CT107. Hệ Điều Hành
Chương 7. Quản lý bộ nhớ

Giảng viên: Trần Công Án (tcan@cit.ctu.edu.vn)

Bộ môn Mạng máy tính & Truyền thông

Khoa Công Nghệ Thông Tin & Truyền Thông

Đại học Cần Thơ

Mục Tiêu

- Mô tả chi tiết các phương pháp tổ chức bộ nhớ.
- Giải thích các kỹ thuật quản lý bộ nhớ bao gồm phân trang và phân đoạn.
- Một số ví dụ thực tế về quản lý bộ nhớ: quản lý phân đoạn trong bộ xử lý Intel Pentium và quản lý địa chỉ bộ nhớ trong HĐH Linux.

Nội Dung

Giới Thiệu Bộ Nhớ

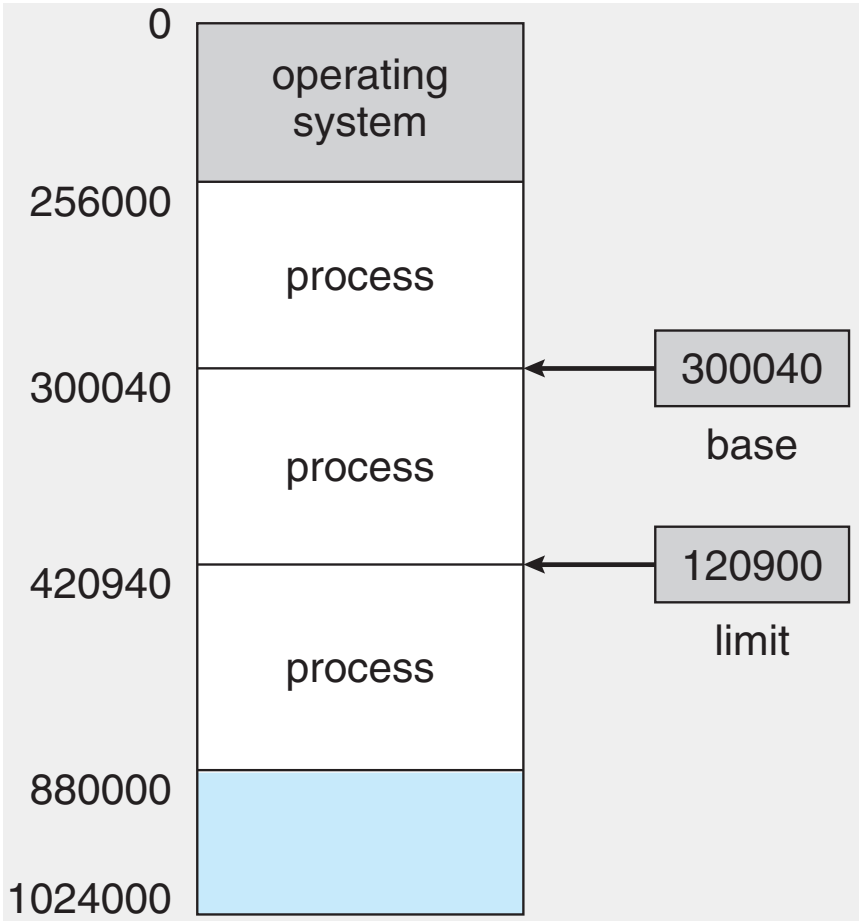
- CPU chỉ có thể truy xuất trực tiếp thanh ghi và bộ nhớ chính.

  $\Rightarrow$ Để thực thi một chương trình, đoạn mã của chương trình phải được tải vào trong bộ nhớ chính và đặt trong một tiến trình.
- Thanh ghi: một dạng bộ nhớ đặc biệt, đặt bên trong CPU và chỉ mất tối đa 1 chu kỳ CPU để truy xuất.
- Bộ nhớ chính: tốc độ truy xuất chậm hơn thanh ghi, đòi hỏi vài chu kỳ.
- Bộ nhớ cache: là bộ nhớ trung gian giữa thanh ghi và bộ nhớ chính, tốc độ truy xuất nhanh, chỉ chậm hơn thanh ghi.
- Việc bảo vệ bộ nhớ là cần thiết để đảm bảo thực thi đúng đắn của các tiến trình, đặc biệt trong môi trường đa nhiệm.

Thanh Ghi Nền & Thanh Ghi Giới Hạn

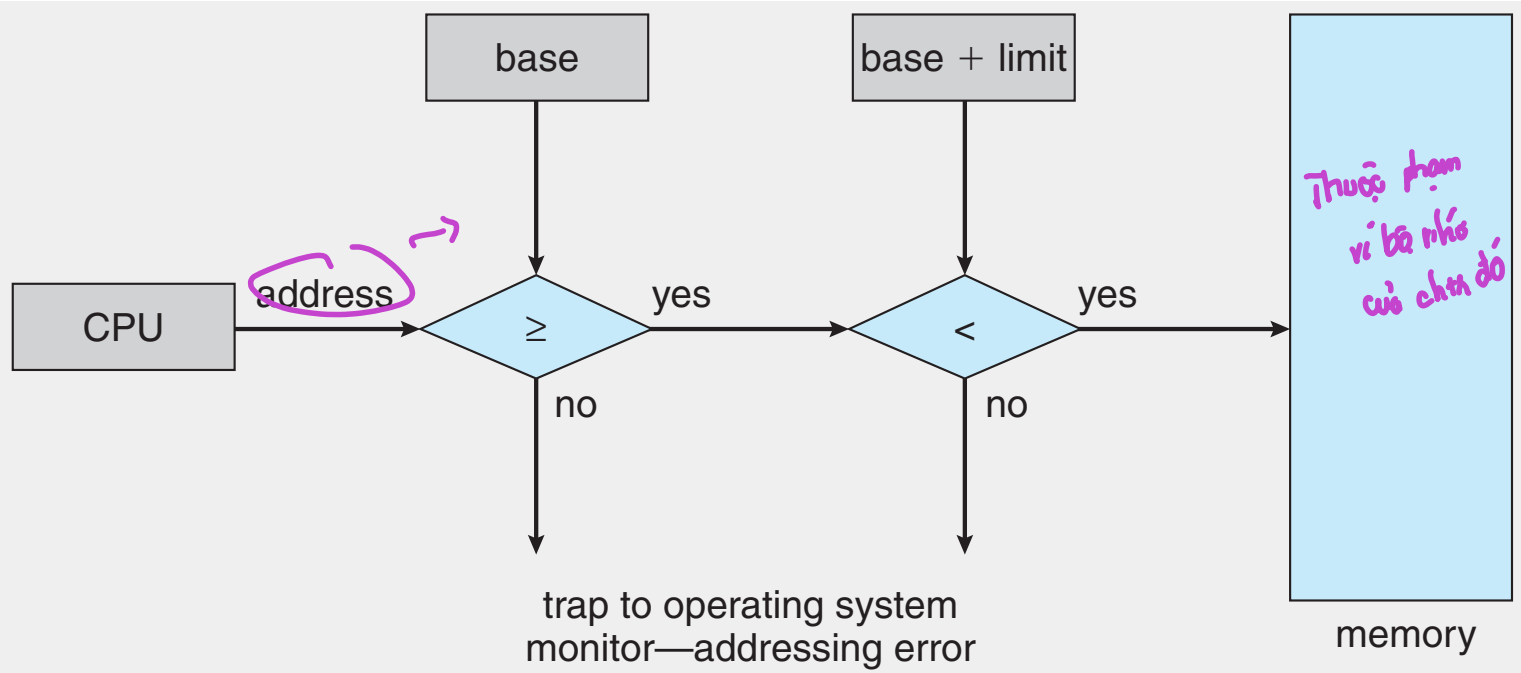
- Hỗ trợ việc phân chia vùng nhớ cho các tiến trình.
- Thanh ghi nền (base): xác định giới hạn vùng nhớ vật lý thấp nhất.
- Thanh ghi giới hạn (limit): xác định kích thước của vùng nhớ.

$\Rightarrow$ Địa chỉ vùng nhớ mà một tiến trình có thể truy xuất: [*base*, *base*+*limit*]



Bảo Vệ Không Gian Nhớ Bằng Phần Cứng

- Được thực thi bởi CPU, sử dụng t/ghi cơ sở và t/ghi giới hạn.
- Hai thanh ghi chỉ có thể được thay đổi bởi HĐH, với quyền đặc biệt.



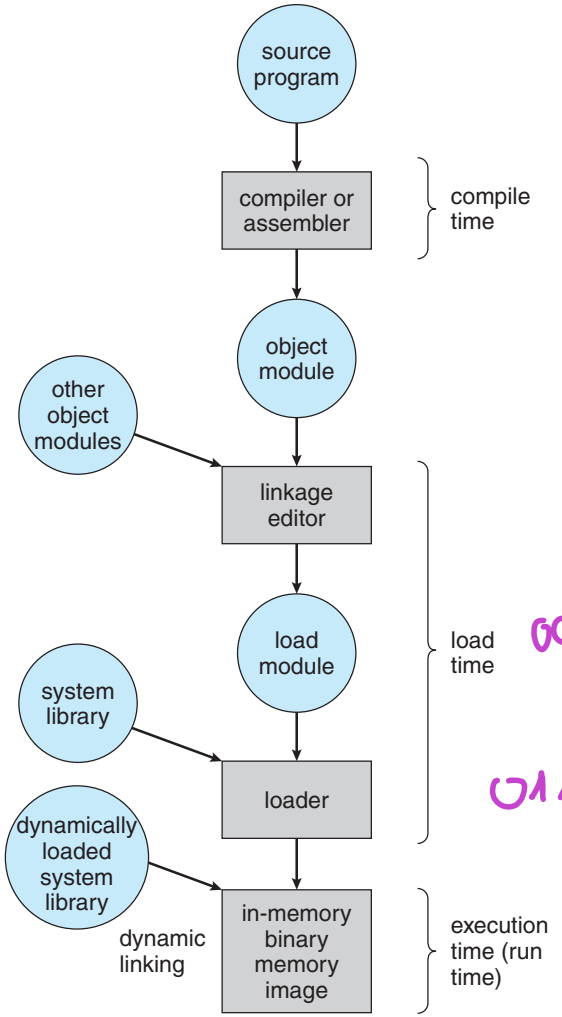
Gắn Kết (Binding) Địa Chỉ

- Tập hợp các chương trình chờ đợi để được nạp vào bộ nhớ tạo thành một hàng đợi vào (input queue).
- Các tiến trình có thể được nạp vào bất kỳ vùng nào (sẵn sàng) trên bộ nhớ, không nhất thiết từ địa chỉ 00000.
- Một chương trình thường trãi qua 1 số bước trước khi được thực thi
  $\Rightarrow$ Sự biểu diễn địa chỉ bộ nhớ trong từng g/đoạn có thể khác nhau:
  - Các địa chỉ tượng trưng trong chương trình nguồn. ✗
  - Các địa chỉ có thể tái định vị khi biên dịch.
  - Các địa chỉ tuyệt đối khi nạp (loading) hoặc kết nối (linking).
- Binding: ánh xạ địa chỉ từ không gian này sang 1 không gian khác.

Gắn Kết Dữ Liệu & Chỉ Thị Vào Bộ Nhớ

- Có thể diễn ra tại 3 giai đoạn khác nhau:
  - Thời điểm biên dịch: có thể sinh ra mã lệnh tuyệt đối (absolute code) nếu biết trước vị trí vùng nhớ. Tuy nhiên, phải biên dịch lại nếu vị trí bắt đầu của vùng nhớ thay đổi.
  - Thời điểm nạp: việc gắn kết xảy ra ở thời điểm nạp nếu trình biên dịch sinh ra mã lệnh có thể tái định vị (relocatable code) – khi nó không biết vị trí vùng nhớ khi biên dịch.
  - Thời điểm thực thi: việc gắn kết xảy ra ở thời điểm này nếu tiến trình có thể bị di chuyển từ phân đoạn (segment) bộ nhớ này sang phân đoạn bộ nhớ khác khi nó đang thực thi.
    - Cần sự hỗ trợ của phần cứng (e.g. t/ghi nền và t/ghi giới hạn)
    - Được sử dụng bởi nhiều HĐH.

Các Bước Xử Lý Một Chương Trình



Không Gian Địa Chỉ Vật Lý & Luận Lý

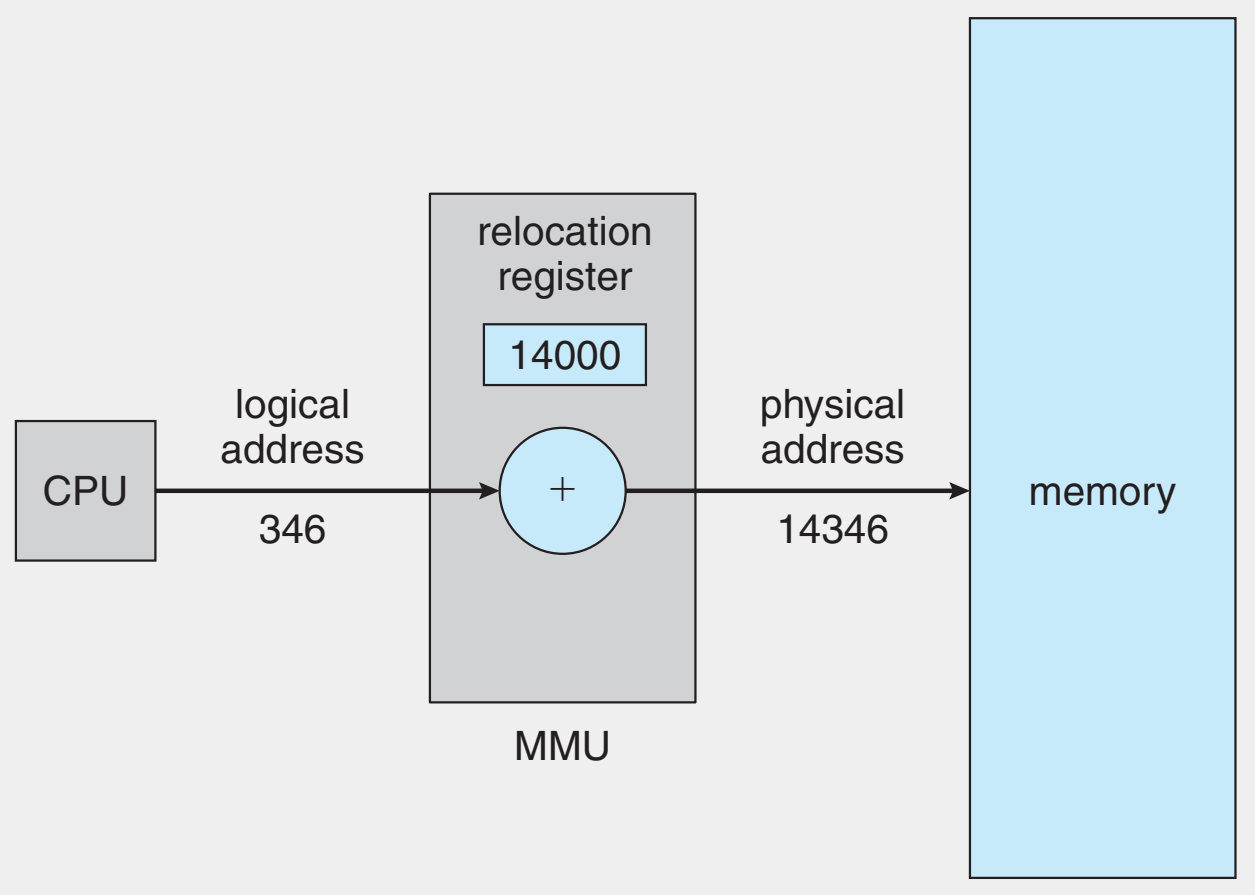
- Việc gắn kết không gian bộ nhớ vật lý và luận lý là trọng tâm của cơ chế quản lý bộ nhớ.
  - Địa chỉ luận lý (logical address): sinh ra bởi CPU, còn được gọi là địa chỉ ảo (vitual address).
  - Địa chỉ vật lý (physical address): được nhìn thấy bởi bộ quản lý bộ nhớ.
- Địa chỉ luận lý và vật lý là giống nhau trong sơ đồ gắn kết địa chỉ tại thời điểm biên dịch và nạp chương trình; và sẽ khác nhau trong sơ đồ gắn kết tại thời điểm thực thi.
- Không gian địa chỉ luận lý: tập tất cả các địa chỉ luận lý.
- Không gian địa chỉ vật lý: tập tất cả các địa chỉ vật lý.

Bộ Quản Lý Bộ Nhớ - MMU

- Là thiết bị phần cứng làm nhiệm vụ ánh xạ địa chỉ luận lý sang địa chỉ vật lý.
- Có nhiều phương pháp được sử dụng, ví dụ như phương pháp dùng thanh ghi tái định vị (reallocation register) – địa chỉ bộ nhớ sử dụng bởi tiến trình sẽ được cộng thêm giá trị của thanh ghi tái định vị khi nó truy xuất bộ nhớ.

  $\Rightarrow$ Thanh ghi tái định vị chính là thanh ghi cơ sở.
- Tiến trình người dùng chỉ dựa trên địa chỉ luận lý, không cần biết đến địa chỉ vật lý – sự ánh xạ đến địa chỉ vật lý xảy ra trong thời gian thực thi, khi tiến trình cần truy xuất bộ nhớ.

Bộ Quản Lý Bộ Nhớ – Sự Tái Định Vị Động



Nạp Động (Dynamic Loading)

- Các hàm-thư-viện (library routines) chỉ được nạp vào bộ nhớ khi nó được gọi. *(vs. nạp tĩnh: toàn bộ chương trình sẽ được nạp trước khi tiến trình bắt đầu thực thi)*
- Các hàm-thư-viện sẽ được giữ trên đĩa theo định dạng nạp có thể tái định vị (dùng địa chỉ tương đối – relative address – kể từ địa chỉ 0).
- Bộ nạp mã có thể tái định vị (reallocatable linking loader) được dùng để nạp các hàm-thư-viện (mã) cần thực thi.
- Ưu điểm: tăng hiệu năng sử dụng bộ nhớ – các hàm-thư-viện không được gọi sẽ không được nạp lên bộ nhớ.
- Hữu ích khi 1 lượng lớn các mã lệnh không thường được gọi.

Liên Kết Động (Dynamic Linking)

- Liên kết tĩnh: mã của các hàm-thư-viện và mã chương trình được kết hợp vào trong mã nhị phân của chương trình.
- Liên kết động: liên kết giữa các hàm-thư-viện và mã chương trình được thực hiện khi thực thi.
  - Một đoạn mã (gọi là stub) được sử dụng để định vị các hàm-thư-viện thường trú trong bộ nhớ (hoặc cách nạp nếu hàm-thư-viện chưa được nạp).
  - Khi cần thực thi hàm-thư-viện, stub thay thế chính nó bằng địa chỉ của hàm-thư-viện và thực thi hàm-thư-viện.
  - Stub có thể phải nạp hàm-thư-viện lên bộ nhớ nếu chưa có sẵn trên bộ nhớ.

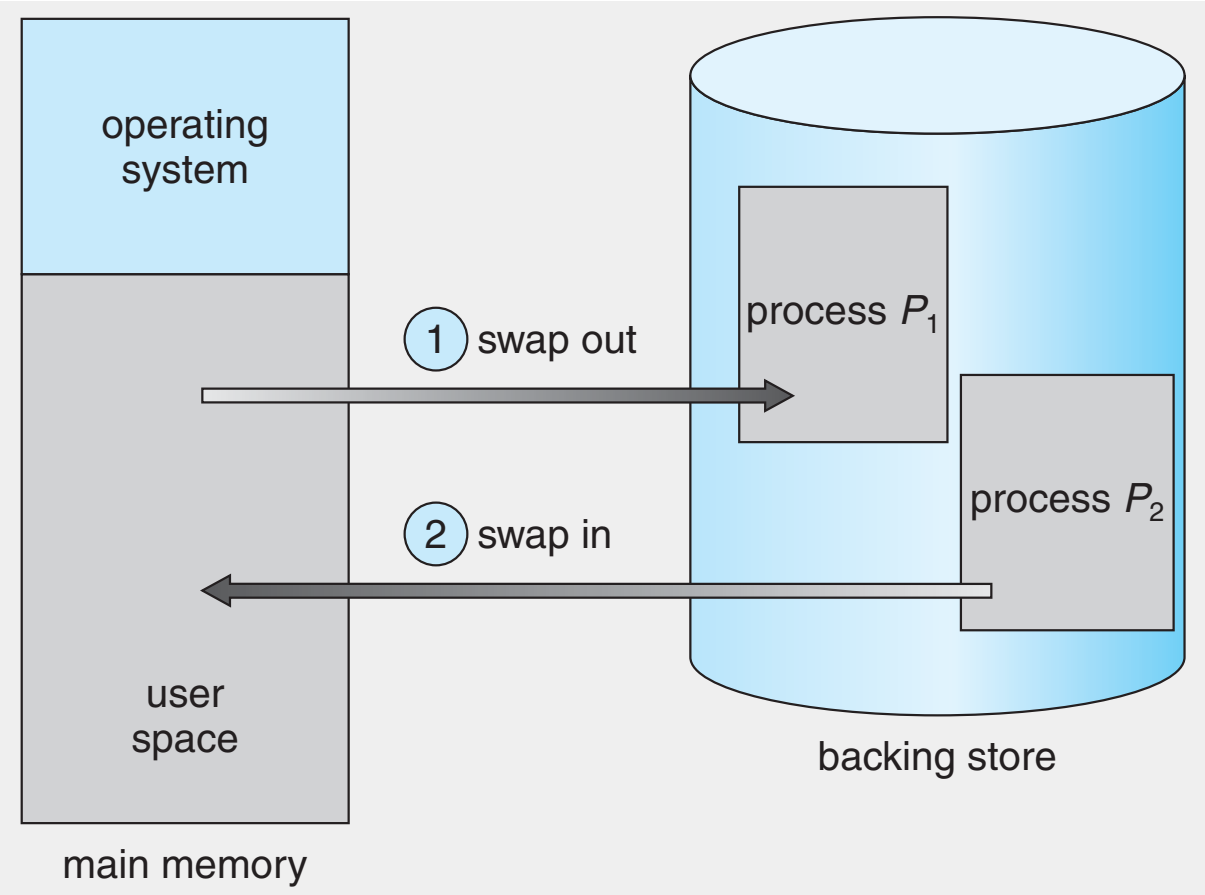
Liên Kết Động (Dynamic Linking)

- Liên kết động thường đòi hỏi sự hỗ trợ từ HĐH:
  - HĐH là thực thể duy nhất có thể kiểm tra một hàm-thư-viện có ở trong cùng không gian nhớ với chương trình gọi không.
  - Thực hiện phân quyền truy cập cho hàm-thư-viện khi cần thiết.
- Liên kết động đặc biệt hữu ích cho phép tạo các thư viện hàm chia sẻ (shared libraries).

Hoán Đổi (Swapping)

- Là một kỹ thuật cho phép tổng không gian bộ nhớ của các tiến trình lớn hơn tổng không gian nhớ vật lý:
  - Một (hay một phần) tiến trình có thể được di chuyển tạm thời từ bộ nhớ chính ra các thiết bị lưu trữ phụ (cuộn ra – roll/swap out) rồi sau đó di chuyển ngược vào bộ nhớ chính để tiếp tục thực thi (cuộn vào – roll/swap in).
- Cho phép tăng độ đa nhiệm của các hệ thống đa chương.
- Tốc độ của thiết bị lưu trữ phụ phải đủ nhanh để sao chép hiện trạng bộ nhớ (memory image) của các tiến trình và cho phép truy cập trực tiếp các dữ liệu này.

Swap Out, Swap In



Cài Đặt Hoán Đổi

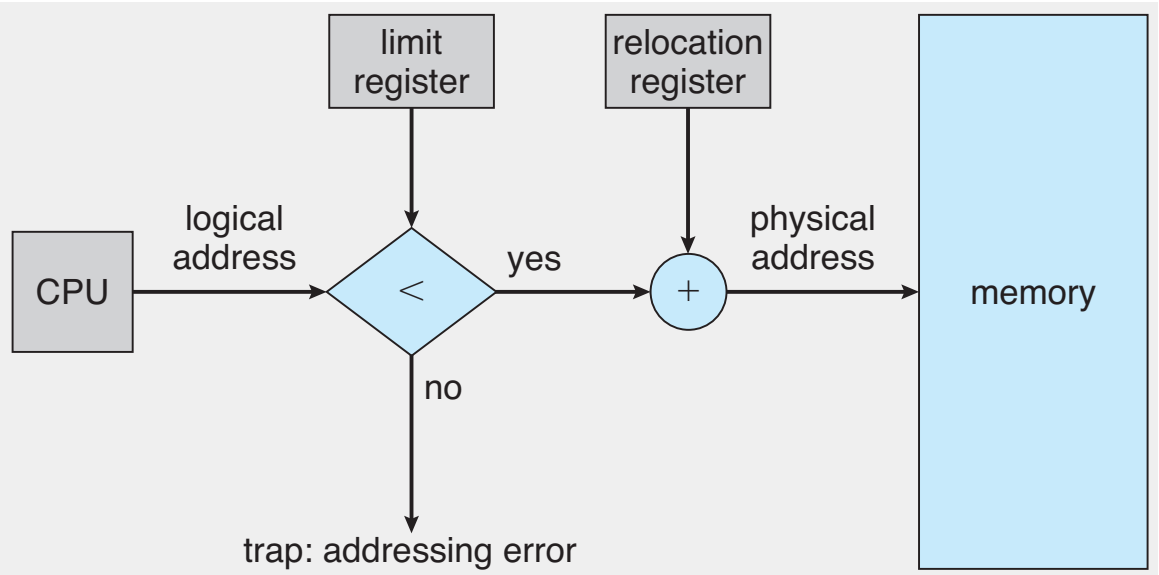
- Phần chính của thời gian hoán đổi là thời gian chuyển dữ liệu (transfer time), thường tỷ lệ với kích thước bộ nhớ hoán đổi.
- Hệ thống duy trì một hàng đợi sẵn sàng để lưu trữ danh sách các tiến trình sẵn sàng thực thi và đang được hoán đổi ra vùng lưu trữ phụ.
- Thời gian chuyển ngữ cảnh trong cách tiếp cận này tương đối cao $\Rightarrow$ thường không khả thi trong thực tế.
- Các phiên bản được sửa đổi của swapping được sử dụng trong các hệ điều hành hiện tại:
  - Swapping bình thường bị vô hiệu hóa.
  - Nó chỉ được kích hoạt khi nhu cầu bộ nhớ đạt đến 1 ngưỡng nào đó.

Cấp Phát Bộ Nhớ Kề Nhau

- Là một trong các p/pháp cấp phát bộ nhớ được sử dụng đầu tiên.
- Bộ nhớ chính thường được chia thành 2 phần:
  - Phần thường trú của HĐH: tổ chức trong vùng nhớ thấp (các vector ngắt).
  - Tiến trình người dùng: được tổ chức trong vùng nhớ cao.
- Mỗi tiến trình được cấp phát một vùng nhớ đơn, liên tục.
- Bộ quản lý bộ nhớ thực hiện ánh xạ địa chỉ luận lý sang vật lý vào thời gian thực thi (động):
  - địa chỉ vật lý = địa chỉ luận lý + giá trị thanh ghi tái định vị.

Bảo vệ vùng nhớ

- Thanh ghi tái định vị được sử dụng để bảo vệ vùng nhớ của các tiến trình người dùng và HĐH (mã lệnh và dữ liệu).
  - Thanh ghi tái định vị: chứa địa chỉ vật lý thấp nhất của tiến trình.
  - Thanh ghi giới hạn: chứa phạm vi địa chỉ luận lý của tiến trình.



Cấp Phát Đa Phân Khu Động

- Dùng cho hệ thống đa chương: mỗi t/trình được cấp phát 1 phân khu.
- Cấp độ đa chương được xác định bởi số lượng phân khu.
- K/thước mỗi phân khu có thể thay đổi tùy vào nhu cầu của t/trình.
- Lỗ trống (hole): là vùng nhớ còn trống chưa được cấp phát.
  - Có thể nằm rãi rác trong bộ nhớ (do sự cấp phát, thu hồi bộ nhớ).
  - Khi 1 t/trình xuất hiện, nó được phân 1 lỗ trống đủ chứa nó.
  - Khi 1 t/trình kết thúc, vùng nhớ dành cho nó sẽ được thu hồi thành lỗ trống và kết hợp với lỗ trống liền kề nếu có.
- Hệ thống sẽ duy trì thông tin về các phân khu đã cấp phát và lỗ trống.

Chiến Lược Cấp Phát

- Làm thế nào để đáp ứng một yêu cầu bộ nhớ kích thước n?
  - First-fit: Cấp phát lỗ trống đủ lớn đầu tiên.
  - Best-fit: Cấp phát lỗ trống đủ lớn nhỏ nhất.
    - Phải kiểm tra toàn bộ các lỗ trống nếu các lỗ trống không được sắp xếp.
    - Sẽ phát sinh lỗ trống có kích thước nhỏ nhất.
  - Worst-fit: Cấp lỗ trống lớn nhất
    - Có thể phải k/tra toàn bộ các lỗ trống.
    - Sinh ra lỗ trống còn lại lớn nhất.
- First-fit và best-fit cho hiệu năng và tốc độ tốt hơn worst-fit.

Sự Phân Mảnh

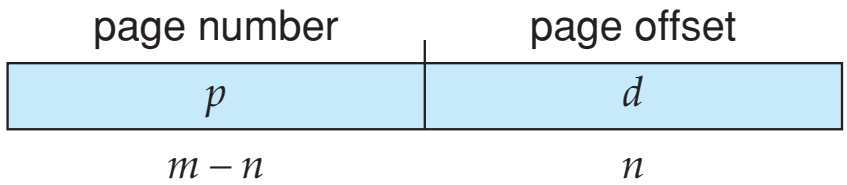
- Phân mảnh trong: Phân khu cấp phát cho tiến trình lớn hơn nhu cầu.
- Phân mảnh ngoài: Các lỗ trống nằm rãi rác $\Rightarrow$ có thể xảy ra trường hợp tổng kích các lỗ trống lớn hơn như cầu nhưng chúng không nằm liên tục nên không thể cấp phát.
- Khử phân mảnh ngoài: cô đặc lại bộ nhớ – sắp xếp lại bộ nhớ để gom các lỗ trống lại.
  - chỉ thực hiện được khi việc tái định vị là động (thực hiện lúc thực thi).
- Một phương pháp khử phân mảnh ngoài là cấp phát không liên tục.

Cơ Chế Phân Trang (Paging)

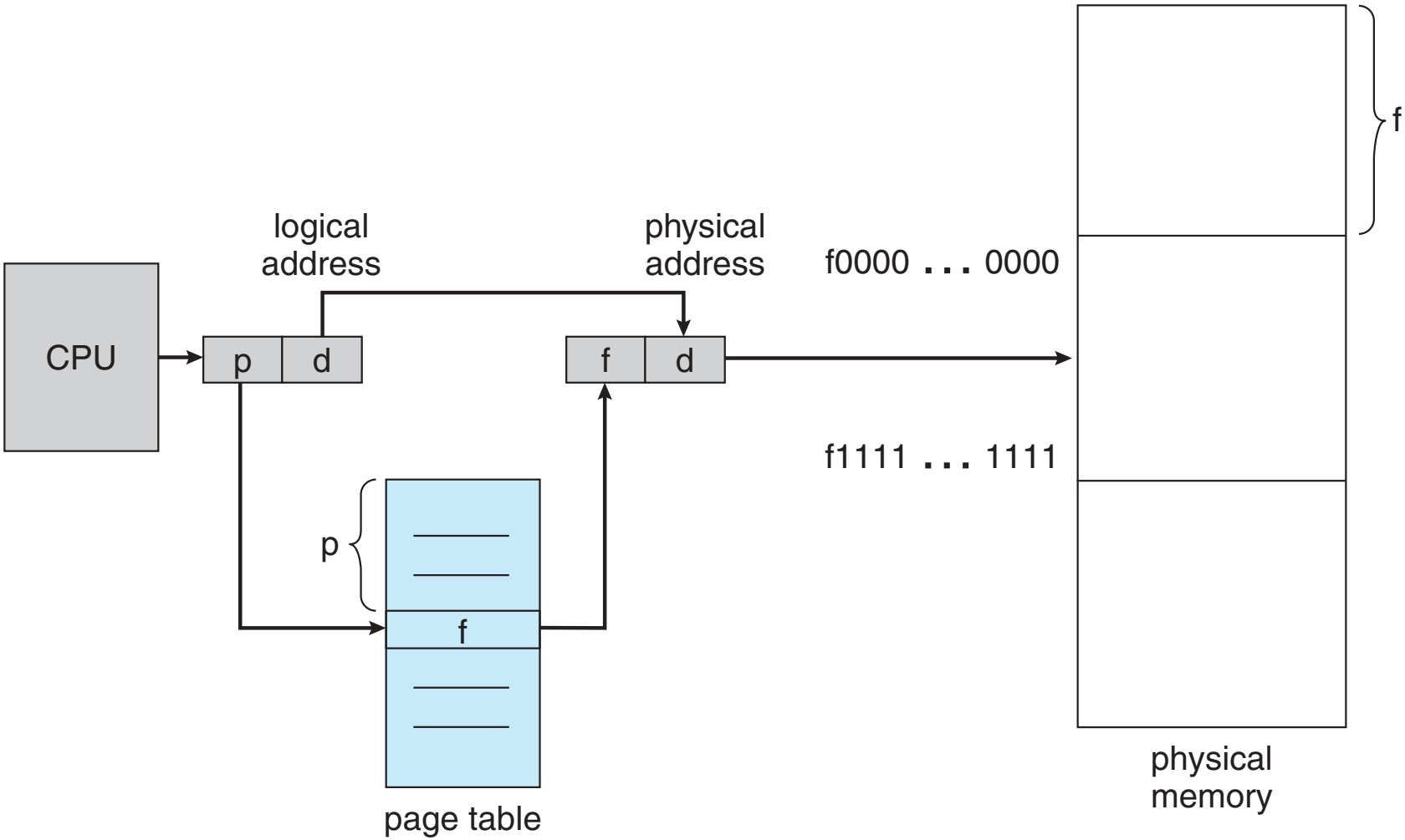
- Cho phép không gian địa chỉ vật lý cấp phát cho 1 tiến trình có thể không liên tục nhau $\Rightarrow$ tránh được phân mãnh ngoài bộ nhớ.
- Bộ nhớ vật lý được chia thành các khối có kích thước cố định (2^n bytes, thường từ 512K – 16MB), gọi là các khung (frame).
- Vùng nhớ luận lý của tiến trình cũng được chia thành các khối có kích thước cố định (bằng kích thước frame), gọi là trang nhớ (page).
  - Một chương trình cần n trang sẽ được cấp phát n khung.
  - Có thể phát sinh phân mảnh trong.
- Hệ thống sẽ theo dõi các khung trống và thiết đặt một bảng trang (table page) để ánh xạ địa chỉ luận lý sang địa chỉ vật lý.

Định Địa Chỉ Trong Phân Trang

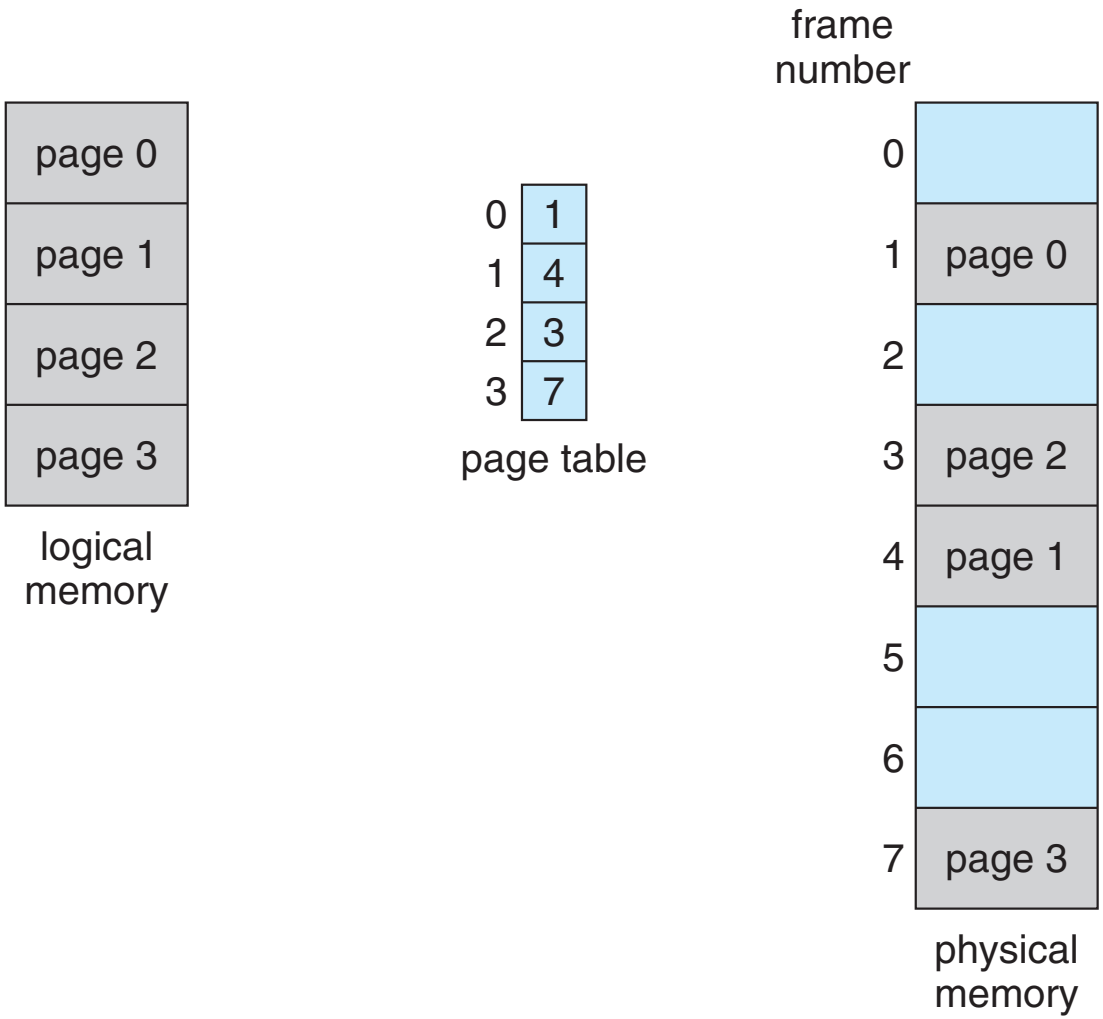
- Một địa chỉ luận lý bao gồm:
  - Số hiệu trang (page number – p): dùng làm chỉ mục trong bảng phân trang để tìm ra chỉ số khung tương ứng (địa chỉ nền) trong bộ nhớ vật lý.
  - Độ dời trang (page offset – d): được kết hợp với địa chỉ nền để định vị địa chỉ vật lý bộ nhớ.
- Nếu kích thước của vùng nhớ luận lý là 2^m bytes và kích thước trang là 2^n bytes thì m-n bits cao được dùng để đánh số trang và n bits thấp được dùng để gán độ dời trang.



Cấu Trúc Hệ Thống Dịch Địa Chỉ



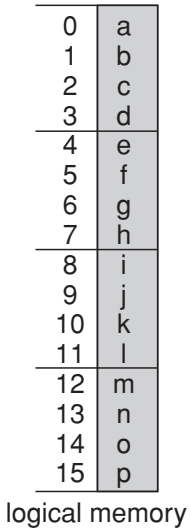
Mô Hình Phân Trang

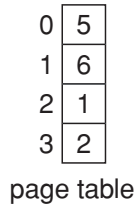


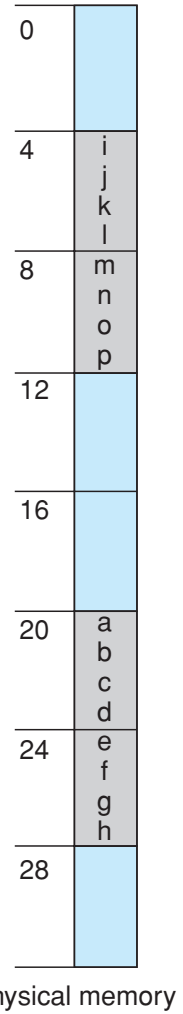
Ví Dụ Về Phân Trang

VD: CPU đọc địa chỉ luận lý: 9

=> Tính địa chỉ vật lý.







- Một trang có kích thước 4 bytes
- Bộ nhớ vật lý: 32 bytes (8 khung).
- Địa chỉ luận lý 3 được ánh xạ vào địa chỉ vật lý 23.

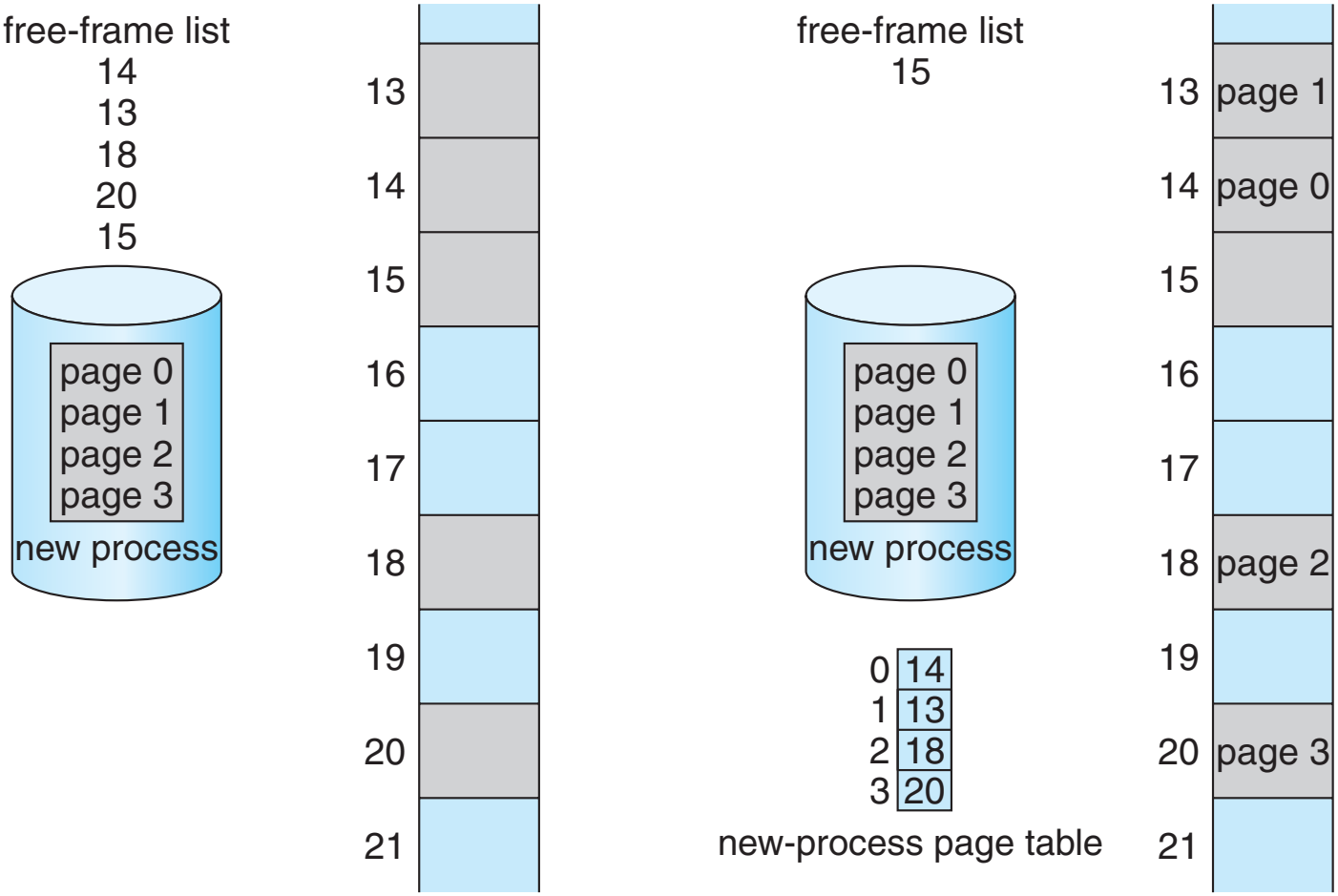
① Địa chỉ luận lý 14 nằm ở trang : 14 div 4 = 3

② Độ dời của địa chỉ 14 so với đầu trang 14 mod 4 = 2

③ Tra bảng trang ra có : Trang 3 được đặt tại Khung 2

④ Địa chỉ vật lý của địa chỉ luận lý 14 là 2x4 + 2 = 10

Ví Dụ Về Phân Trang - Quản Lý Khung Trống



Phân Mảnh Trong Trong Phân Trang

- Trang cuối cùng được cấp phát có thể không được sử dụng hết $\Rightarrow$ phân mảnh trong.
- Ví dụ:
  - Kích thước trang: 2.048 bytes.
  - Kích thước tiến trình: 72.766 bytes.
    - 35 trang + 1.086 bytes
    - Cấp phát: 36 trang.
  - Phân mảnh trong: 2.048 - 1.086 = 962 bytes Không được sử dụng
- Trường hợp phân mảnh xấu nhất = kích thước 1 trang - 1 bytes.

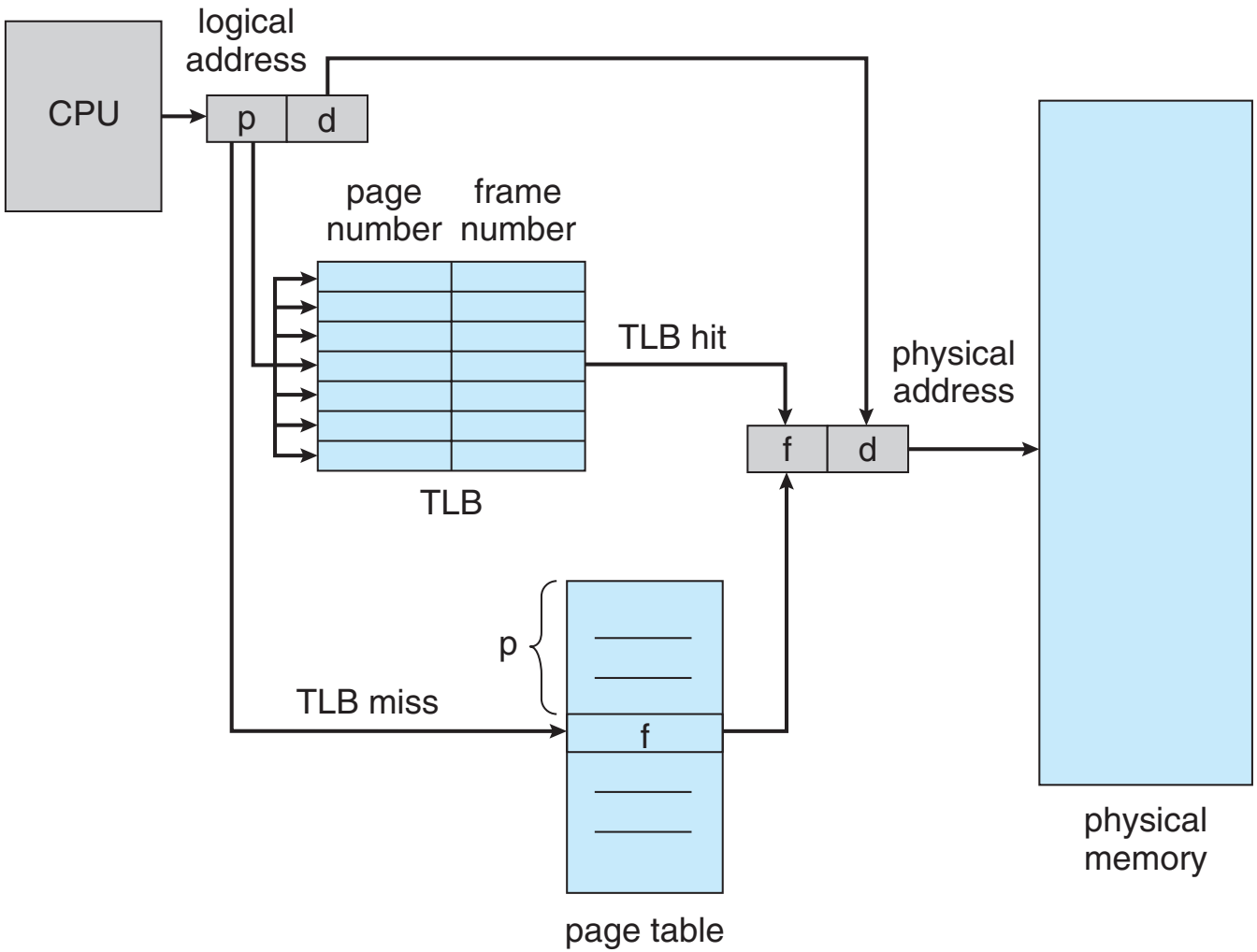
Cài Đặt Bảng Trang

- Bảng trang thường được lưu giữ trong bộ nhớ chính:
  - Mỗi tiến trình có một bảng trang.
  - Thanh ghi bảng trang nền: trỏ đến bảng trang.
  - Thanh ghi chiều dài bảng trang: chỉ định kích thước của bảng trang.
- Mỗi tác vụ truy cập bộ nhớ cần 2 thao tác truy cập vùng nhớ:
  - 1 thao tác truy xuất bảng trang, sử dụng p để lấy số khung.
  - 1 thao tác truy xuất bộ nhớ vật lý, sử dụng d để tính đ/chỉ vật lý.
  - Thường sử dụng cache phần cứng để tăng tốc độ các thao tác này như thanh ghi kết hợp (associative register) hoặc bộ đệm tìm kiếm phụ cho việc dịch địa chỉ (Translation Look-aside Buffer – TLBs).

Bộ Nhớ Kết Hợp – TLBs

- Là thanh ghi cực nhanh, chứa ánh xạ giữa trang và khung.
- Có kích thước nhỏ: 64 – 1024 mục.
- Được xem như là bộ đệm của bảng trang.
- Cơ chế làm việc (ánh xạ $\langle p, d \rangle$ vào bộ nhớ vật lý):
  - Nếu p nằm trong TLBs: lấy ngay được số frame.
  - Ngược lại: truy xuất bảng trang để lấy số khung trang như bình thường. Đồng thời, đưa ánh xạ trang–khung mới sử dụng vào TLBs.

Phân Trang Với TLB



Thời Gian Truy Xuất Hiệu Dụng

- Thời gian truy xuất hiệu dụng (EAT):

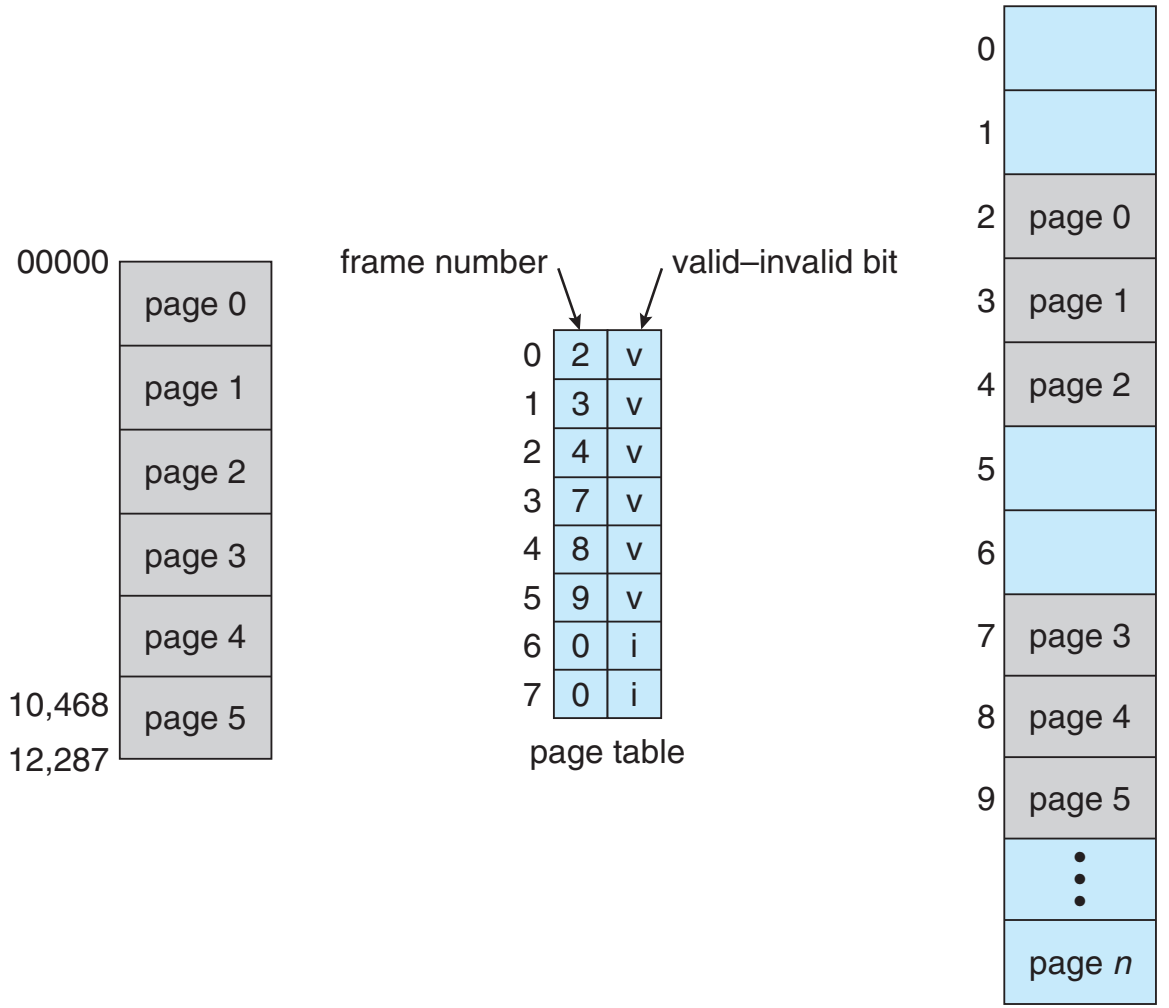
$$\begin{aligned} EAT &= (\varepsilon + m)\alpha + (\varepsilon + 2m)(1 - \alpha) \\ &= (2 - \alpha)m + \varepsilon \end{aligned}$$

  - ε: thời gian tìm kiếm trên thanh ghi kết hợp (đ/vị thời gian).
  - m: thời gian 1 chu kỳ truy xuất bộ nhớ.
  - α: tỷ lệ tìm thấy (hit ratio) – tỷ số giữa số lần chỉ số trang được tìm thấy trong TLB trên tổng số lần truy cập bộ nhớ.
- Ví dụ: $\varepsilon = 20ns, \alpha = 0.8, m = 100ns \Rightarrow ETA = 140ns$

Bảo Vệ Bộ Nhớ (Memory Protection)

- Mỗi khung được gắn một số bit bảo vệ (protection bits) để chỉ định các tiến trình có thể thực hiện các thao tác gì trên khung.
  - read-only, read-write, execute, ...
- Mỗi mục trong bảng trang được gắn 1 bit xác định tính hợp lệ:
  - *valid*: trang tương ứng đang nằm trong không gian địa chỉ luận lý của tiến trình $\Rightarrow$ trang hợp lệ (có thể truy xuất).
  - *invalid*: trang tương ứng không nằm trong không gian địa chỉ luận lý của bất kỳ tiến trình nào $\Rightarrow$ không hợp lệ (không thể truy xuất).
  - Giải pháp khác bằng phần cứng: dùng thanh ghi độ dài bảng trang (page-table length register – PTLR) để xác định kích thước bảng trang.

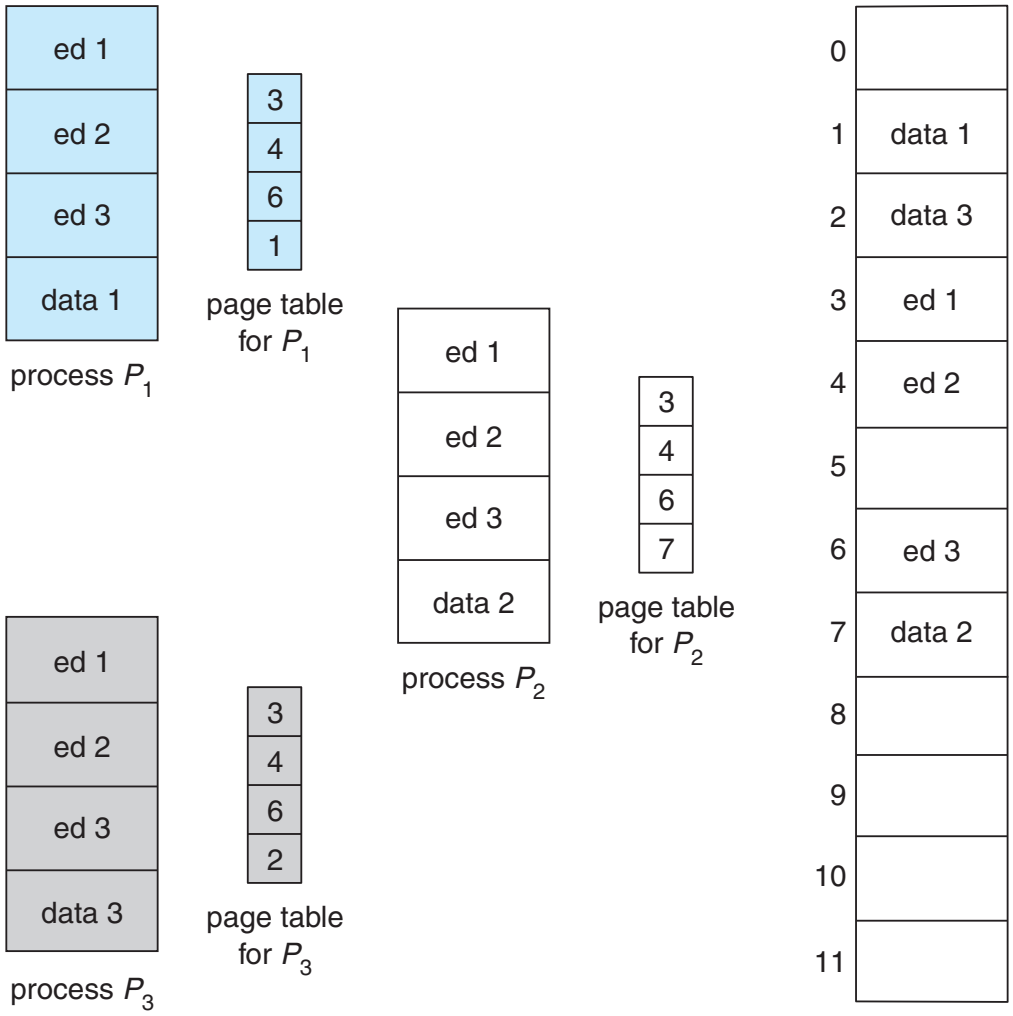
Bit Hợp Lệ, Không Hợp Lệ



Chia Sẻ Trang (Shared Page)

- Chia sẻ mã lệnh:
  - Các tiến trình của cùng 1 ch/trình có thể chia sẻ trang chứa mã lệnh.
  - Trang được chia sẻ thường được gán thuộc tính chỉ đọc.
- Giao tiếp liên tiến trình:
  - Các trang nhớ có thể được chia sẻ giữa các tiến trình để thực hiện giao tiếp liên tiến trình.
  - Phải được gán thuộc tính đọc-ghi.
- Ngoài ra, mặc nhiên thì các luồng trong cùng một tiến trình có thể chia sẻ chung các trang nhớ trong cùng một không gian tiến trình.

Chia Sẻ Trang (Shared Page)



Các cấu Trúc Bảng Trang

- Kích thước bảng trang có thể rất lớn nếu bảng trang được cấu trúc đơn giản.
  - Xét 1 hệ thống máy tính 32-bit; kích thước trang: 4K (2^{12})

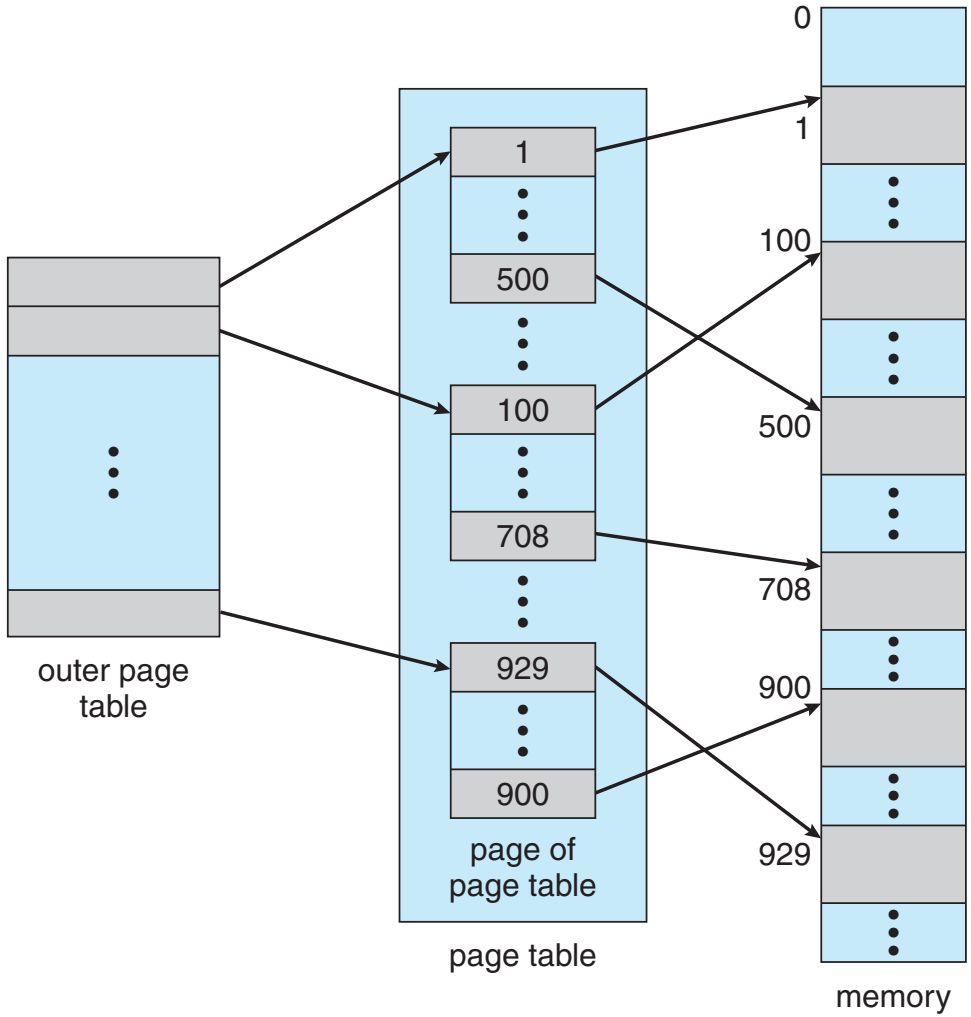
    $\Rightarrow$ Số mục của bảng trang: $2^{32}/2^{12} = 2^{20}$
  - Nếu kích thước mỗi mục là 4 bytes thì kích thước bảng trang là 4MB.

    $\Rightarrow$ Tương đối lớn cho việc truy xuất và cấp phát liên tục.
- Một số cấu trúc bảng trang:
  - Bảng trang phân cấp (hierarchical page table).
  - Bảng trang được băm (hashed page table).
  - Bảng trang đảo (inverted page table).

Bảng Trang Phân Cấp

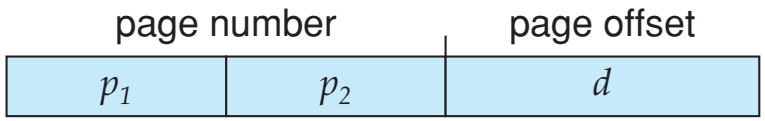
- Phân không gian địa chỉ luận lý vào nhiều bảng trang.
- Số bảng trang được gọi là số cấp (số mức).
- Ví dụ: bảng trang phân cấp đơn giản là bảng trang 2 cấp (two-level page table), trong đó không gian bộ nhớ được chia vào 2 bảng trang
  - bảng trang cấp 1 (bảng trang ngoài – outer page table): chỉ mục cho bảng trang bên trong.
  - bảng trang cấp 2 (bảng trang trong – inner page table): ánh xạ địa chỉ luận lý vào địa chỉ vật lý.
- Đây là 1 hình thức phân trang cho bảng trang.

Bảng Trang Phân Cấp



Cấu Trúc Địa Chỉ Luận Lý

- Cấu trúc địa chỉ bộ nhớ luận lý cho bảng trang 2 cấp gồm 3 phần:

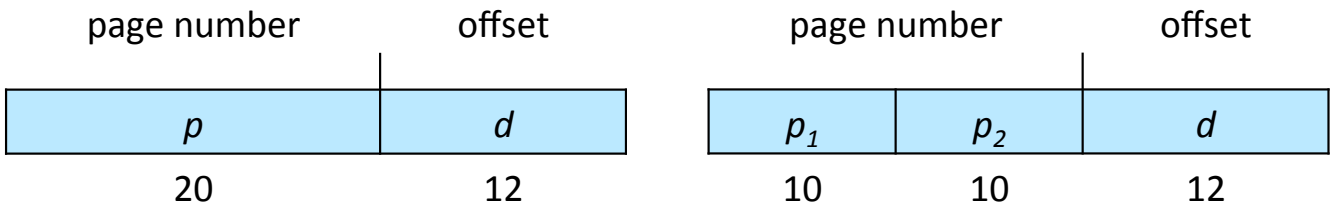


  - p_1: chỉ mục bảng trang cấp 1 (bảng trang ngoài – outter page table).
  - p_2: chỉ mục bảng trang cấp 2 (bảng trang trong – inner page table).
  - d: độ dời bên trong khung.
- Cấu trúc trên có thể được tổng quát hóa cho trường hợp bảng trang phân cấp cấp n.

Cấu Trúc Địa Chỉ Luận Lý – Ví Dụ

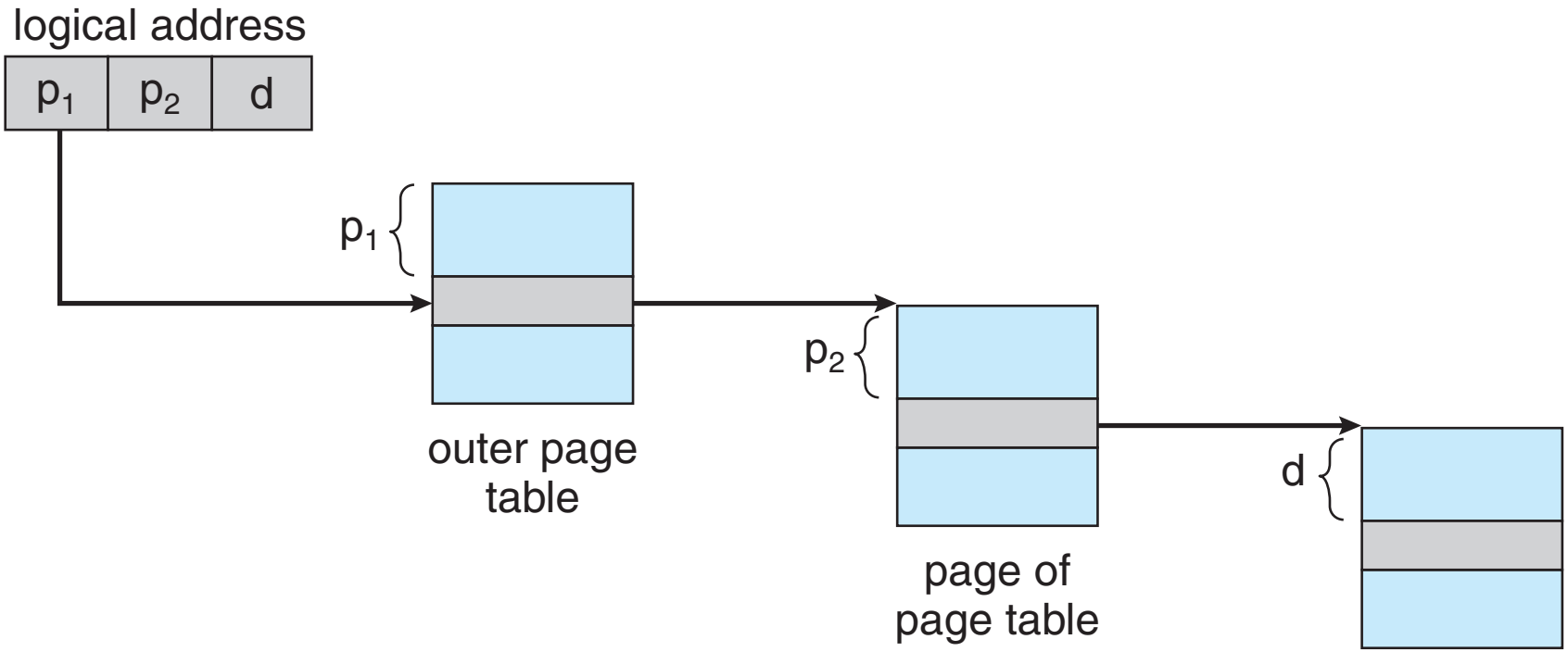
- Trong một hệ thống 32-bit với trang nhớ có kích thước 4K (2^{12}):
  - Bảng trang có thể chứa đến $2^{32-12} = 2^{20}$ mục (entries).

    $\Rightarrow$ Cần 20 bits để đánh số hiệu trang, 12 bits chỉ định độ dời.



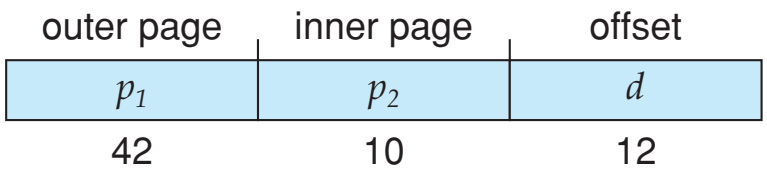
- Bảng trang 2 cấp: số hiệu trang được chia thành 2 phần (2 cấp).
  - Nếu muốn mỗi bảng trang cấp 2 có thể được chứa trong 1 trang bộ nhớ, giả sử mỗi mục trong bảng trang chiếm 4B: cần 10 bits để đánh chỉ số cho các mục trong bảng trang.
  - Còn lại 10 bits dùng để đánh chỉ số các mục của bảng trang cấp 1.

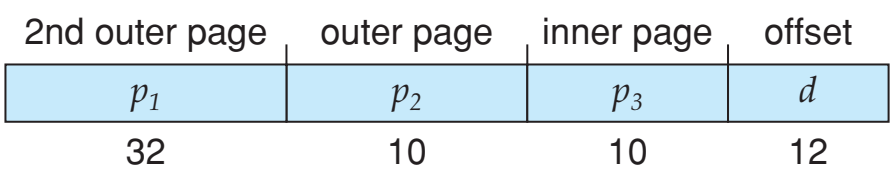
Sơ Đồ Chuyển Đổi Địa Chỉ



Không Gian Nhớ Luận Lý 64-Bit

- Bảng trang 2 cấp không phù hợp cho không gian nhớ 64-bit.
  - Giả sử kích thước trang là 4K $\Rightarrow$ bảng trang có 2^{52} mục.
  - Nếu dùng bảng trang 2 cấp, bảng trang cấp 2 có 2^{10} mục $\Rightarrow$ sử dụng 10 bits để đánh chỉ số cho bảng trang cấp 2.
  - Bảng trang cấp 1 có tối đa 2^{42} mục $\Rightarrow$ kích thước lên đến 4TB.
- Giải pháp là tăng số cấp của bảng trang: số cấp trong trường hợp này là tương đối lớn vì đến 3 cấp thì k/thước bảng trang cấp 1 vẫn là 16G.





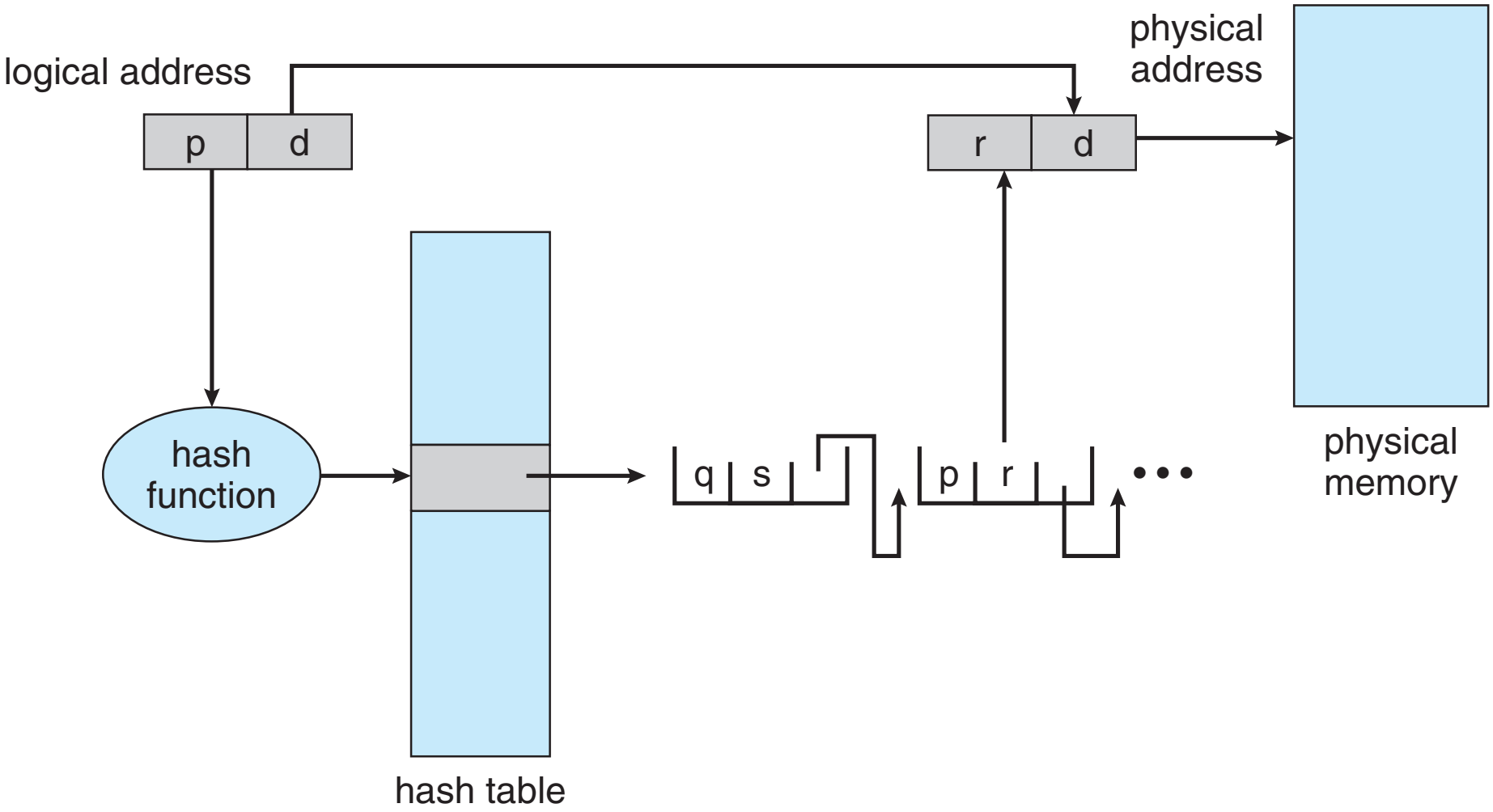
Nhận Xét

- Ưu điểm:
  - Cấu trúc bảng trang phân cấp giúp tiết kiệm bộ nhớ cấp phát cho bảng trang: chỉ cấp phát đúng số bảng trang tiến trình cần dùng.
  - Phương pháp cấp phát, ánh xạ địa chỉ nhớ đơn giản.
  - Cho phép chia sẻ bộ nhớ giữa các tiến trình ở mức trang.
- Khuyết điểm:
  - Đòi hỏi thực hiện nhiều thao tác tìm kiếm (ánh xạ) địa chỉ nhớ cho mỗi truy cập bộ nhớ.
  - Phải tốn chi phí bộ nhớ cho bảng trang.
  - Kích thước bảng trang sẽ rất lớn trong các hệ thống có không gian địa chỉ lớn.

Bảng Trang Băm (Hashed Page Table)

- Dùng bảng băm để giảm không gian bảng trang.
  - Được sử dụng trong các hệ thống có không gian bộ nhớ $>$ 32-bits.
- Số hiệu trang ảo được băm vào bảng trang.
- Mỗi mục từ của bảng băm sẽ chỉ đến một danh sách liên kết các phần tử được băm vào cùng vị trí.
- Một phần tử trong danh sách là một cặp ⟨chỉ số trang ảo, chỉ số frame⟩.
- Tìm trang (p, d):
  - Số trang ảo được băm để tìm vị trí của mục từ trong bảng băm.
  - Duyệt qua d/sách l/kết của mục từ để tìm số khung trang tương ứng.

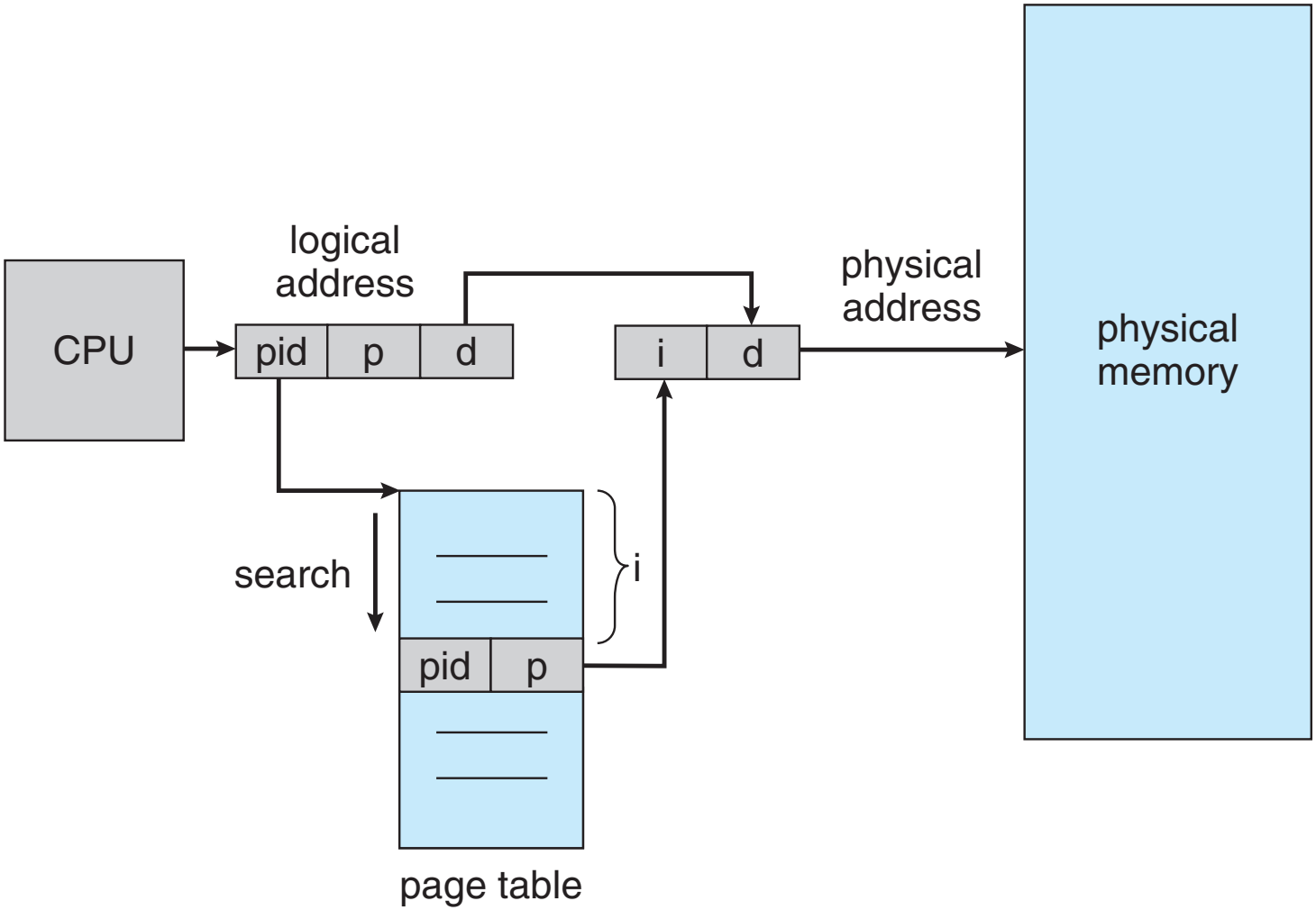
Bảng Trang Băm (Hashed Page Table)



Bảng Trang Đảo (Inverted Page Table)

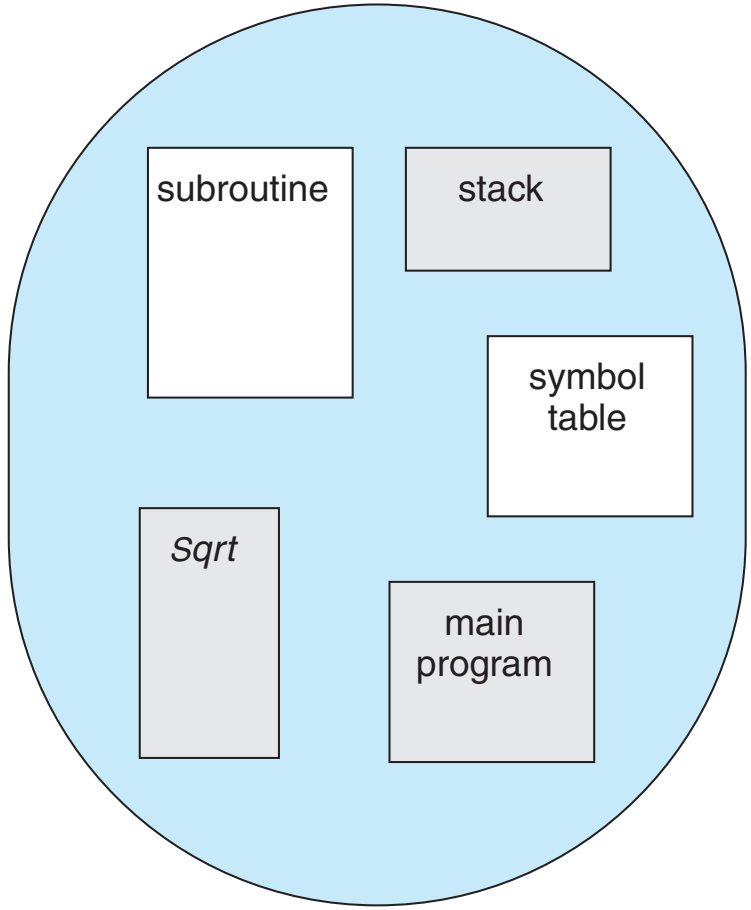
- Mỗi mục từ của bảng trang đại diện cho 1 khung trong bộ nhớ.
- Một mục từ chứa địa chỉ ảo của trang được chứa trong bộ nhớ thực và thông tin về tiến trình sở hữu nó.
- Phương pháp này giảm bộ nhớ cần thiết lưu trữ các bảng trang, nhưng lại mất thời gian tìm kiếm bảng trang khi truy xuất 1 trang được yêu cầu.
- Có thể dùng bảng băm để giới hạn tìm kiếm cho 1 hoặc 1 vài mục từ của bảng trang.

Bảng Trang Đảo (Inverted Page Table)



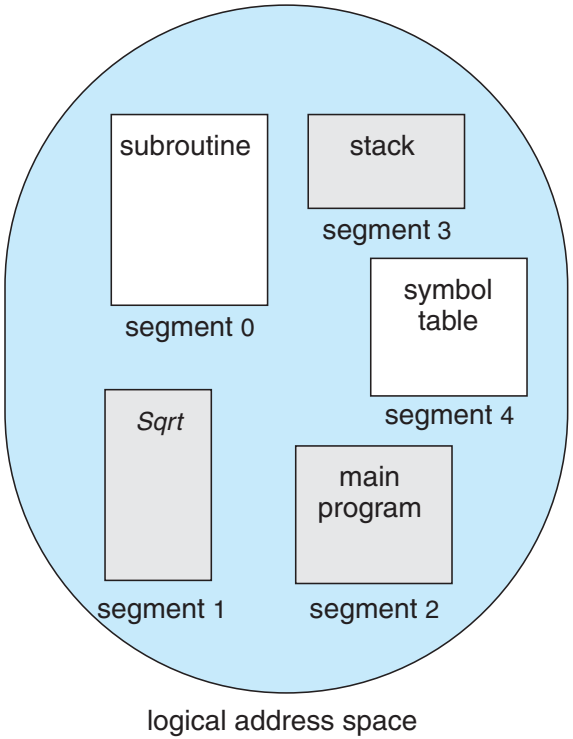
Phân Đoạn (Segmentation)

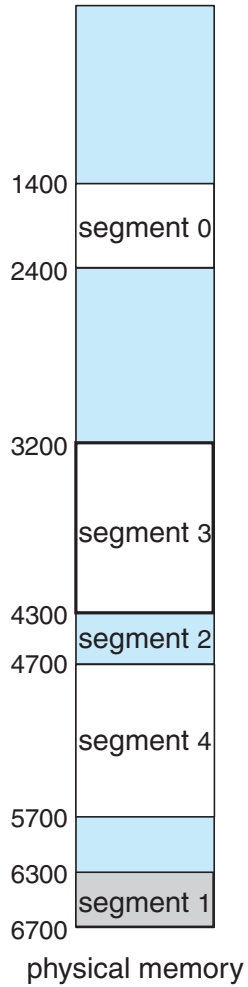
- Bộ nhớ ở góc độ người dùng:
  - Chương trình là một tập các đoạn (segment).
  - Mỗi đoạn là một đơn vị luận lý như: chương trình chính, hàm, đối tượng, mảng, ...
- Phân đoạn:
  - Là một sơ đồ quản lý bộ nhớ hỗ trợ cái nhìn về bộ nhớ theo góc độ người dùng.



Hai Góc Nhìn Bộ Nhớ

- Phân đoạn cung cấp cơ chế ánh xạ giữa không gian địa chỉ luận lý và không gian địa chỉ vật lý.



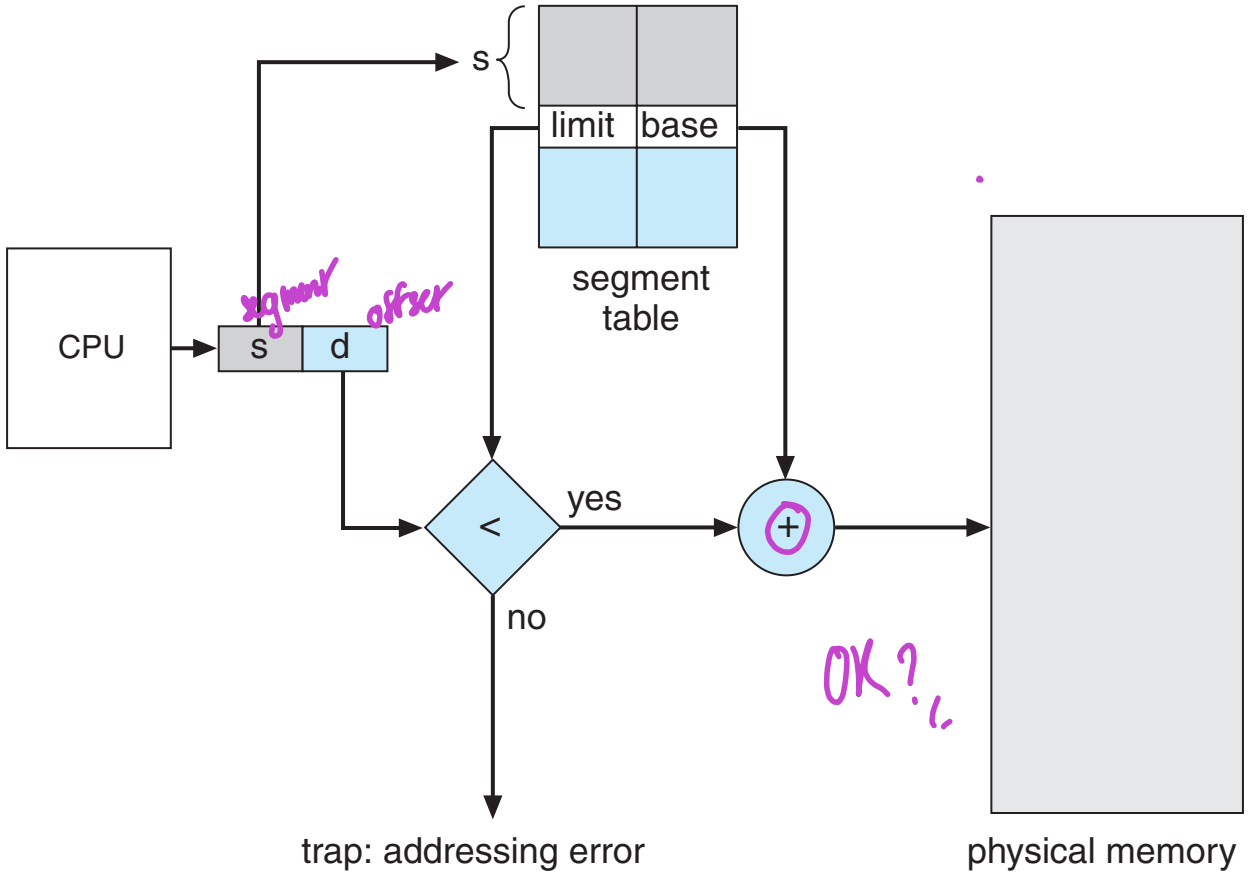


Kiến Trúc Hệ Thống Phân Đoạn

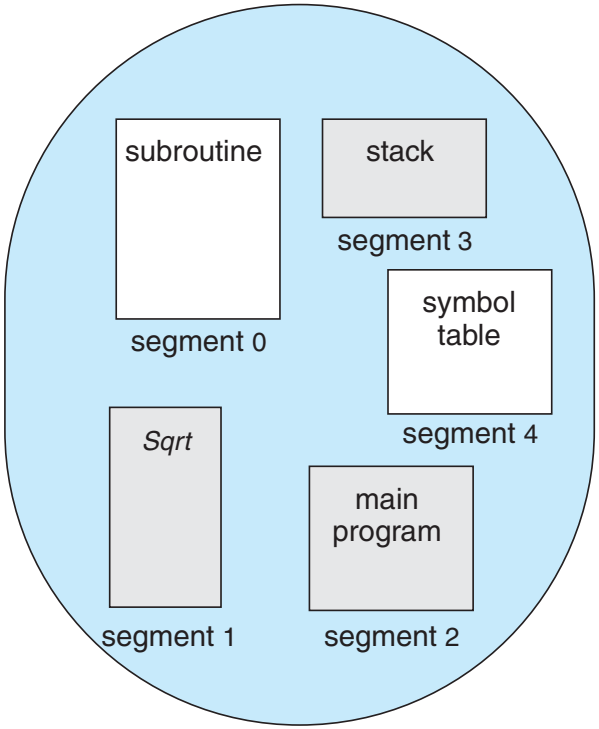
- Một địa chỉ luận lý bao gồm một cặp: $\langle segment\_number, offset \rangle$
- Bảng phân đoạn (segment table): chứa các mục ánh xạ các địa chỉ luận lý (2 chiều) vào các địa chỉ vật lý 1 chiều; mỗi mục gồm:
  - *base*: chứa địa chỉ vật lý khởi đầu của đoạn trong bộ nhớ vật lý.
  - *limit*: chỉ định chiều dài của đoạn.
- Segment-table base register (STBR): là thanh ghi trỏ đến bảng phân đoạn trong bộ nhớ.
- Segment-table length register (STLR): là thanh ghi chỉ ra số lượng các đoạn đang được sử dụng bởi chương trình.

Phần Cứng Hỗ Trợ Phân Đoạn

- Một đoạn có số hiệu s hợp lệ nếu $s <$ STLR.

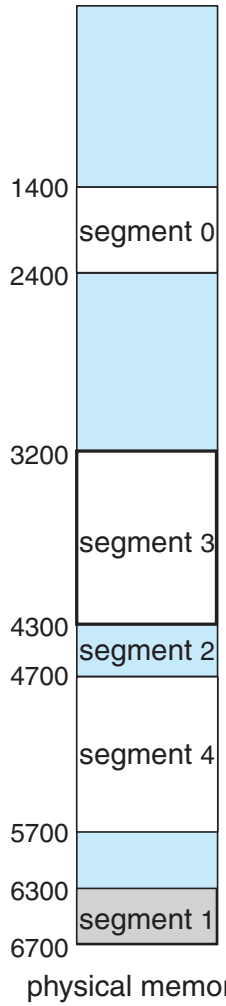


Ví Dụ Về Phân Đoạn



	limit	base
0	1000	1400
1	400	6300
2	400	4300
3	1100	3200
4	1000	4700

segment table

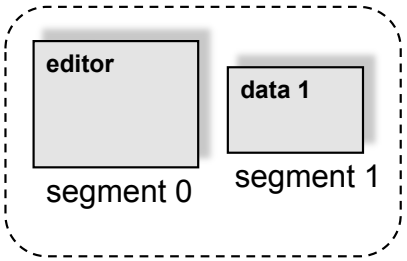


Ưu điểm:
Tránh phân mảnh.
(Ko có)

Bảo Vệ Và Chia Sẻ Đoạn

- Bảo vệ đoạn: kết hợp mỗi mục trong bảng phân đoạn
  - 1 bit bảo vệ để đánh dấu đoạn hợp lệ/không hợp lệ.
  - các bits kiểm soát quyền (đọc, ghi, thực thi, . . .)

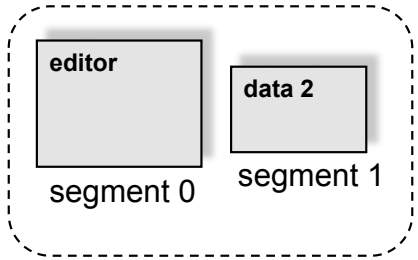
- Chia sẻ đoạn:



logical address space process P_1

	limit	base
0	25286	43062
1	4425	68348

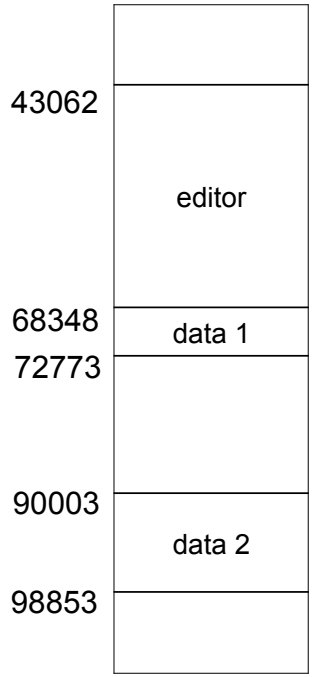
segment table process P_1



logical address space process P_2

	limit	base
0	25286	43062
1	8850	90003

segment table process P_2

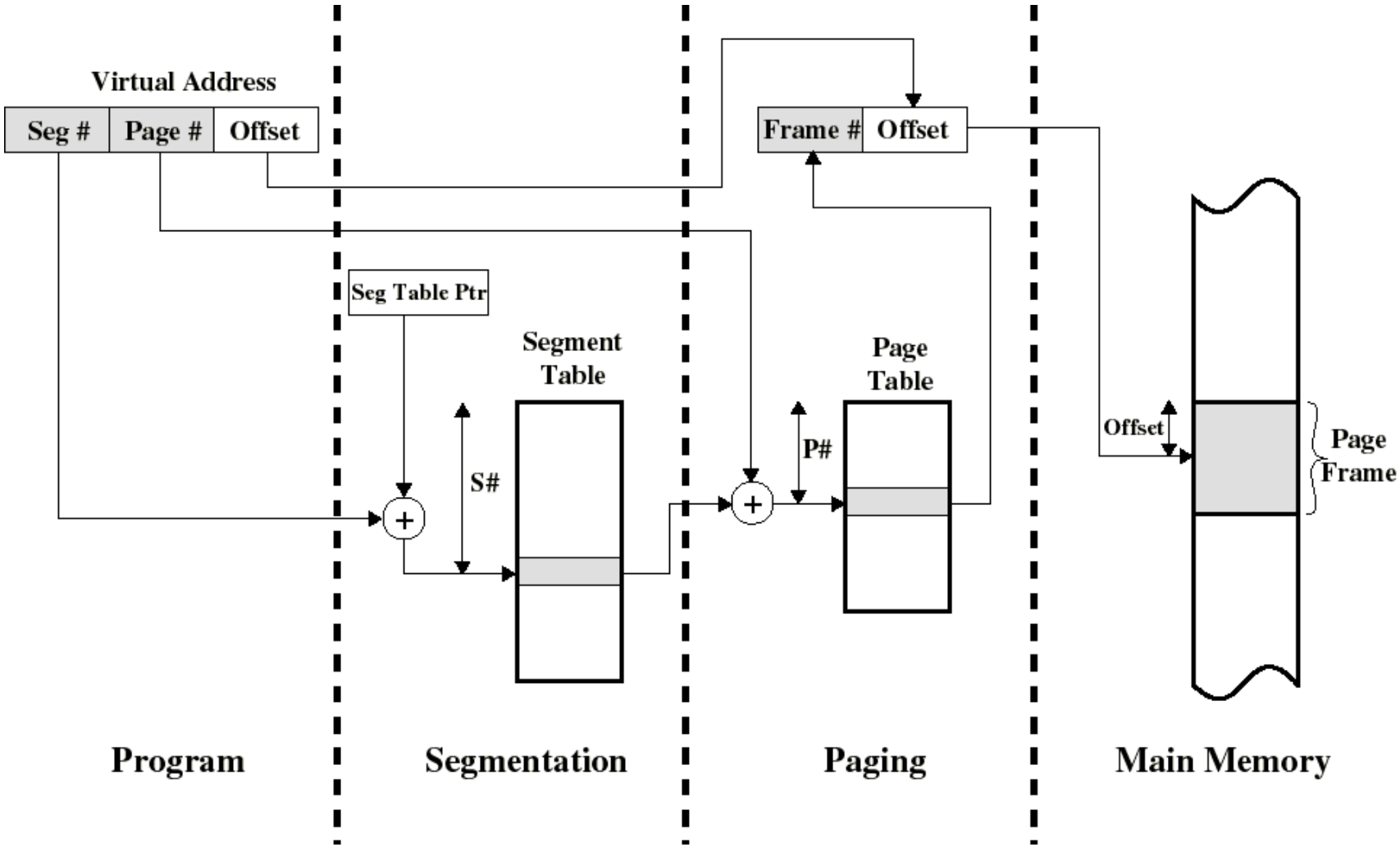


physical memory

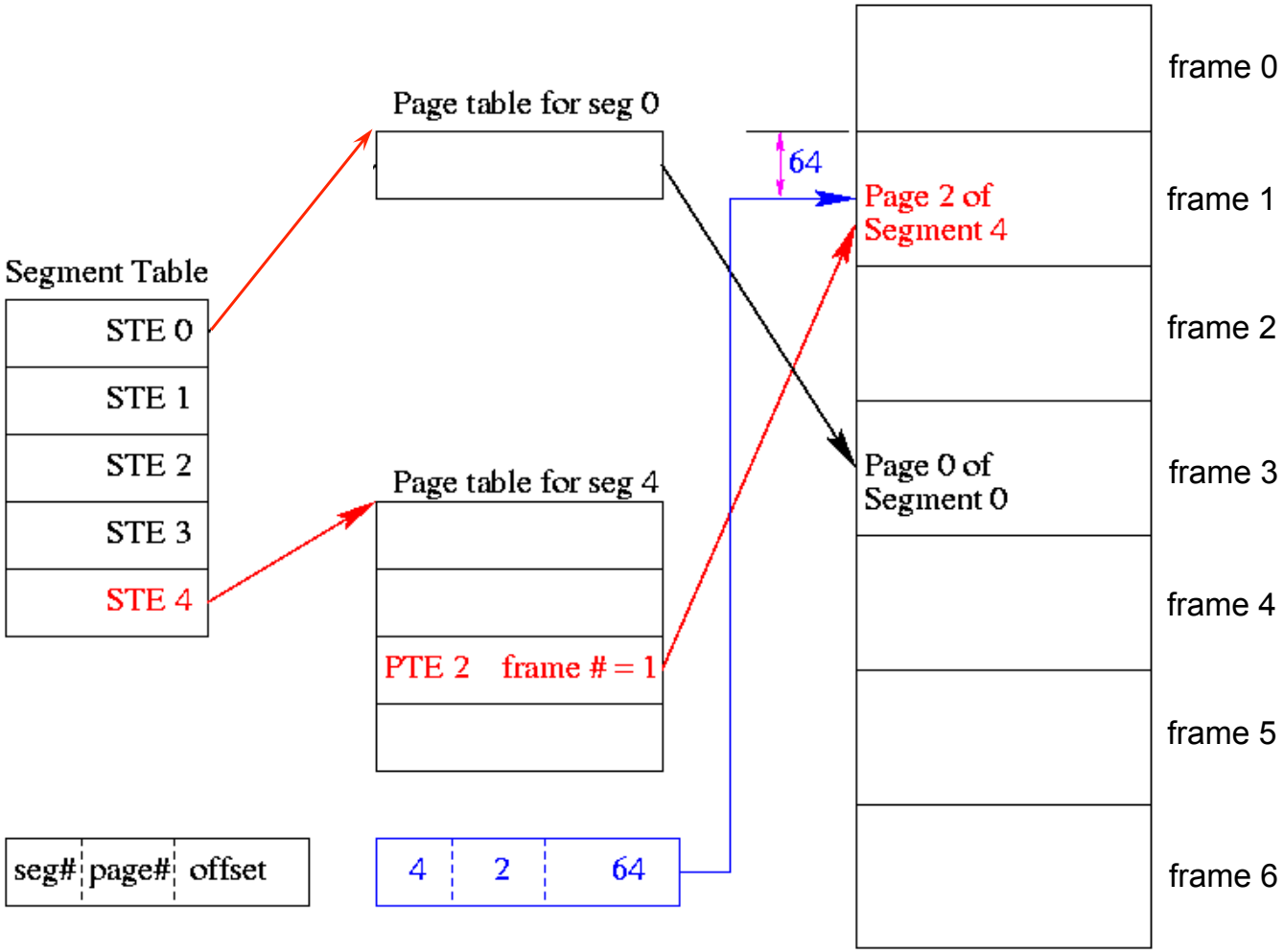
Kết Hợp Phân Trang Và Phân Đoạn

- Nhằm giải quyết trường hợp bảng phân đoạn quá lớn.
- Ý tưởng: phân đoạn bảng phân trang (segmented paging)
  - Mỗi tiến trình sẽ có 1 bảng phân đoạn và nhiều bảng phân trang.
  - Mỗi mục bảng phân đoạn sẽ ánh xạ vào 1 bảng phân trang.
  - Cấu trúc một địa chỉ luận lý: $\langle seg\_number, page\_number, offset \rangle$
    - *seg_number*: chỉ số của các mục trong bảng phân đoạn, dùng để xác định địa chỉ bảng phân trang.
    - *page_number*: chỉ số các mục trong bảng phân trang, dùng để xác định số khung.
    - *offset*: xác định địa chỉ nhớ vật lý.

Dịch Địa Chỉ


Virtual Address
Seg #
Page #
Offset
Seg Table Ptr
Segment Table
S#
Page Table
P#
Frame #
Offset
Offset
Page Frame
Program
Segmentation
Paging
Main Memory

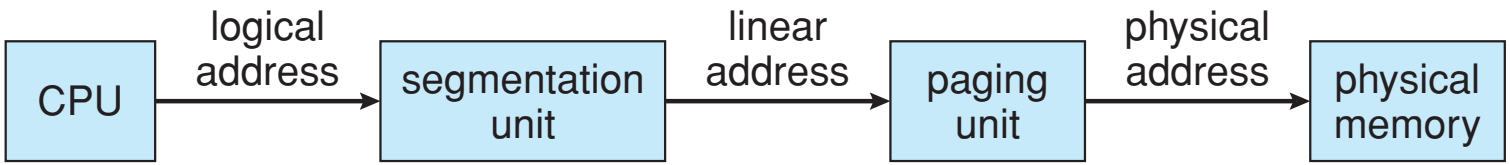
Ví Dụ


Page table for seg 0
64
Page 2 of Segment 4
frame 0
frame 1
frame 2
frame 3
frame 4
frame 5
frame 6
Segment Table
STE 0
STE 1
STE 2
STE 3
STE 4
Page table for seg 4
Page 0 of Segment 0
PTE 2 frame # = 1
seg# page# offset
4 2 64

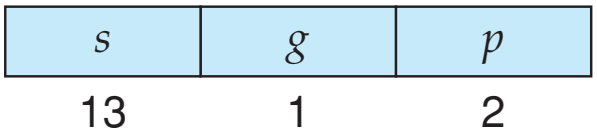
Phụ Lục – Ví Dụ

Phân Trang Trong BXL Intel Pentium

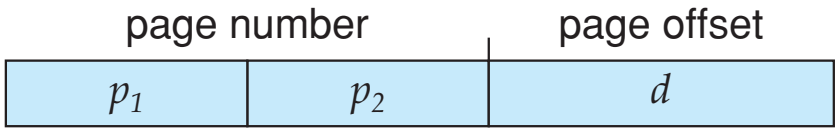
- Sử dụng 2 sơ đồ: phân đoạn và kết hợp phân đoạn với phân trang.
- Sơ đồ dịch địa chỉ và phân đoạn:



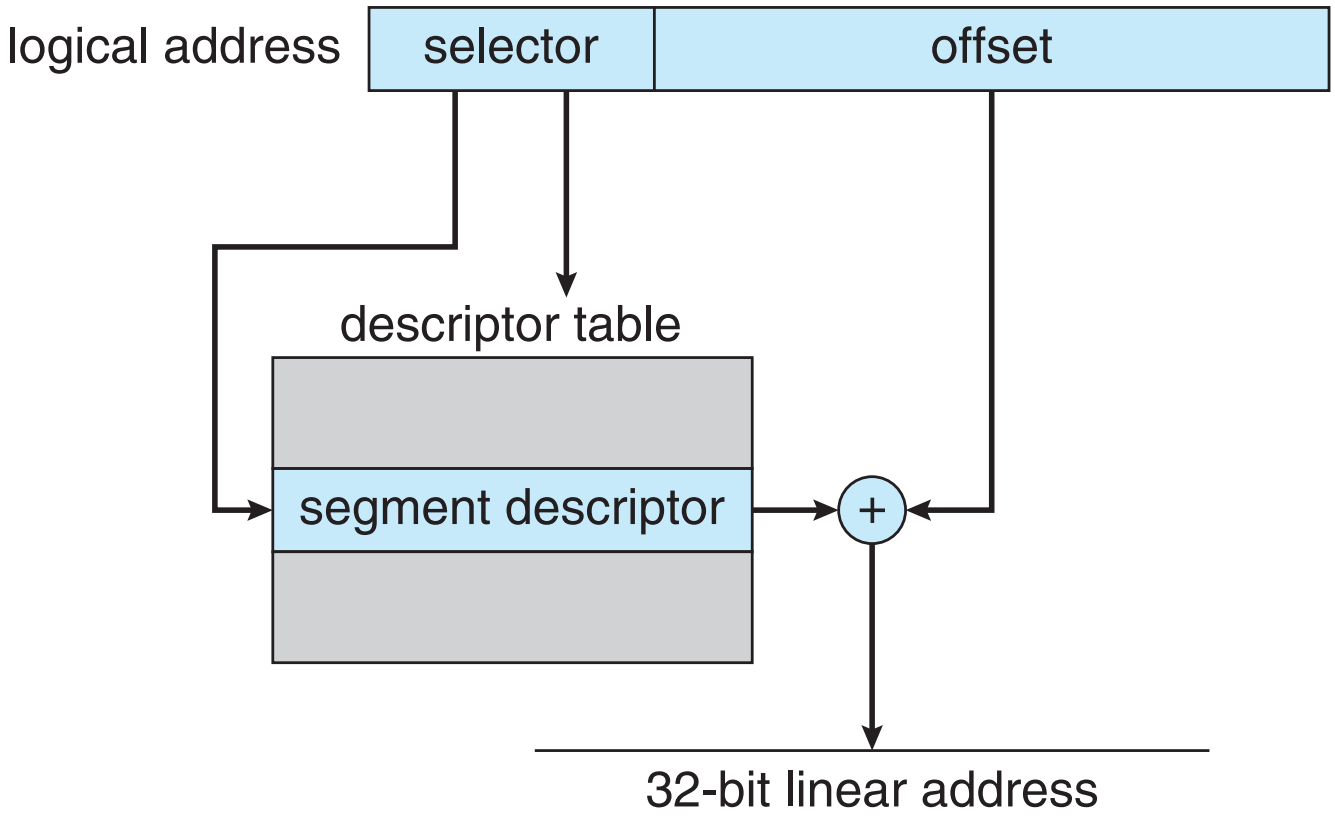
- Địa chỉ luận lý (segmentation):



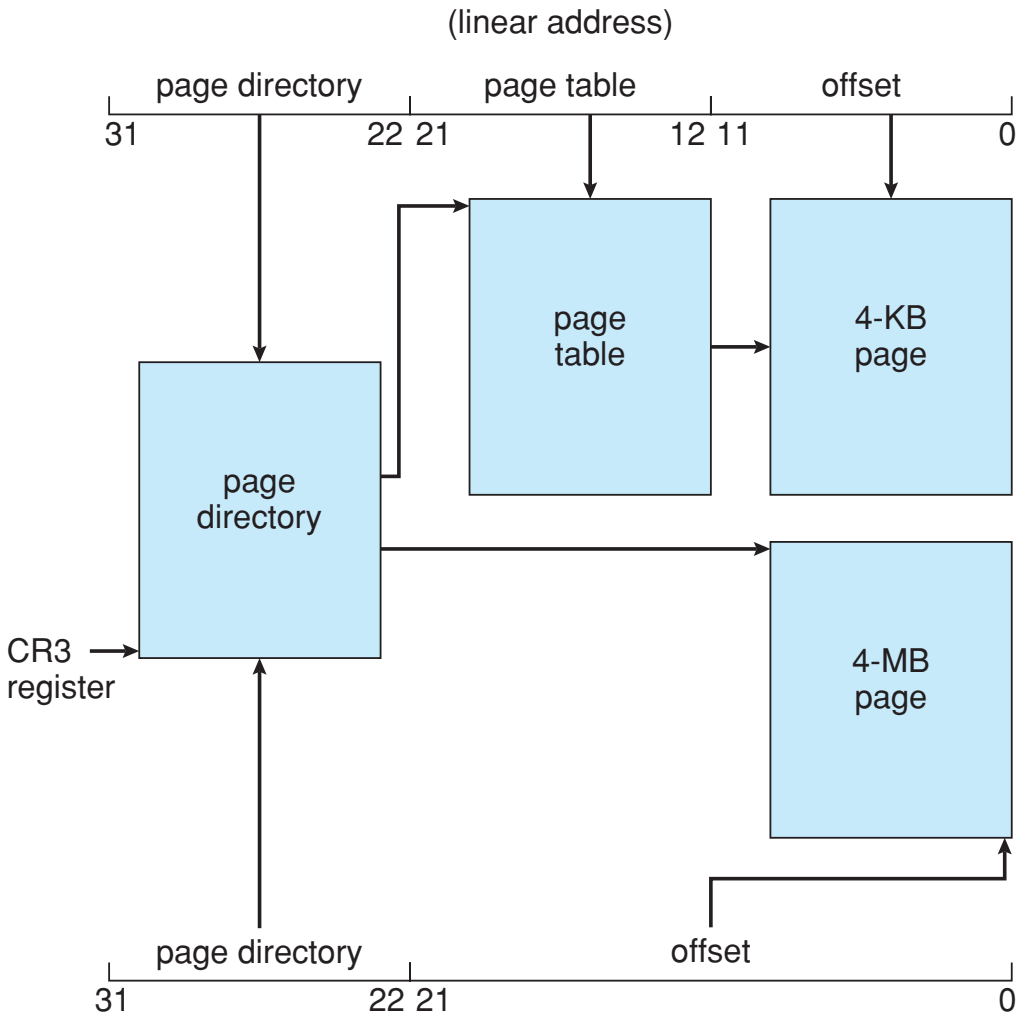
- Địa chỉ tuyến tính (paging):



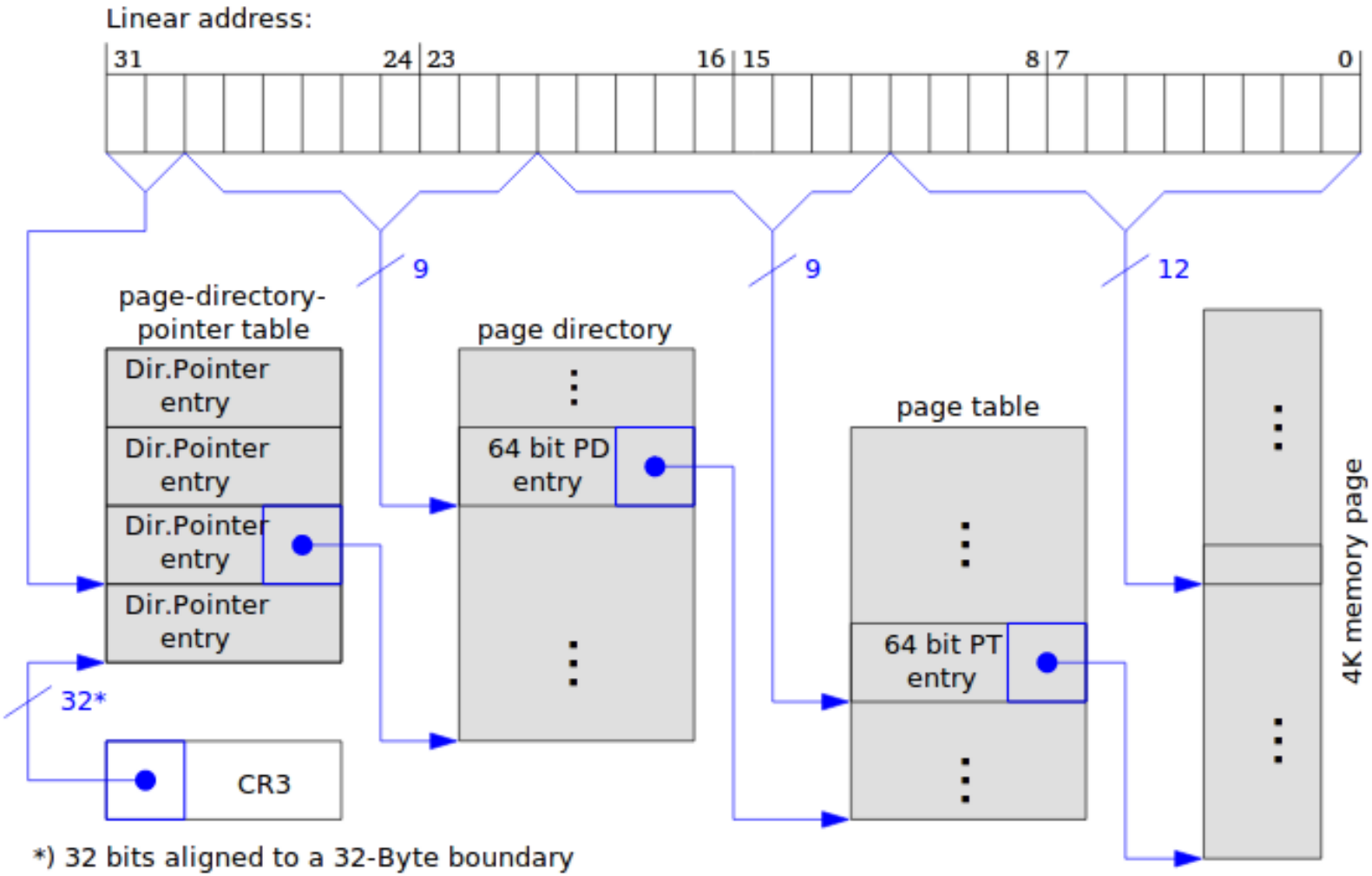
Phân Đoạn


logical address
selector
offset
descriptor table
segment descriptor
+
32-bit linear address

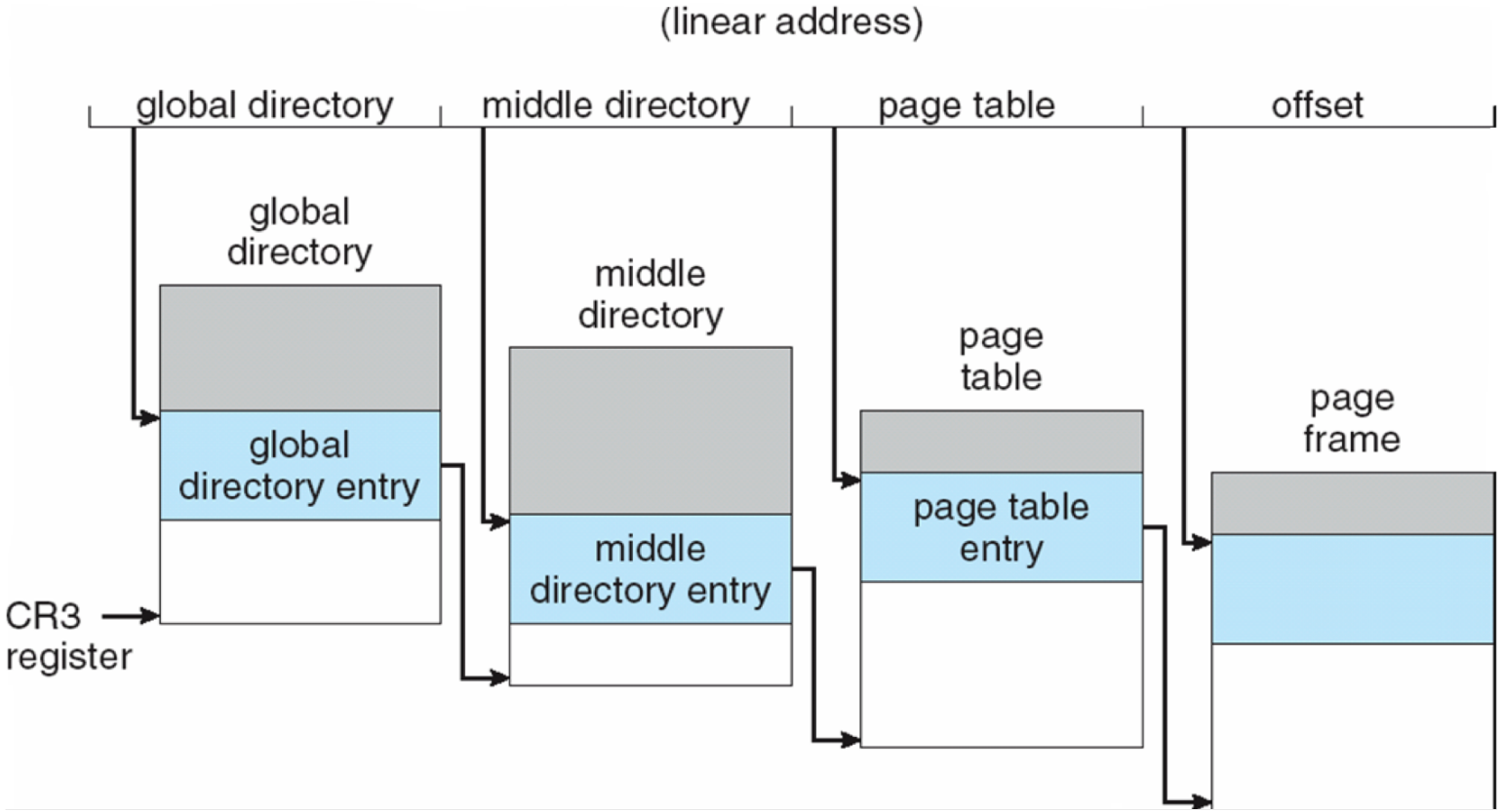
Phân Trang


(linear address)
page directory
page table
offset
31
22 21
12 11
0
page directory
page table
4-KB page
4-MB page
CR3 register
page directory
offset
31
22 21
0

Mở Rộng Địa Chỉ PAE



Phân Trang 3 Cấp Trên Linux




FINISH