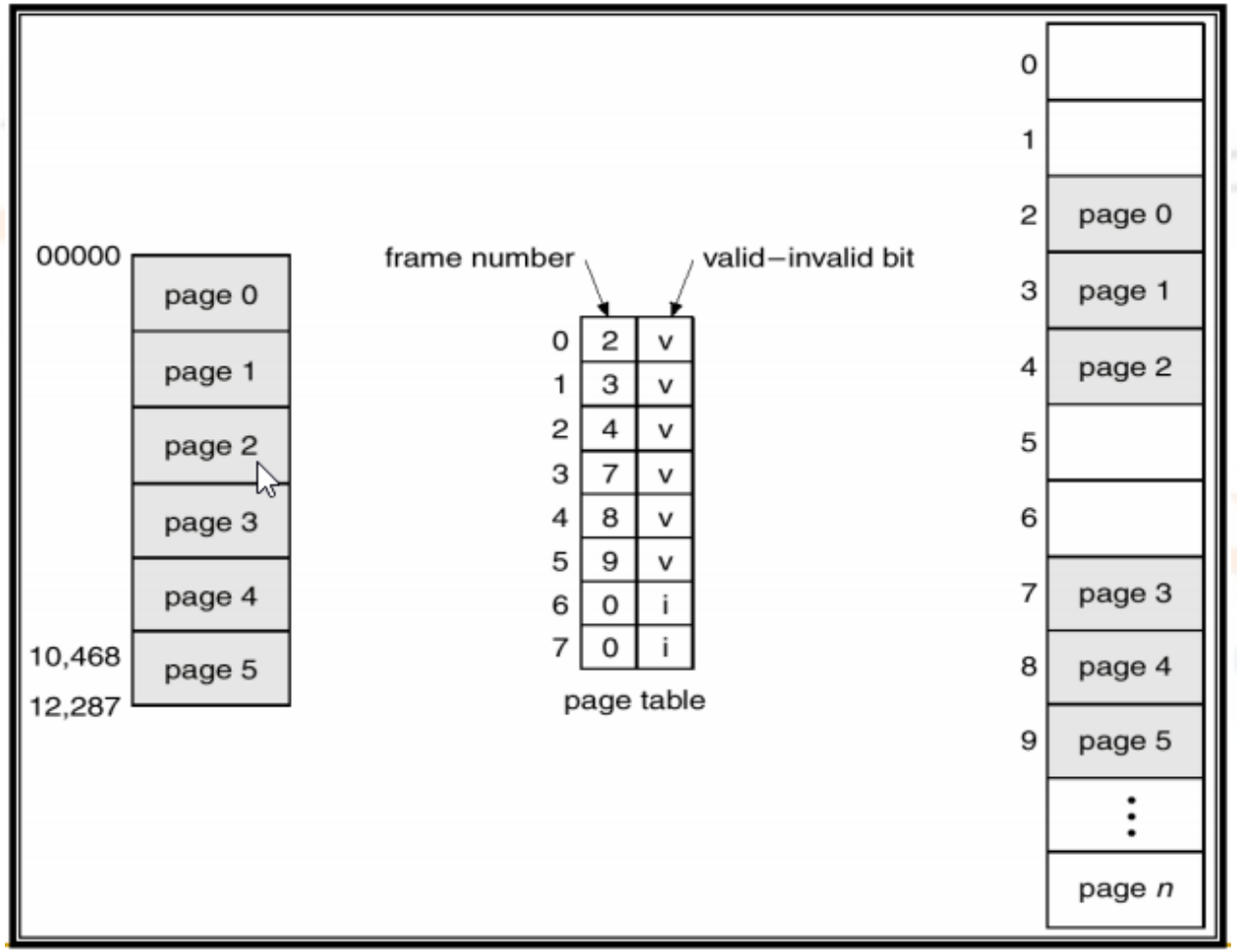
$$EAT = (1 + \varepsilon)\,\alpha + (2 + \varepsilon)(1 - \alpha)$$

$$= 2 + \varepsilon - \alpha$$

Bảo vệ bộ nhớ

- Bảo vệ bộ nhớ bằng cách kết hợp các bit bảo vệ với mỗi frame.
- Bit valid-invalid gắn với mỗi phần tử của bảng trang:
  - “valid” chỉ rằng trang liên kết trong một không gian địa chỉ luận lý, và là một trang hợp lệ.
  - “invalid” chỉ rằng trang không ở trong một không gian địa chỉ luận lý.

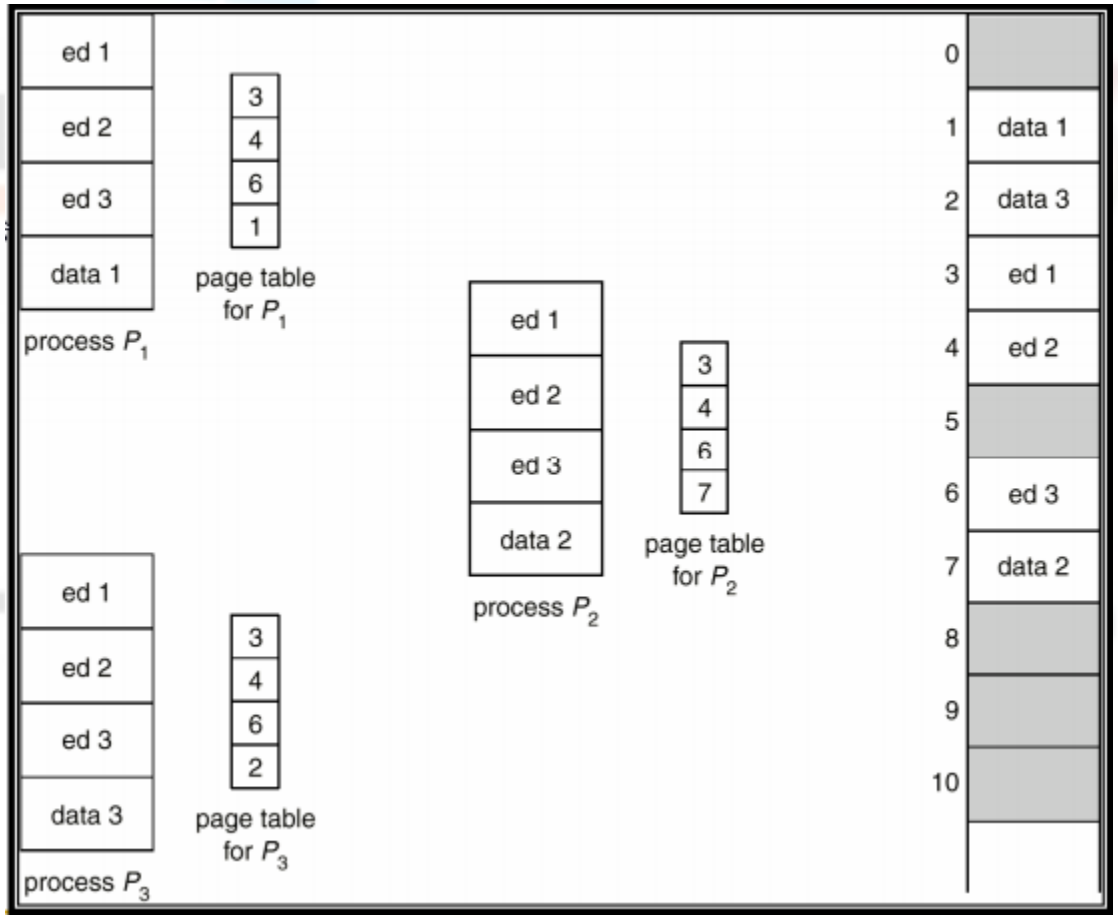
Bit valid và invalid trong bảng trang



Các trang dùng chung

- Mã chia sẻ
  - Một phiên bản mã chỉ đọc được chia sẻ giữa nhiều tiến trình (i.e., text editors, compilers, window systems).
  - Mã chia sẻ phải xuất hiện tại cùng một vị trí trong không gian địa chỉ luận lý của tất cả các tiến trình.
- Mã và dữ liệu riêng
  - Mỗi tiến trình lưu trữ một phiên bản riêng của mã và dữ liệu.
  - Các trang cho mã riêng và dữ liệu riêng có thể xuất hiện mọi nơi trong không gian địa chỉ luận lý.

Ví dụ về các trang chia sẻ



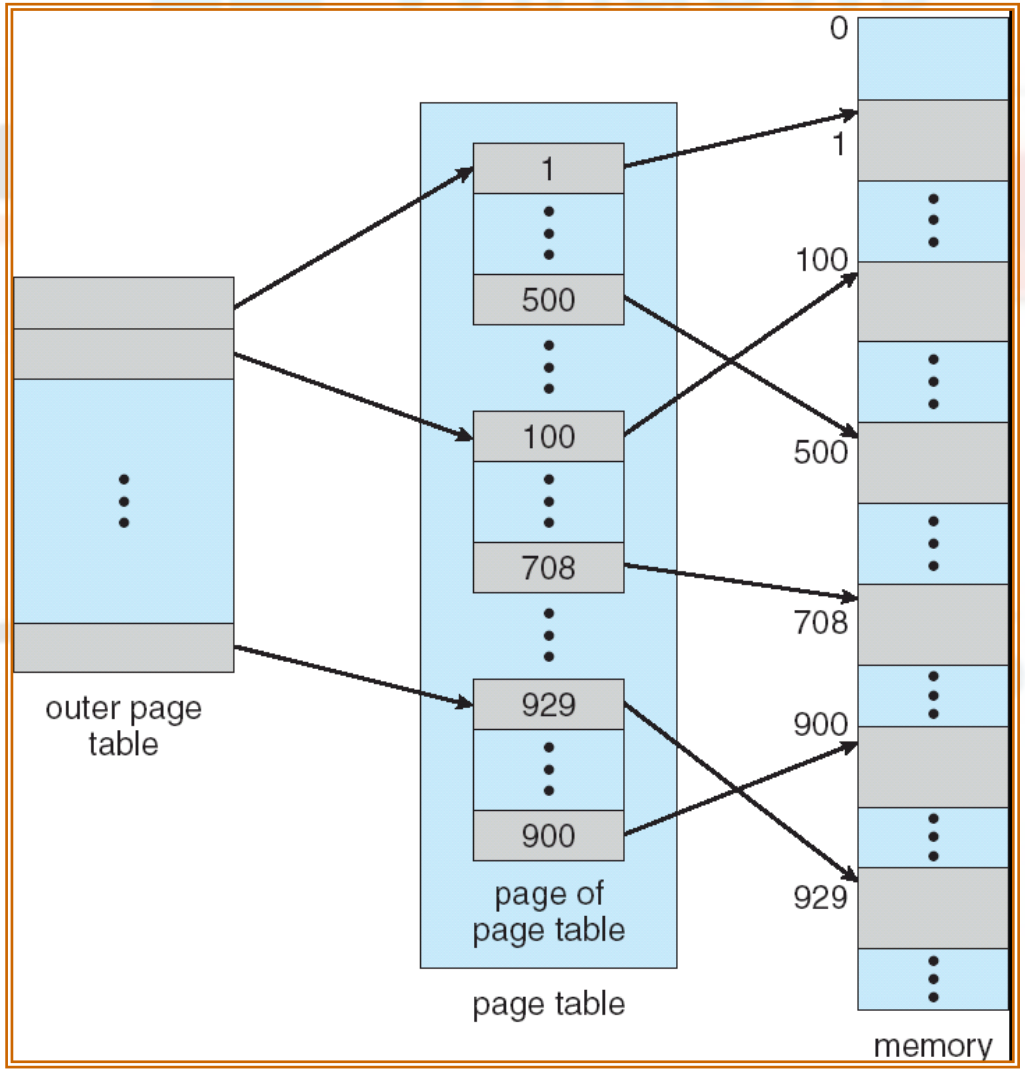
Cấu trúc bảng trang

- Phân trang phân cấp
- Các bảng trang băm
- Các bảng trang đánh chỉ số ngược

Các bảng trang phân cấp

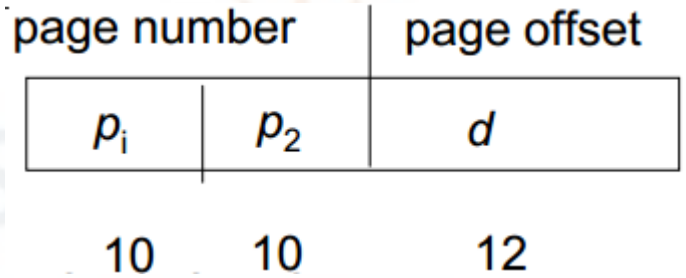
- Chia không gian địa chỉ vật lý thành nhiều bảng trang.
- Một kĩ thuật đơn giản là sử dụng bảng trang hai mức.

Lược đồ bảng trang hai mức



Ví dụ về bảng trang hai mức

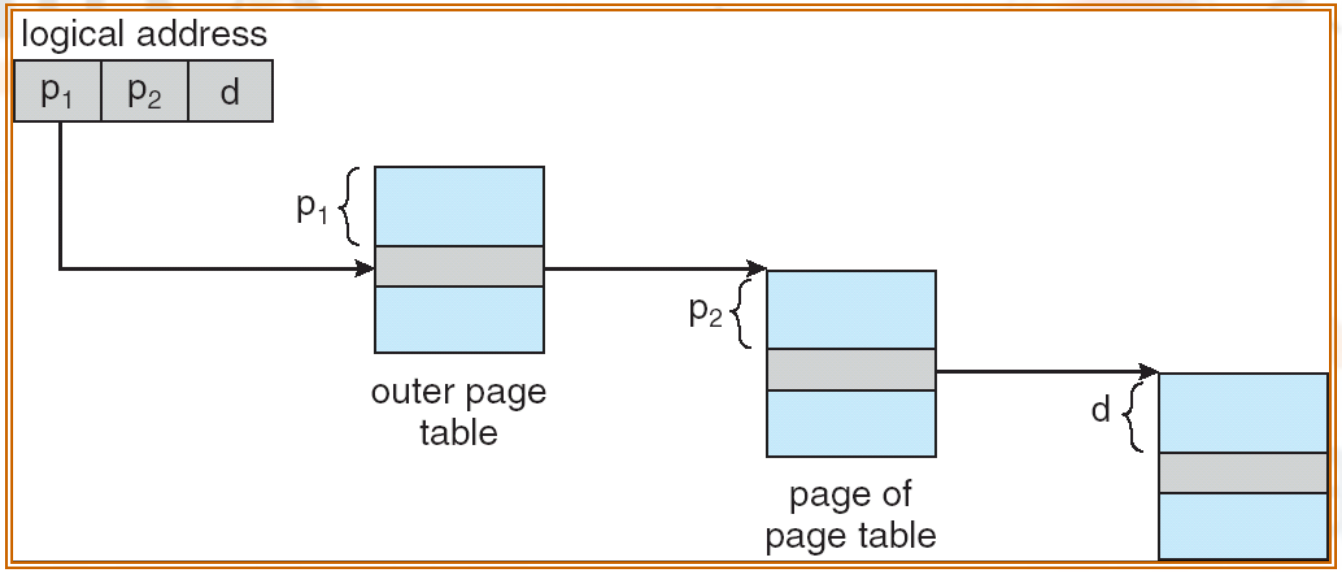
- Một địa chỉ luận lý (trên máy 32 bit với kích cỡ trang 4K) được chia thành:
  - Page number gồm 20 bits
  - Page offset gồm 12 bits
- Khi bảng trang được phân trang, page number được chia tiếp thành:
  - Một page number 10 bit.
  - Một page offset 10 bit.
- Như vậy, không gian địa chỉ luận lý sẽ như sau:



- ở đây p_i là một chỉ số của bảng page ngoài và p_2 là độ dịch chuyển trong trang của bảng trang ngoài.

Lược đồ dịch địa chỉ

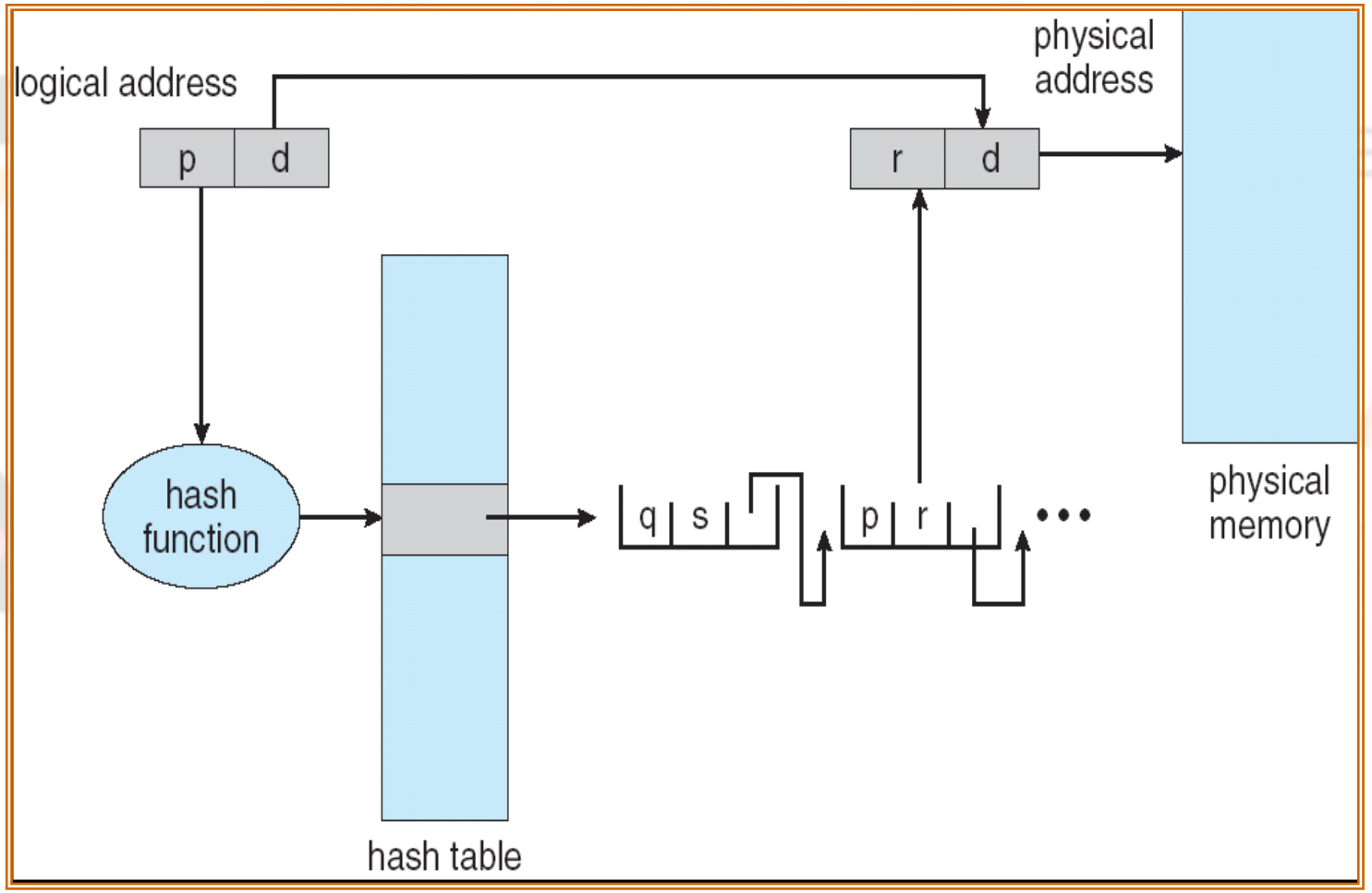
- Lược đồ dịch địa chỉ cho kiến trúc phân trang 32-bit hai mức



Các bảng trang băm

- Thông thường các không gian địa chỉ > 32 bit.
- Chỉ số trang ảo được băm vào một bảng trang. Bảng trang này chứa một chuỗi các phần tử được băm vào cùng một vị trí.
- Các chỉ số trang ảo được tìm kiếm trong chuỗi này. Nếu tìm thấy, frame vật lý tương ứng được lấy ra

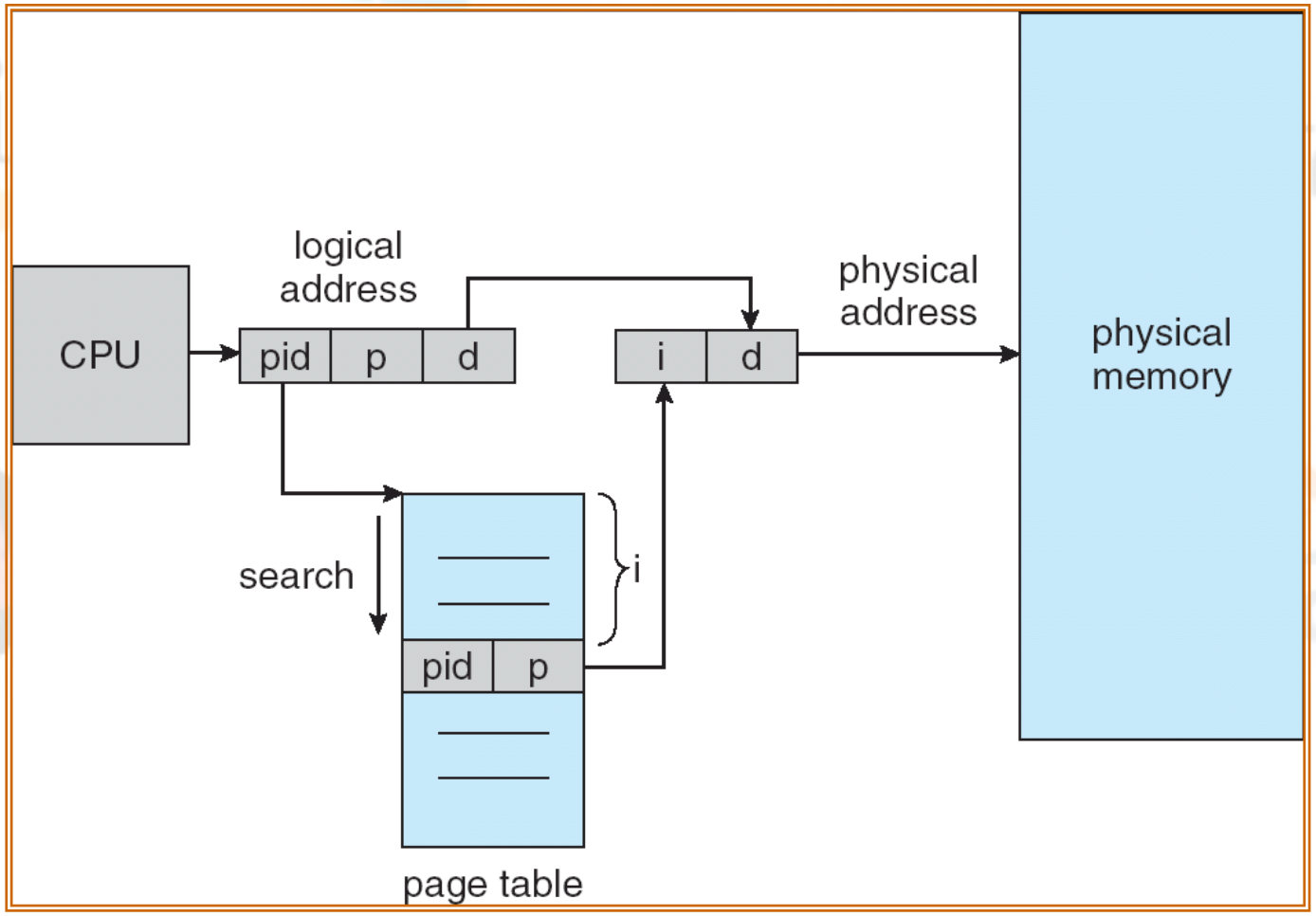
Bảng trang băm



Bảng trang ngược

- Mỗi phần tử tương ứng với một trang thật trong bộ nhớ.
- Mỗi phần tử gồm địa chỉ ảo của trang được lưu trong phần bộ nhớ thật, với thông tin về tiến trình chứa trang đó.
- Giảm bộ nhớ cần thiết để lưu trữ mỗi bảng trang, nhưng tăng thời gian cần thiết để tìm kiếm bảng khi một yêu cầu truy cập trang xuất hiện.
- Sử dụng trang băm để giới hạn tìm kiếm đến một, hoặc một vài phần tử của bảng trang.

Kiến trúc bảng trang ngược

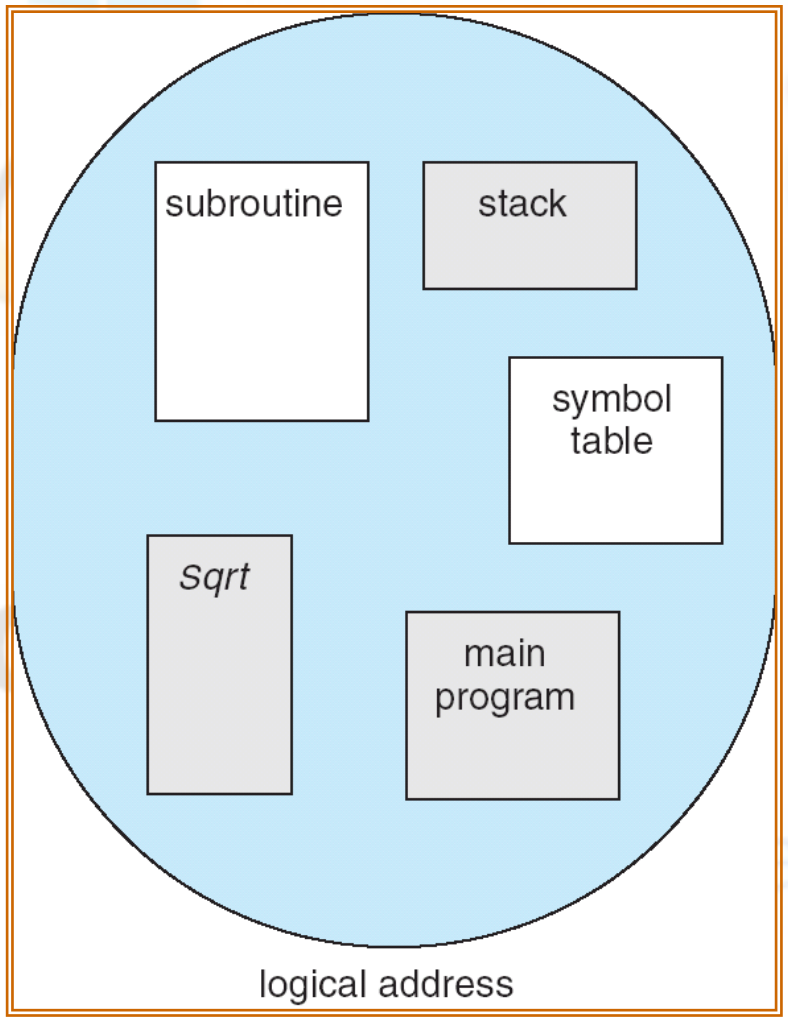


Phân đoạn (Segmentation)

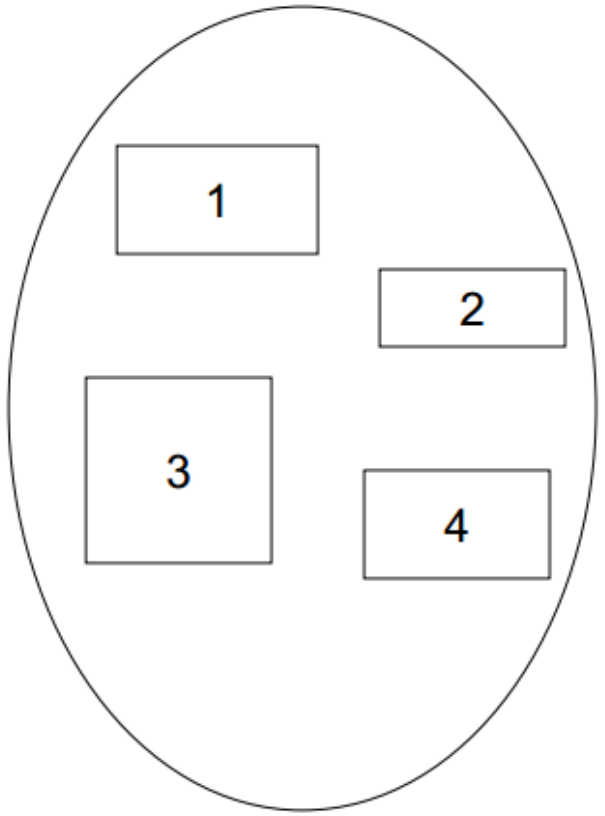
- Lược đồ quản lý bộ nhớ hỗ trợ quan điểm của người dùng về bộ nhớ:
  - Một chương trình là một tập các đoạn. Một đoạn là một đơn vị luận lý gồm có:

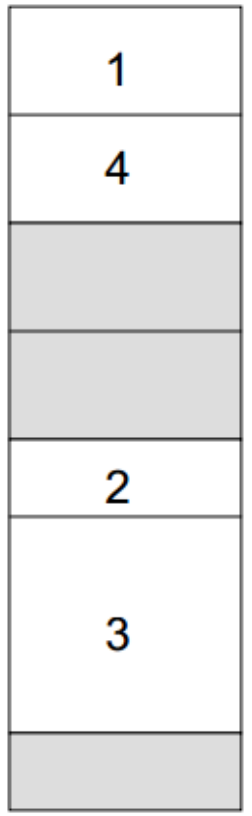
    main program,
    procedure,
    function,
    method,
    object,
    local variables, global variables,
    common block,
    stack,
    symbol table, arrays

Quan điểm người dùng của một chương trình



Quan điểm luận lý của segmentation





user space physical memory space

Kiến trúc phân đoạn

- Địa chỉ luận lý gồm hai phần: <segment-number, offset>,
- Bảng Segment– ánh xạ các địa chỉ vật lý hai chiều; mỗi phần tử của bảng có:
  - Base – chứa địa chỉ vật lý bắt đầu nơi mà các segment lưu trú trong bộ nhớ.
  - Limit – xác định kích cỡ của segment.
- Segment - table base register (STBR) trỏ đến bảng segment trong bộ nhớ.
- Segment - table length register (STLR) thể hiện số các segments được sử dụng bởi một chương trình;

segment number s là hợp lệ nếu s< STLR.

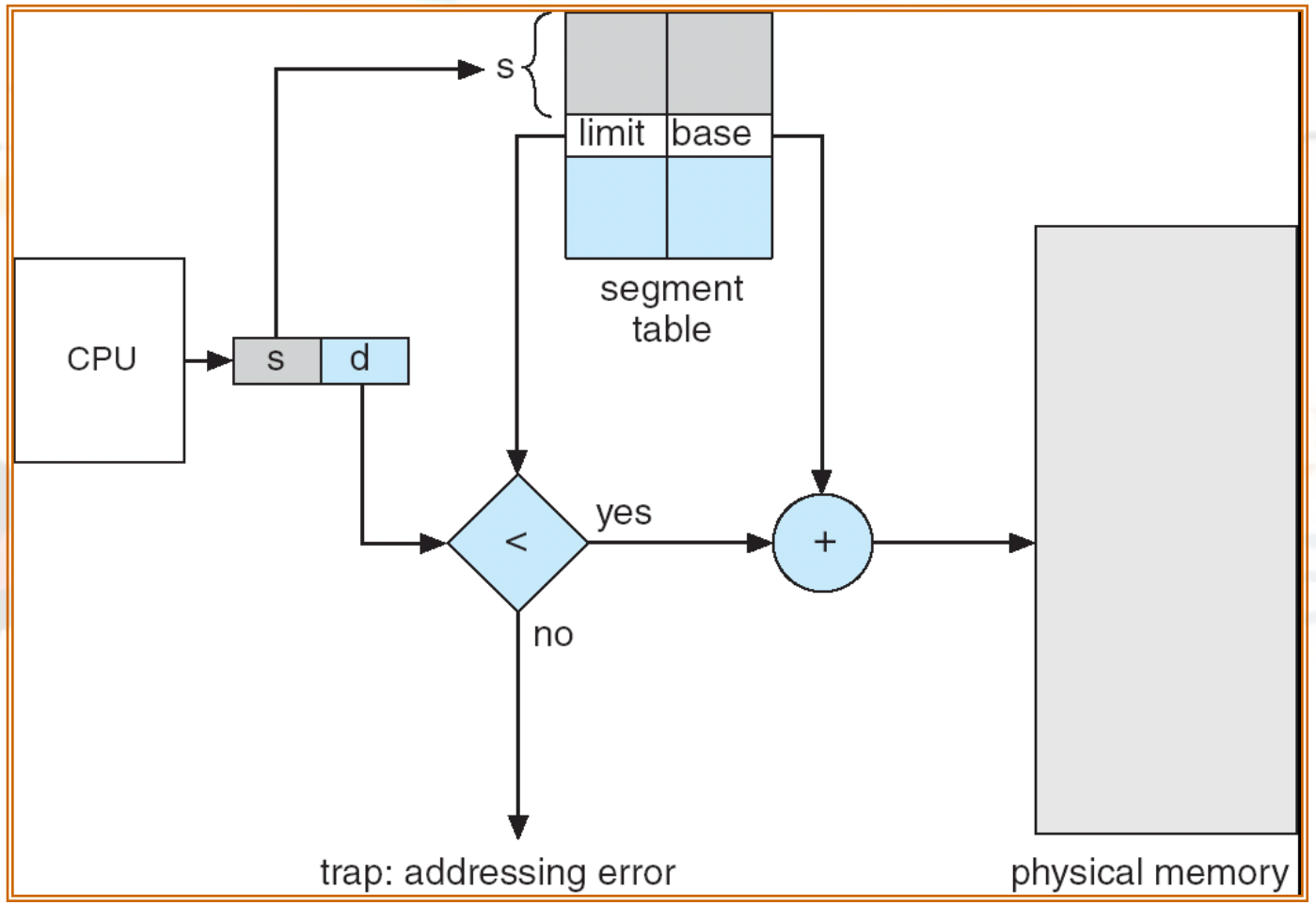
... Kiến trúc phân đoạn

- Xác định lại vị trí.
  - Động
  - Dùng bảng segment (tĩnh)
- Chia sẻ.
  - Các đoạn được chia sẻ.
  - Cùng segment number
- Phân phối.
  - First fit /Best fit
  - Phân mảnh ngoài

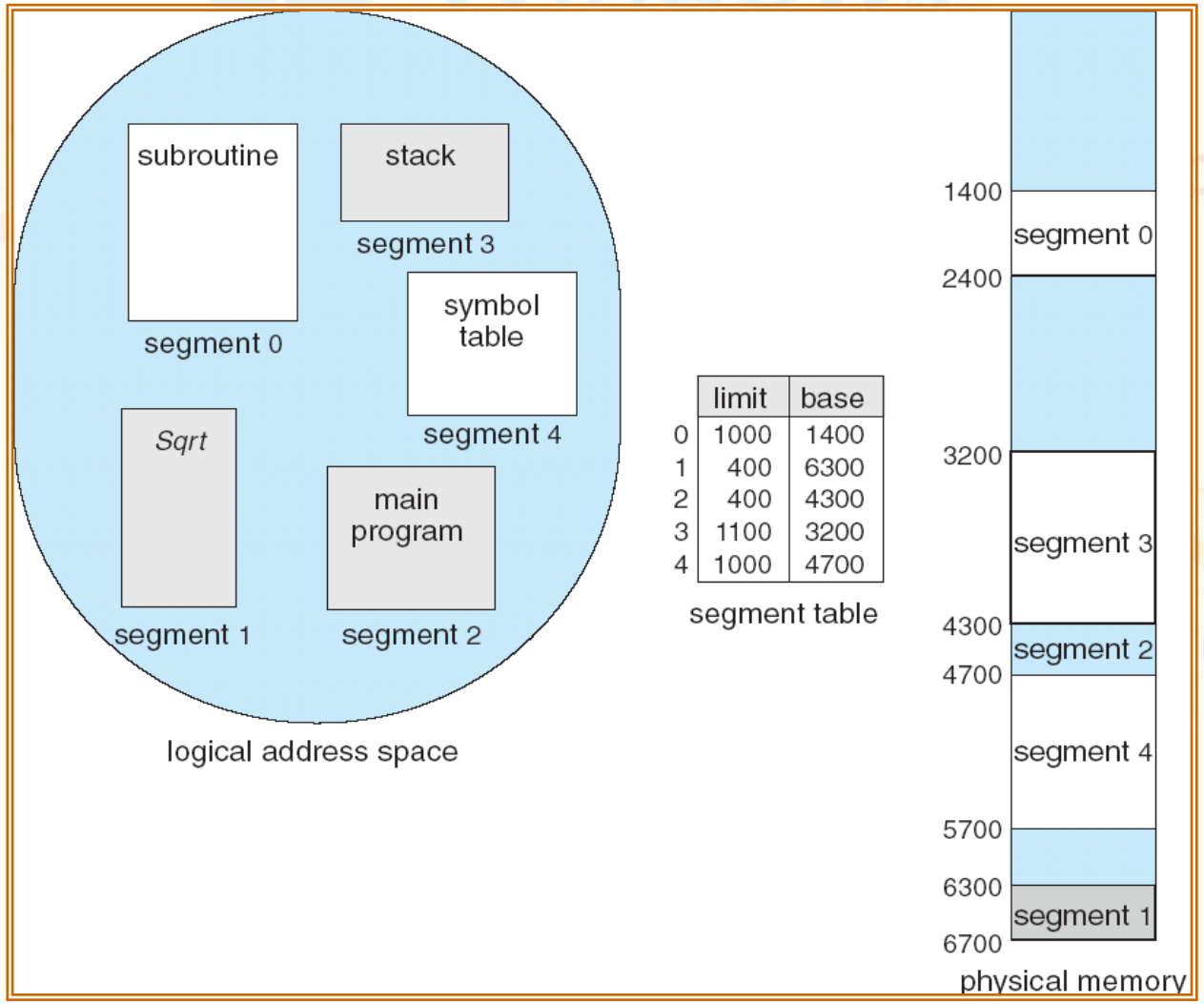
...Kiến trúc phân đoạn

- Bảo vệ. Một phần tử trong bảng segment liên kết với:
  - Bit hợp lệ = 0 ⇒segment không hợp lệ
  - Ưu tiên read/ write/ execute
- Các bits bảo vệ kết hợp với các segments; chia sẻ mã ở mức segment.
- Khi các segment thay đổi kích cỡ, phân phối bộ nhớ là phân phối động.
- Một ví dụ segmentation được cho trong hình vẽ sau

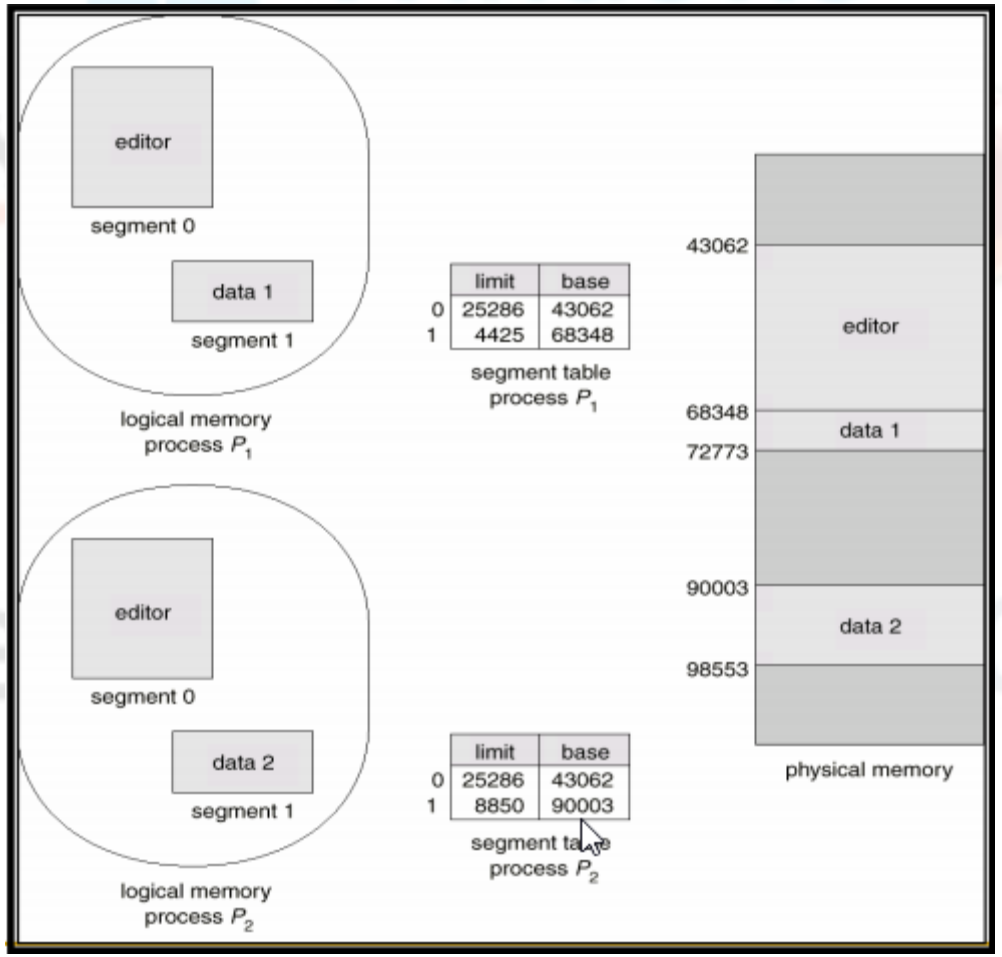
Phần cứng phân đoạn



Ví dụ về phân đoạn



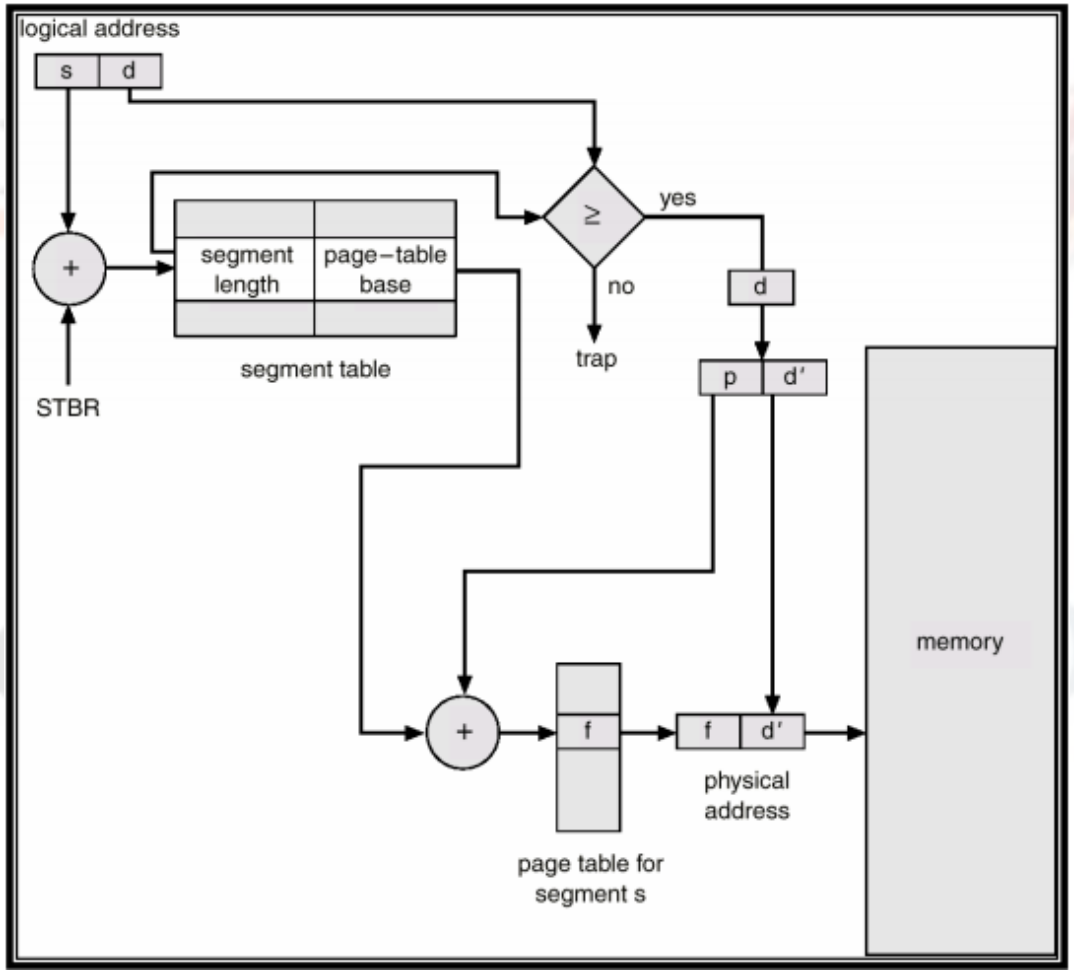
Chia sẻ các đoạn



Phân đoạn kết hợp với phân trang

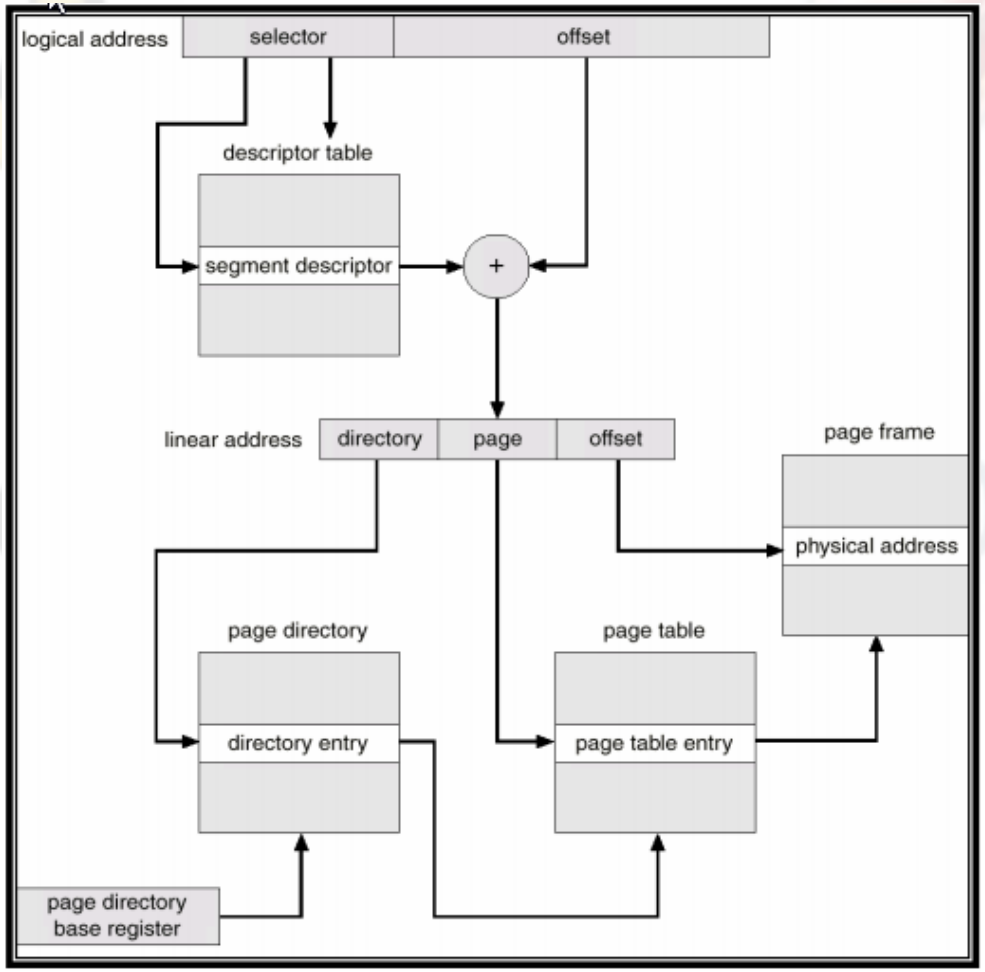
- Hệ thống MULTICS giải quyết bài toán phân mảnh ngoài bằng cách phân trang các segments.
- Giải pháp khác với segmentation thuần túy là phần tử bảng segment không chứa địa chỉ cơ sở của segment mà địa chỉ cơ sở của bảng trang cho segment này.

Lược đồ dịch địa chỉ MULTICS



Phân đoạn kết hợp với phân trang - Intel 386

- Như trong hình vẽ sau, Intel 386 sử dụng segmentation với paging cho việc quản lý bộ nhớ với lược đồ hai tầng phân trang




Question?