HỆ ĐIỀU HÀNH
(OPERATING SYSTEM)

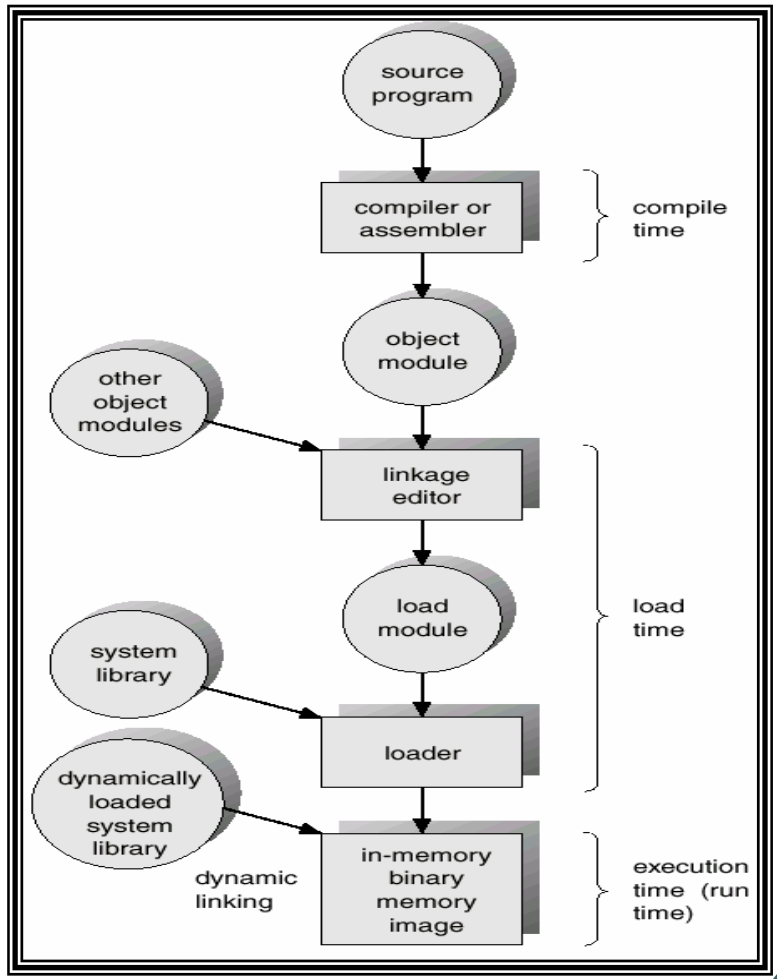
Trình bày:Nguyễn Hoàng Việt
Khoa Công Nghệ Thông Tin
Đại Học Cần Thơ

Chương 7: Quản lý bộ nhớ

- Tổng quan
- Hoán vị (Swapping)
- Định vị kề nhau (Contiguous Allocation)
- Phân trang (Paging)
- Phân đoạn (Segmentation)
- Phân đoạn với phân trang (Segmentation with Paging)

Tổng quan (1)

- Chương trình phải được mang vào bộ nhớ và được đặt vào một quá trình để thực thi.
- Hàng đợi vào (Input queue): tập hợp các quá trình trên đĩa đang chờ để được mang vào bộ nhớ để chạy.
- Chương trình người dùng phải qua vài bước trước khi được chạy thực sự.
  - Việc định địa chỉ theo các phương pháp khác nhau qua một số bước:
    - ✓ Các địa chỉ tượng trưng trong chương trình nguồn.
    - ✓ Các địa chỉ tái định vị khi biên dịch.
    - ✓ Các địa chỉ tuyệt đối khi nạp (loading) hoặc nối kết (linking).
- Gắn kết (binding) là việc ánh xạ từ một không gian địa chỉ đến một không gian địa chỉ khác.



Tổng quan (2)
Gắn kết các chỉ thị và dữ liệu vào bộ nhớ

Việc gắn kết địa chỉ (address binding) của các chỉ thị và dữ liệu vào địa chỉ bộ nhớ có thể diễn ra tại 3 giai đoạn khác nhau:

- **Thời điểm biên dịch (compile time)**: Nếu vị trí vùng nhớ được biết trước, thì có thể sinh ra mã lệnh tuyệt đối (absolute code); tuy nhiên chương trình phải được biên dịch lại nếu vị trí bắt đầu của vùng nhớ thay đổi.
- **Thời điểm nạp (load time)**: trình biên dịch phải sinh ra mã lệnh có thể tái định vị (relocatable code) nếu không thể biết được vị trí vùng nhớ tại thời điểm biên dịch.
- **Thời điểm thực thi (execution time)**: Việc gắn địa chỉ bị trì hoãn cho đến thời điểm thực thi nếu quá trình có thể di chuyển được từ đoạn nhớ (segment) này đến đoạn nhớ khác khi thực thi. Cần thêm sự hỗ trợ của phần cứng để ánh xạ địa chỉ (ví dụ như các thanh ghi cơ sở (base) và giới hạn (limit)).

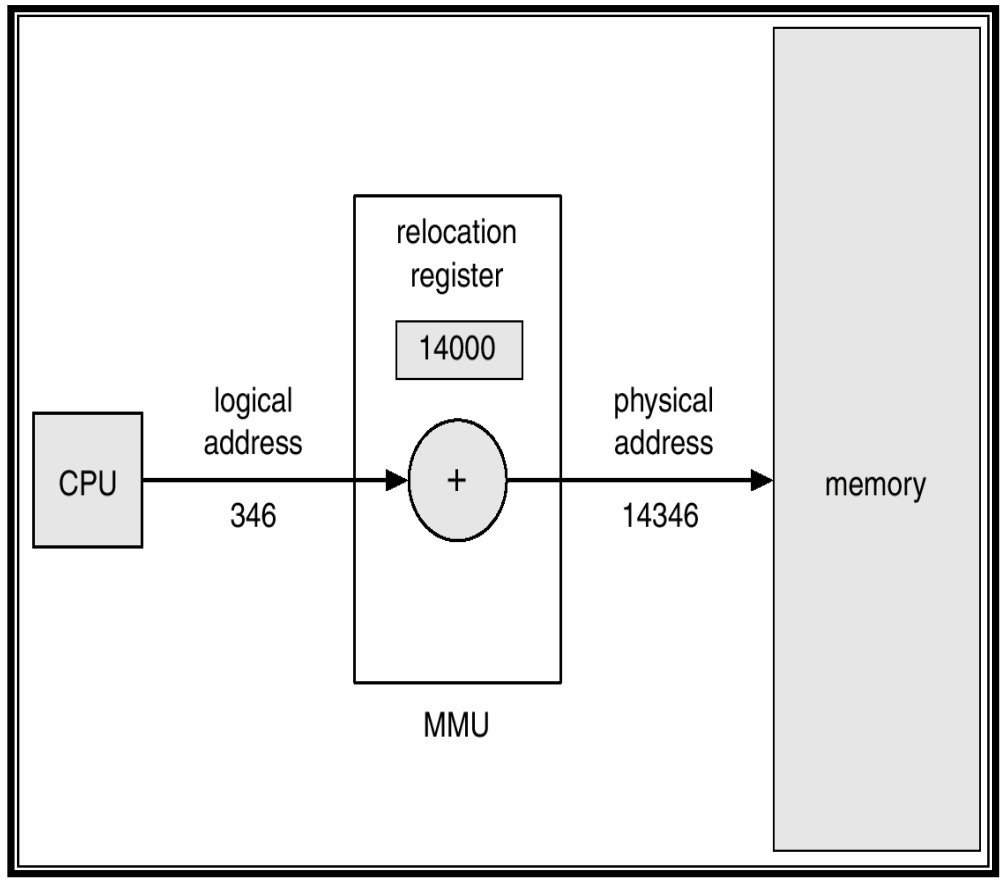
Tổng quan (3)
Không gian địa chỉ vật lý và luận lý

- Khái niệm về **không gian địa chỉ luận lý** được gắn kết (bind) vào **không gian địa chỉ vật lý**, chính là trọng tâm của cơ chế quản lý bộ nhớ.
  - **Địa chỉ luận lý (logical address)**: được sinh ra bởi CPU, cũng được xem là địa chỉ ảo (virtual address).
  - **Địa chỉ vật lý (physical address):** địa chỉ được nhìn thấy bởi bộ quản lý bộ nhớ.
- Địa chỉ luận lý và vật lý là như nhau trong sơ đồ gắn kết địa chỉ tại thời điểm biên dịch và nạp chương trình.
- Địa chỉ luận lý và địa chỉ vật lý sẽ khác nhau trong sơ đồ gắn kết địa chỉ tại thời điểm thực thi.
- Bộ quản lý bộ nhớ (Memory management Unit – MMU): thiết bị phần cứng làm nhiệm vụ ánh xạ địa chỉ ảo sang địa chỉ vật lý.
  - Sơ đồ MMU đơn giản dùng thanh ghi tái định vị (relocation/base register).

Tổng quan (4)
Tái định vị động (Dynamic Relocation)

- Giá trị trong thanh ghi tái định vị được thêm vào mọi địa chỉ được sinh ra bởi quá trình người dùng tại thời điểm nó được đưa vào bộ nhớ.
- Chương trình người dùng chỉ làm việc với các địa chỉ luận lý; nó chẳng bao giờ thấy được địa chỉ vật lý thực.

Sử dụng thanh ghi tái định vị



Tổng quan (5)
Nạp động (Dynamic Loading)

- Nạp tĩnh (static loading): toàn bộ chương trình và dữ liệu được nạp một lần vào bộ nhớ vật lý cho quá trình để chạy.
- Nạp động (dynamic loading): thường trình (routine) chỉ được nạp khi nó được gọi.
  - Tất cả các thường trình được lưu trên dĩa theo dạng nạp có thể tái định vị.
  - Bộ liên kết tái định vị (relocatable linking loader) được dùng để nạp các thường trình mong muốn.
- Điều này làm tăng hiệu năng sử dụng bộ nhớ; các thường trình không được sử dụng sẽ không bao giờ được nạp.
- Hữu ích khi một số lượng lớn các mã lệnh được cần để giải quyết các tình huống không thường xuất hiện.
- Không cần sự hỗ trợ đặc biệt của hệ điều hành, nạp động được cài đặt thông qua cách thiết kế chương trình (xem lại).

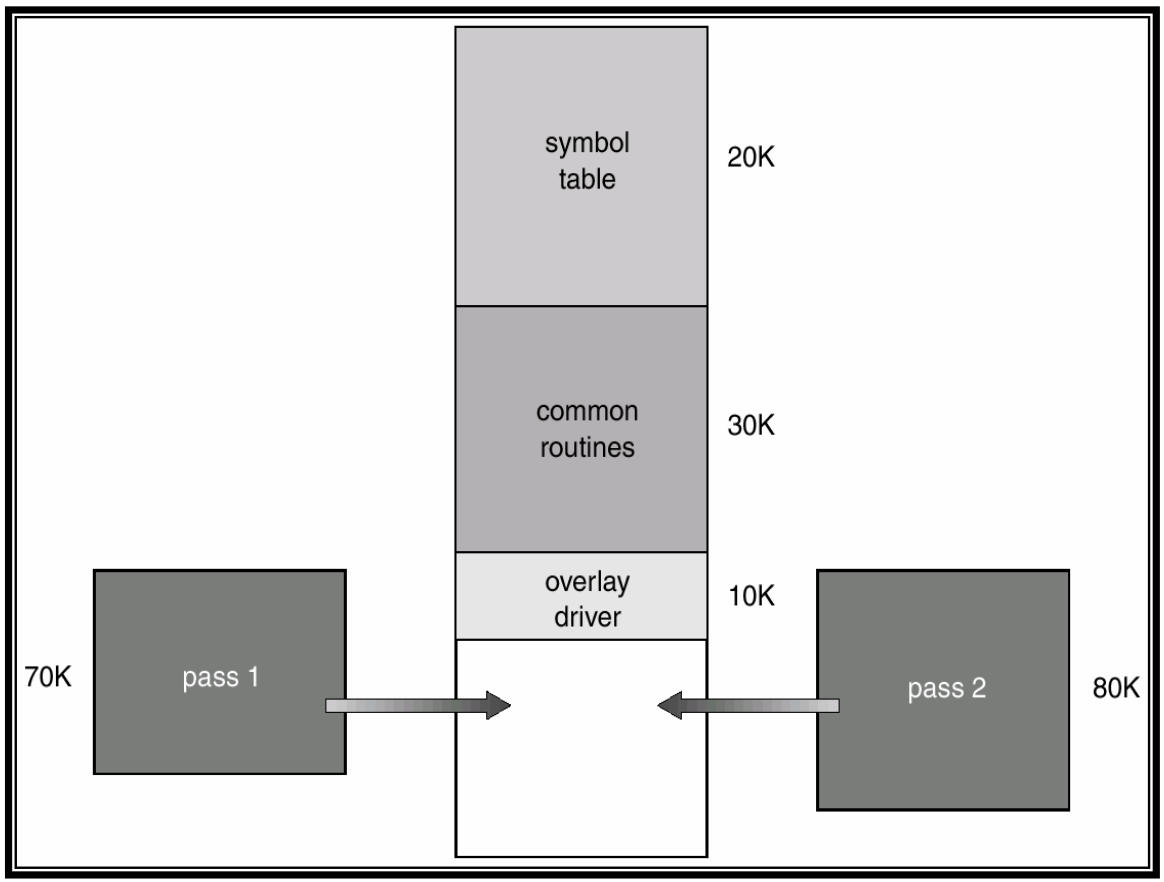
Tổng quan (6)
Liên kết động (Dynamic Linking)

- Liên kết tĩnh (static linking): các thư viện hệ thống được kết hợp với bộ nạp vào ảnh có thể thực hiện (executable image).
- Liên kết động (dynamic linking): việc liên kết bị hoãn lại đến tận thời điểm thực thi.
  - Stub được kèm vào ảnh cho mỗi tham khảo đến thường trình thư viện (library-routine).
  - Liên kết động hữu ích đặc biệt cho các thư viện – thư viện chia sẻ.
- Một đoạn mã lệnh nhỏ, *stub,* được sử dụng để định vị thường trình thư viện thường trú trong bộ nhớ.
- Stub thay thế chính nó bằng địa chỉ của thường trình và thực thi thường trình.
- Hệ điều hành cần phải kiểm tra thường trình có nằm trong không gian địa chỉ dành cho các quá trình hay không.

Tổng quan (7)
Các phần phủ lắp (Overlays)

- Cơ chế phủ lắp cho phép quá trình lớn hơn lượng bộ nhớ cấp phát cho nó.
- Ý tưởng: giữ trong bộ nhớ chỉ những chỉ thị và dữ liệu cần thiết tại một thời điểm cụ thể.
  - Khi cần các chỉ thị khác, chúng nó được nạp vào không gian bị chiếm bởi trước đây bởi những chỉ thị và dữ liệu không còn cần thiết nữa.
- Được cài đặt bởi người dùng:
  - Không cần sự hỗ trợ đặc biệt từ hệ điều hành.
  - Tuy nhiên, thiết kế để lập trình cấu trúc phủ lắp lại phức tạp

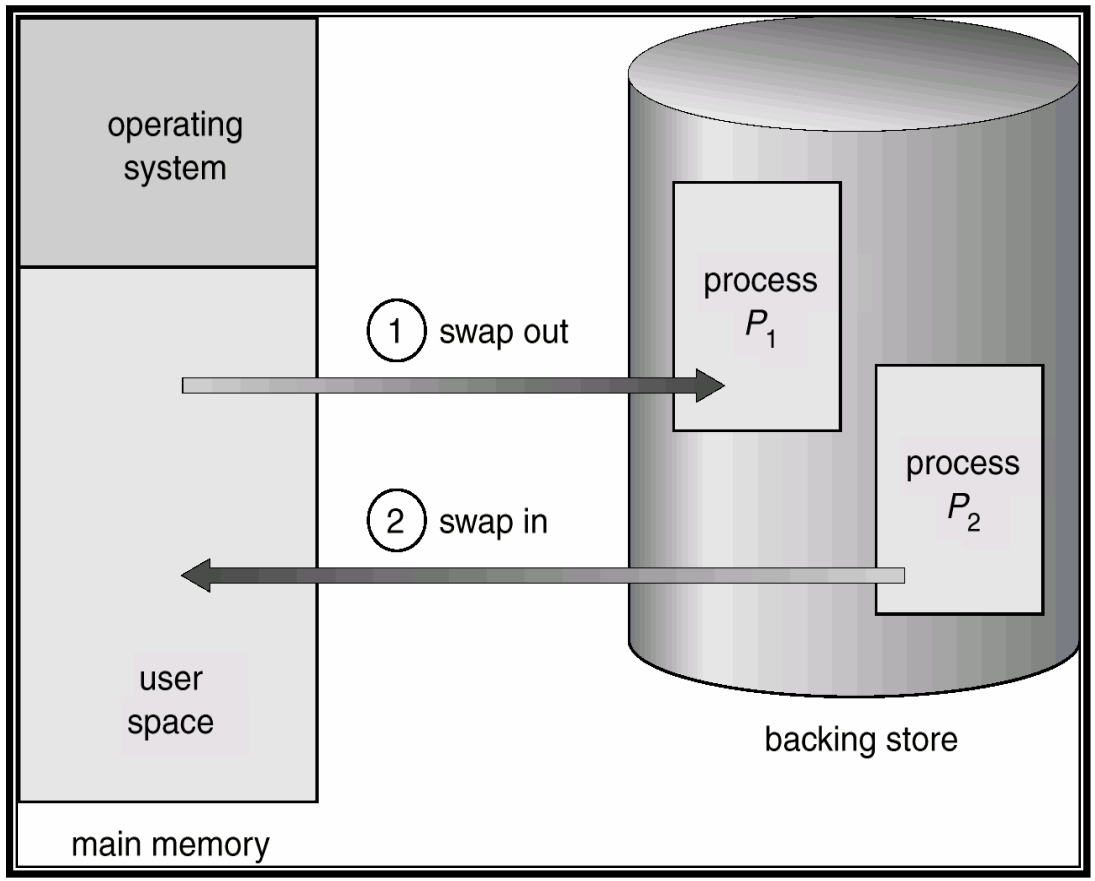
Tổng quan (8)
Ví dụ phần phủ lắp gồm hai phần



Hoán vị (1)
(Swapping)

- Một quá trình có thể được hoán vị tạm thời ra khỏi bộ nhớ, đến thiết bị lưu trữ trợ giúp (backing store) và rồi lại được hoán vị ngược lại bộ nhớ để tiếp tục thực hiện.
- Thiết bị lưu trữ trợ giúp: đĩa tốc độ nhanh đủ lớn để sao chép tất cả ảnh bộ nhớ (memory image) cho tất cả người dùng; phải cho phép truy cập trực tiếp đến các ảnh này.
- **Cuộn ra (Roll out), cuộn vào (roll in):** các biến thể của hoán vị dùng cho các giải thuật định thời dựa vào sự ưu tiên; quá trình có độ ưu tiên thấp hơn được chuyển ra để cho quá trình có độ ưu tiên cao hơn có thể được nạp và thực thi.
- Phần chính của thời gian hoán vị là thời gian chuyển dữ liệu; tổng thời gian chuyển dữ liệu tỷ lệ thuận với lượng bộ nhớ được hoán vị.
- Nhiều phiên bản được sửa đổi của kỹ thuật hoán vị được tìm thấy trong nhiều hệ thống khác nhau, ví dụ Unix, Linux, và Windows.

Hoán vị (2)
Sơ đồ quan sát việc hoán vị

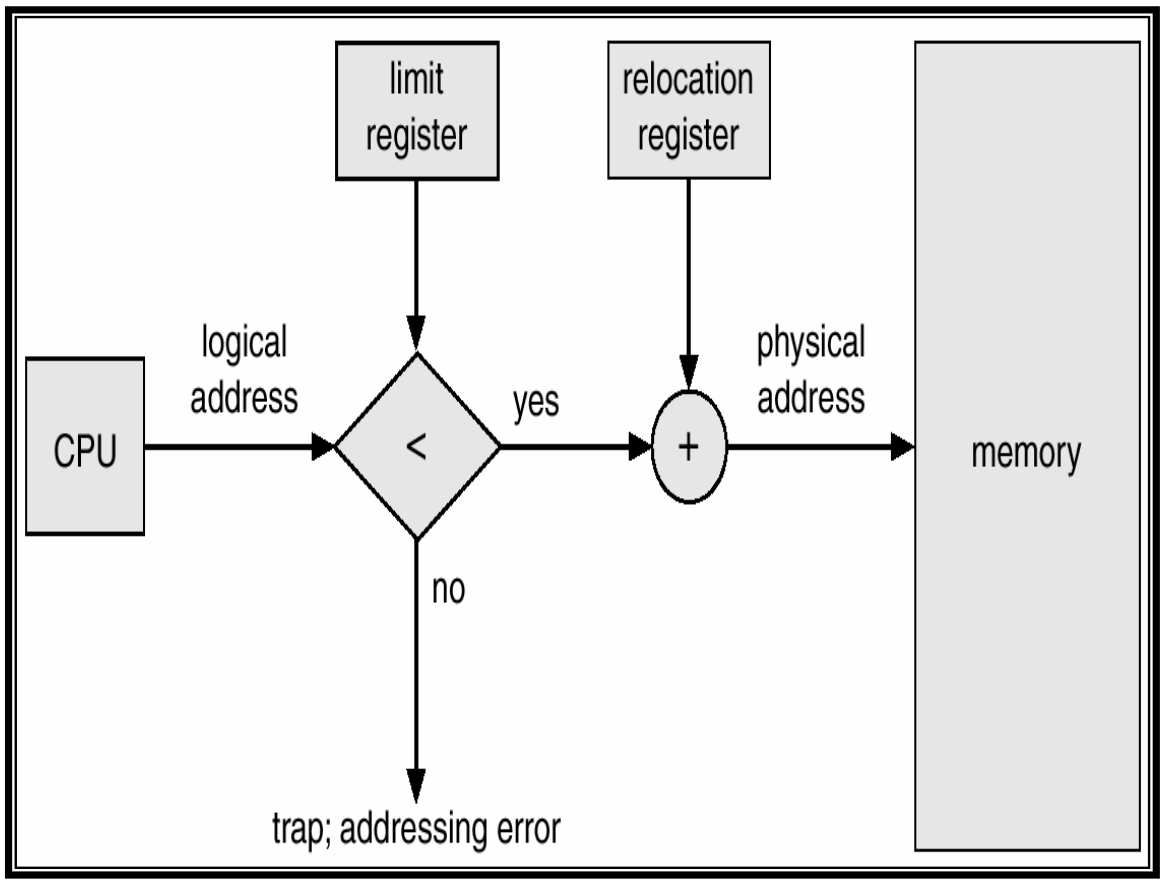


Cấp phát kề nhau (Contiguous Allocation) (1)
Cấp phát đơn phân khu (Single-partition allocation)

- Bộ nhớ chính thường được chia làm 2 phần:
  - Phần thường trú của hệ điều hành: thường được tổ chức trong vùng nhớ thấp với các vector ngắt.
  - Các quá trình người dùng được tổ chức trong vùng nhớ cao.
- Cấp phát đơn phân khu (Single-partition allocation)
  - Sơ đồ thanh ghi tái định vị được sử dụng để bảo vệ các quá trình người dùng với nhau, và để chống việc thay đổi mã lệnh và dữ liệu của hệ điều hành.
  - Thanh ghi tái định vị (relocation register) chứa giá trị của địa chỉ vật lý nhỏ nhất; thanh ghi giới hạn (limit register) chỉ ra phạm vi cho phép của các địa chỉ luận lý – mỗi địa chỉ luận lý phải nhỏ hơn giá trị trong thanh ghi giới hạn.
  - MMU ánh xạ động các địa chỉ luận lý bằng cách cộng giá trị trong thanh ghi tái định vị; địa chỉ ánh xạ được gởi đến bộ nhớ.

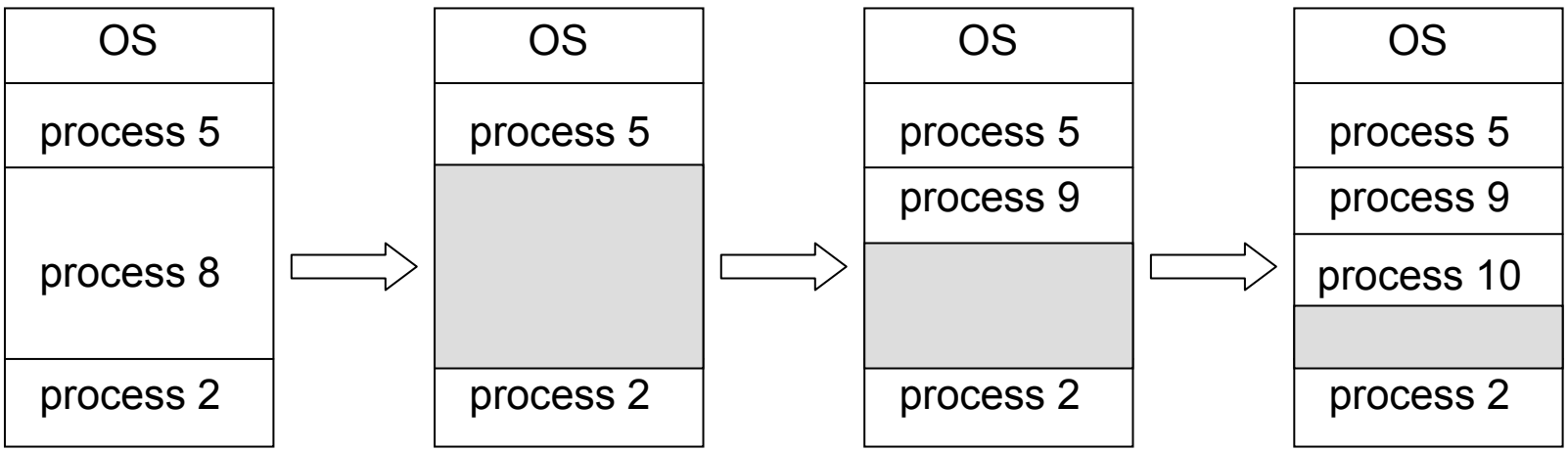
Cấp phát kề nhau (2)

Hỗ trợ phần cứng cho việc tái định vị và thanh ghi giới hạn



Cấp phát kề nhau (3)
Cấp phát đa phân khu (Multiple-partition allocation)

- Cấp phát đa phân khu:
  - **Lỗ hổng (hole):** là khối bộ nhớ chưa được dùng; các lỗ hổng với kích thước khác nhau nằm rải rác trong bộ nhớ.
  - Khi một quá trình xuất hiện, nó được cấp cho lỗ hổng đủ chứa nó.
  - Hệ điều hành duy trì thông tin về:
    - ✓ Những phân khu đã được cấp
    - ✓ Các lỗ hổng



Cấp phát kề nhau (4)
Vấn đề cấp phát bộ nhớ động

Làm thế nào để thỏa mãn yêu cầu bộ nhớ có kích thước *n* từ số các lỗ hổng hiện tại?

- **First-fit**: cấp lỗ hổng đầu tiên đủ lớn.
- **Best-fit**: cấp lỗ hổng nhỏ nhất nhưng đủ lớn; phải tìm kiếm toàn bộ danh sách các lỗ hổng, trừ khi danh sách này được sắp thứ tự theo kích thước lỗ hổng. Việc này có khi lại để lại lỗ hổng nhỏ nhất!
- **Worst-fit**: cấp lỗ hổng lớn nhất; cũng phải tìm cho ra lỗ hổng có kích thước lớn nhất trong toàn bộ danh sách. Việc này có khi để lại lỗ hổng lớn nhất!

First-fit và best-fit thì tốt hơn worst-fit khi ta quan tâm đến tốc độ và hiệu suất sử dụng bộ nhớ.

Cấp phát kề nhau (5)
Sự phân mảnh (Fragmentation)

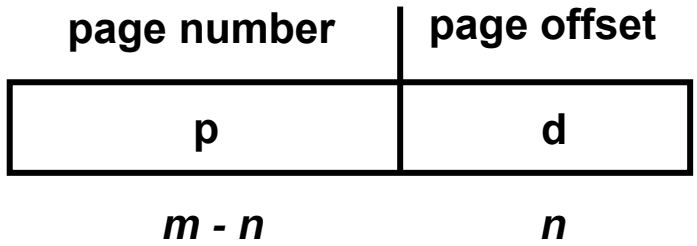
- **Phân mảnh ngoài:** tổng không gian bộ nhớ còn đủ để thỏa mãn yêu cầu nhưng chúng không nằm liền nhau.
- **Phân mảnh trong:** phần bộ nhớ cấp cho quá trình lớn hơn bộ nhớ yêu cầu một ít. Phần sai khác này nằm bên trong của phân khu và không được sử dụng.
- Khử phân mảnh ngoài bằng cách cô đặc bộ nhớ
  - Sắp xếp lại nội dung bộ nhớ để gom các lỗ hổng lại thành một lỗ hổng duy nhất lớn hơn.
  - Sự khử phân mảnh chỉ có thể làm được khi việc tái định vị là động và được làm tại thời điểm thực thi.

Phân trang (1) – Cấp phát không kề nhau (Paging)

- Không gian địa chỉ luận lý của một quá trình có thể không liên tục; quá trình được cấp bộ nhớ vật lý khi nó sẵn dùng.
- Chia bộ nhớ vật lý thành các khối có kích thước cố định gọi là các **khung trang/khung** (page frame/frame): kích thước là lũy thừa của 2, giữa 512 bytes và 8192 bytes.
- Chia bộ nhớ luận lý thành các khối có cùng kích thước với khung trang gọi là **trang** (page).
- Theo dõi tất cả các khung trang còn rảnh.
- Để chạy một chương trình có *n* trang, cần phải tìm đúng *n* khung trang còn trống và nạp chương trình vào.
- Cần thiết lập một **bảng trang** (page table) để dịch các địa chỉ luận lý thành địa chỉ vật lý.
- Gây ra phân mảnh trong.

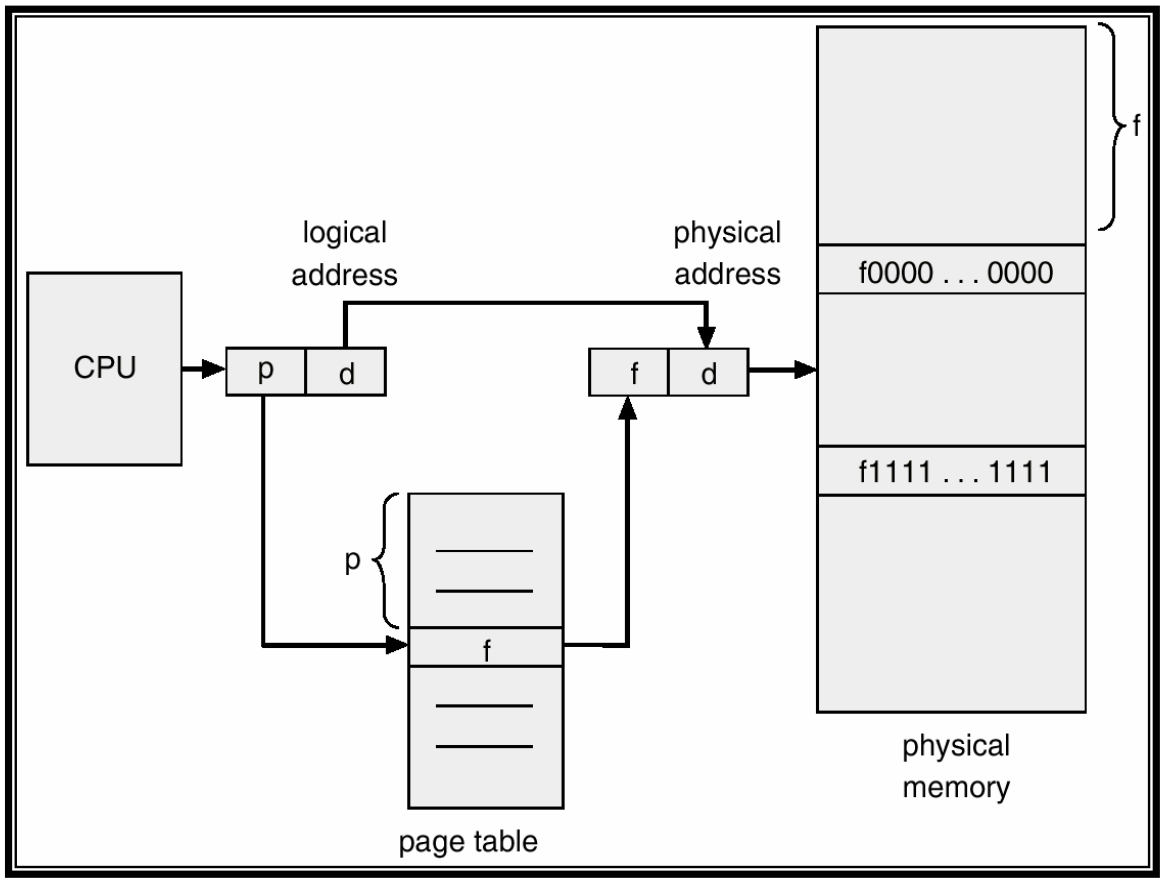
Phân trang (2)
Sơ đồ dịch địa chỉ (Address Translation Scheme)

- Địa chỉ được sinh ra bởi CPU được chia làm 2 phần:
  - **Số hiệu trang – page number (p):** được sử dụng như là một chỉ mục đến bảng trang, mà bảng trang này chứa địa chỉ nền của mỗi trang trong bộ nhớ vật lý.
  - **Offset trong trang – offset number (d):** được kết hợp với địa chỉ nền để xác định địa chỉ vật lý dùng để gởi đến bộ quản lý bộ nhớ.

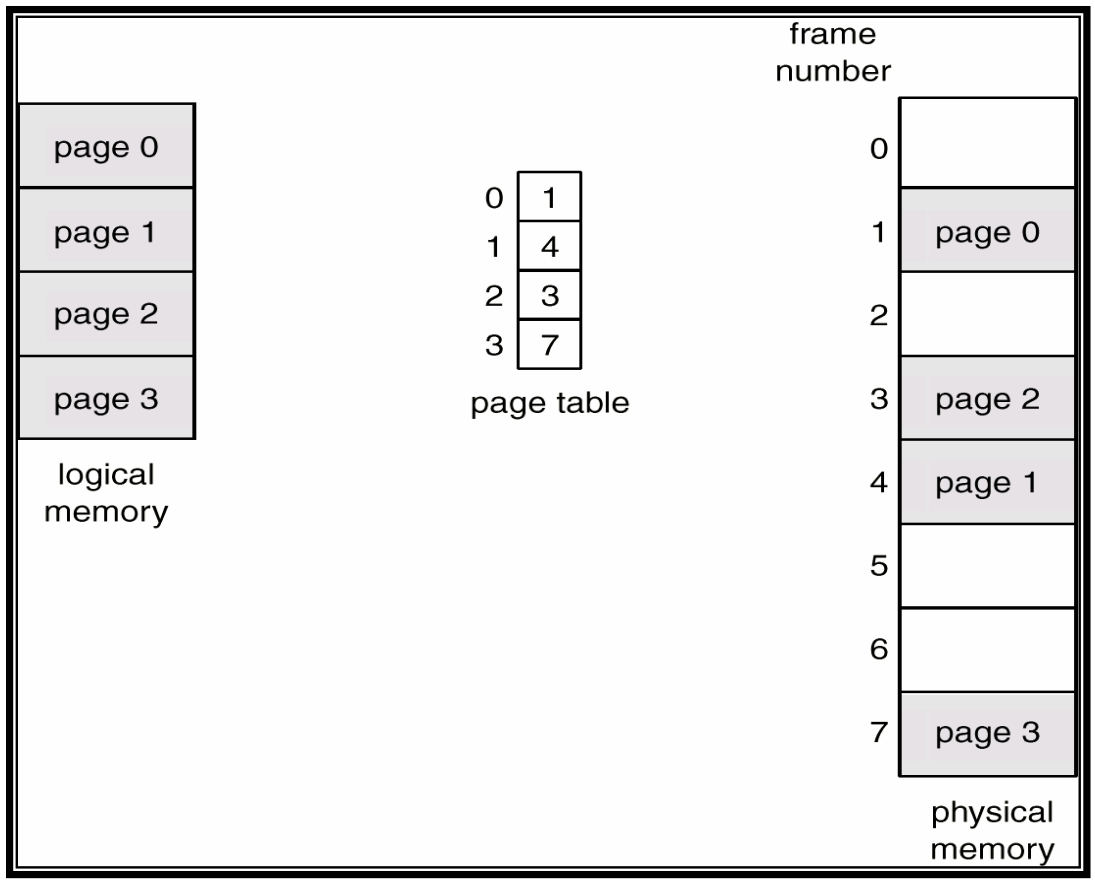


- Kích thước của không gian địa chỉ là 2^m
- Kích thước của trang là 2^n

Phân trang (3)
Cấu trúc hệ thống dịch địa chỉ

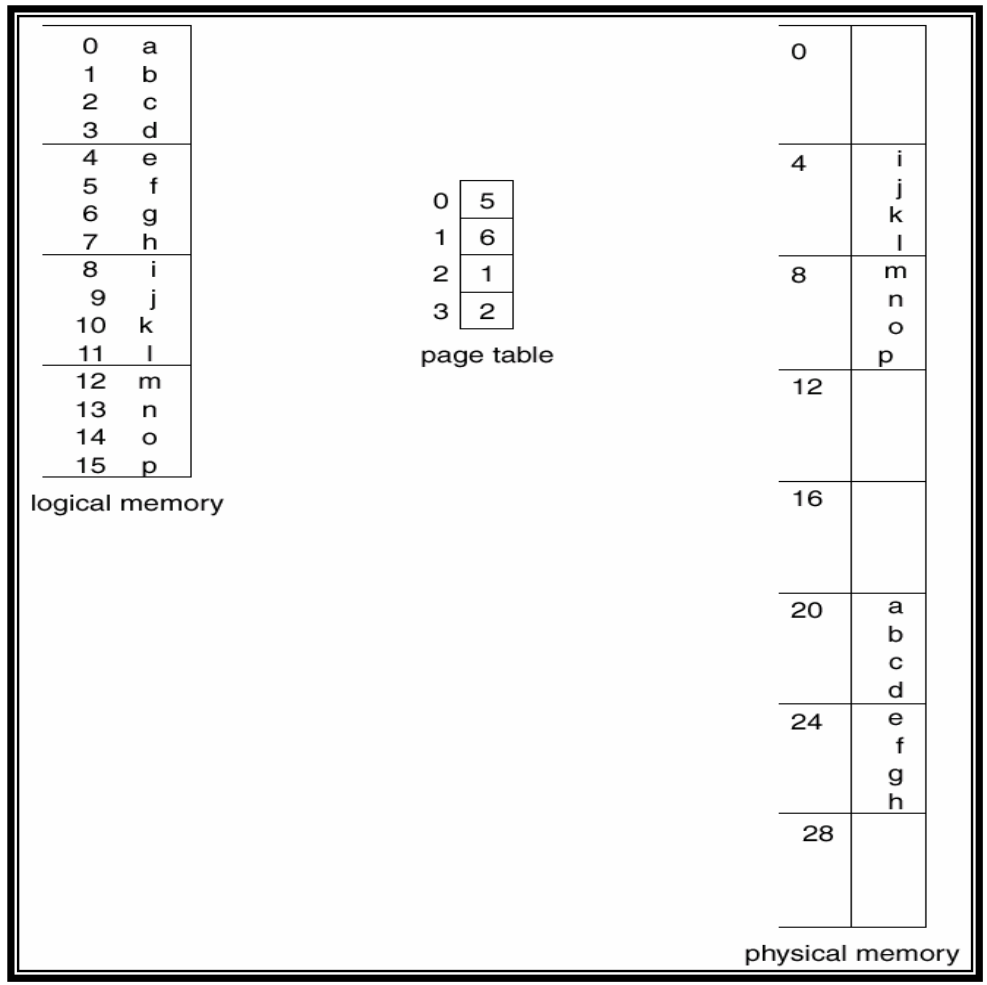


Phân trang (4)
Ví dụ về phân trang



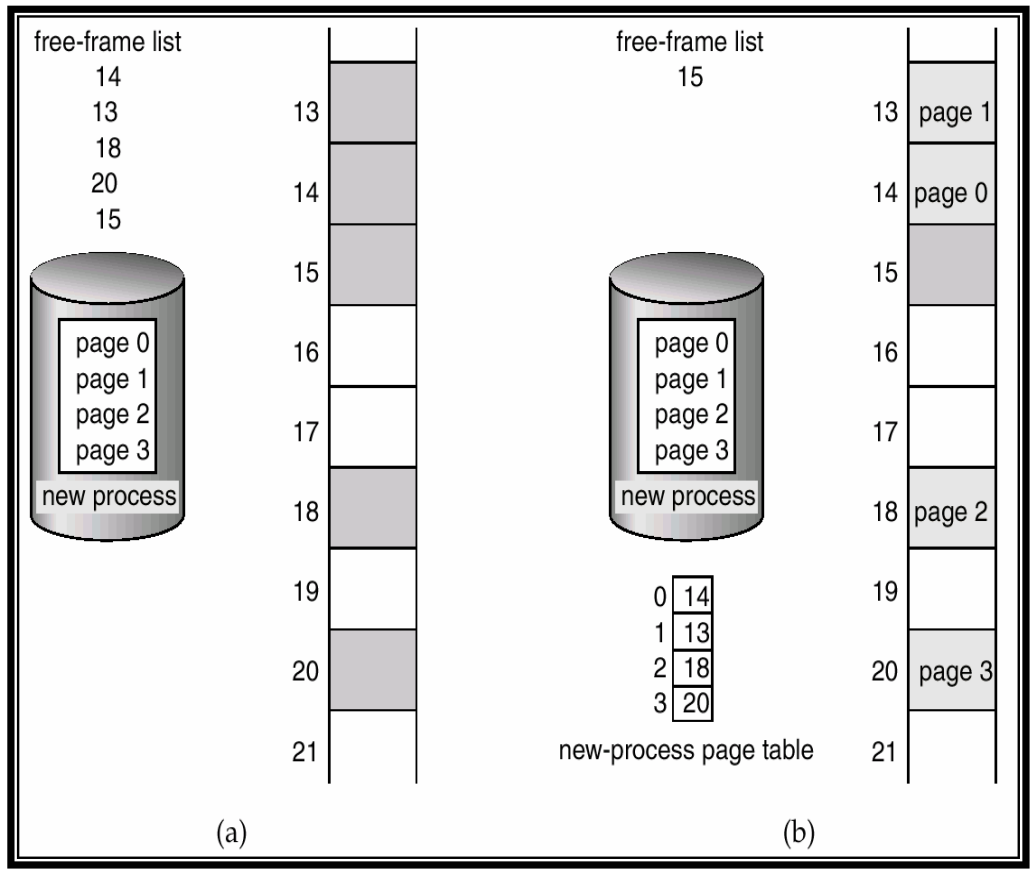
Phân trang (5)
Ví dụ về phân trang

- Ánh xạ từ bộ nhớ luận lý vào bộ nhớ vật lý.
- Một trang kích thước 4 byte.
- Bộ nhớ vật lý kích thước 32 byte (8 khung).
- Địa chỉ luận lý 3 được ánh xạ vào địa chỉ vật lý 23.



Phân trang (6)
Ví dụ về phân trang - Các khung trống

- Theo dõi các khung trống.
- Để chạy chương trình kích thước n trang, cần tìm n khung trống và nạp chương trình.
- Phân mảnh trong: khung cuối cùng được cấp phát có thể không hoàn tòan đầy.



Trước khi cấp phát **Sau khi cấp phát**

Phân trang (7)
Cài đặt bảng trang

- Bảng trang được giữ trong bộ nhớ chính.
- **Thanh ghi nền của bảng trang** (PTBR - Page-table base register) chỉ tới bảng trang.
- **Thanh ghi chỉ chiều dài bảng trang** (PRLR - Page-table length register) chỉ định kích thước của bảng trang.
- Trong sơ đồ này, mỗi truy cập đến dữ liệu/chỉ thị yêu cầu 2 truy cập bộ nhớ: một cho bảng trang và một cho việc truy cập dữ liệu/chỉ thị.
- Vấn đề truy cập bộ nhớ hai lần có thể giải quyết được bằng cách sử dụng một kiểu cache phần cứng tìm kiếm nhanh gọi là bộ nhớ kết hợp hoặc là bộ đệm tìm kiếm phụ cho việc dịch địa chỉ **(TLBs - Translation look-aside buffers)**

Phân trang (8)
Cài đặt bảng trang – Bộ nhớ kết hợp

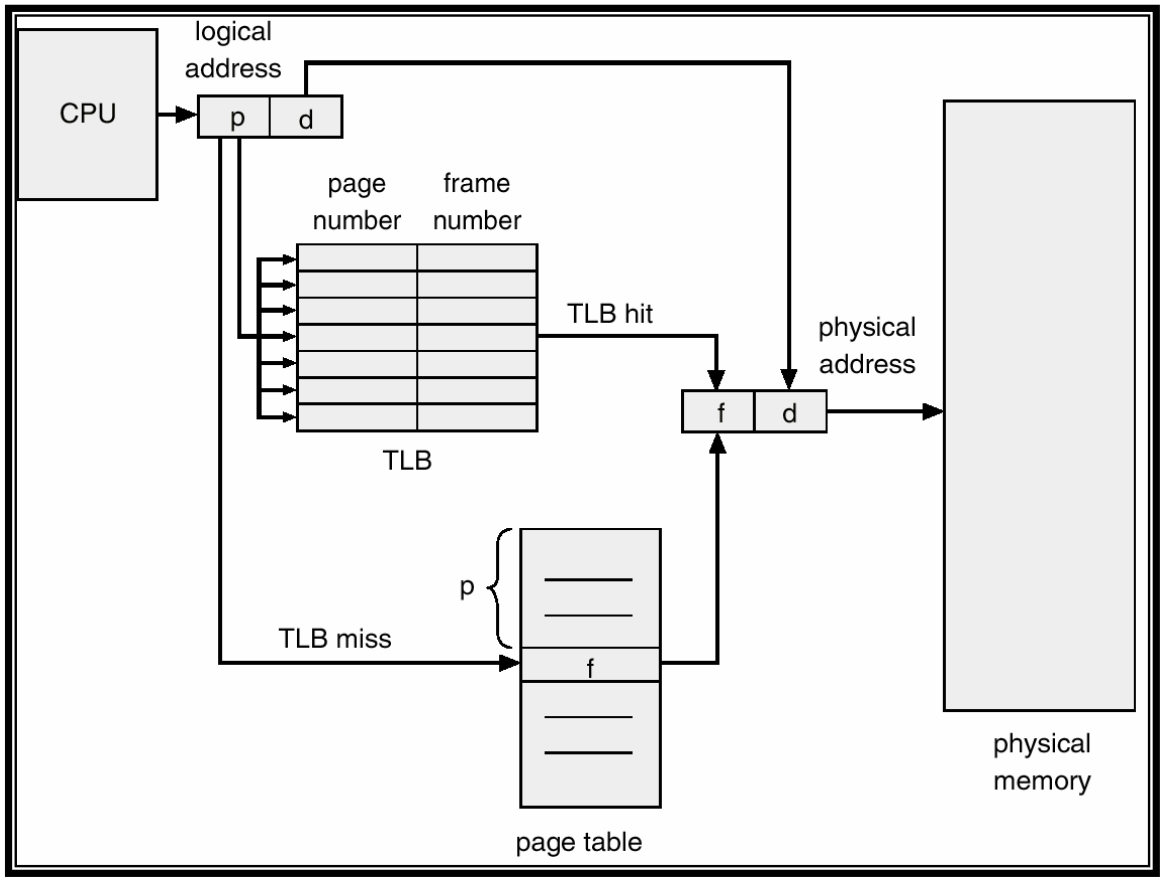
- Bộ nhớ kết hợp (Associative Memory) – tìm kiếm song song

Page #	Frame #

- Dịch địa chỉ (A´, A´´)
  - Nếu A´ đang ở trong thanh ghi kết hợp, lấy ra số hiệu của khung trang.
  - Ngược lại, lấy ra số hiệu khung trang tương ứng từ bảng trang trong bộ nhớ

Phân trang (9)

Cài đặt bảng trang – Phần cứng phân trang với TLB



Phân trang (10)

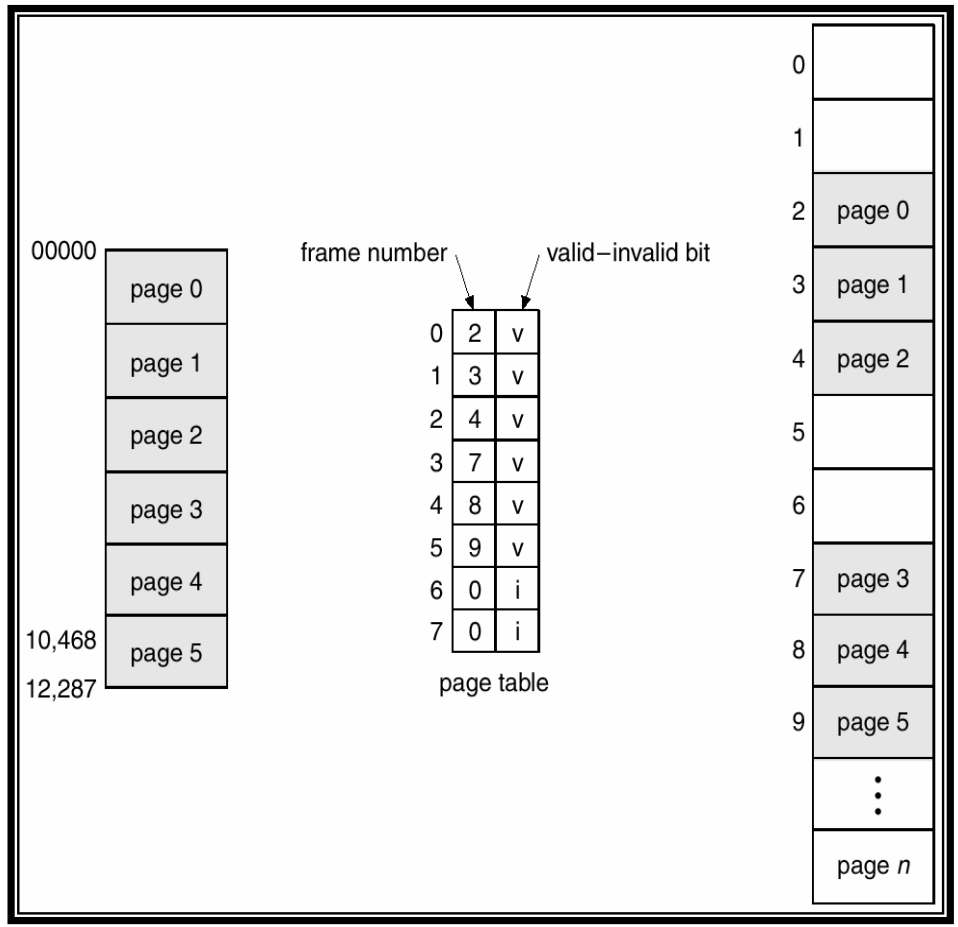
Cài đặt bảng trang – Thời gian truy cập hữu ích

- Thời gian tìm kiếm trên thanh ghi kết hợp = ε đơn vị thời gian
- Giả sử chu kỳ bộ nhớ (thời gian truy cập bộ nhớ) là m
- Tỉ lệ chập (hit ratio): phần trăm số lần số hiệu trang được tìm thấy trong các thanh ghi kết hợp. Tỉ lệ này liên quan đến số lượng thanh ghi kết hợp.
- $\alpha = 0.8$
- $\varepsilon = 20$ ns
- m = 100 ns
- Thời gian truy cập hữu ích (Effective Access Time)

$$EAT = (m + \varepsilon)\,\alpha + (2m + \varepsilon)(1 - \alpha)$$
$$= 2m + \varepsilon - m\alpha$$
$$= 20 + 200 - 80$$
$$= 140 \text{ ns}$$

Phân trang (11)
Bảo vệ bộ nhớ (Memory Protection)

- Việc bảo vệ bộ nhớ được cài đặt bằng cách kết hợp một bit bảo vệ (protection bit) với mỗi khung trang.
- Bit hợp lệ-không hợp lệ (Valid-invalid bit) được gắn vào mỗi đầu mục trong bảng trang:
  - "valid" chỉ ra rằng trang tương ứng đang nằm trong không gian địa chỉ luận lý của quá trình, và do đó là trang hợp lệ.
  - "invalid" chỉ ra rằng trang đó không nằm trong không gian địa chỉ luận lý của quá trình.

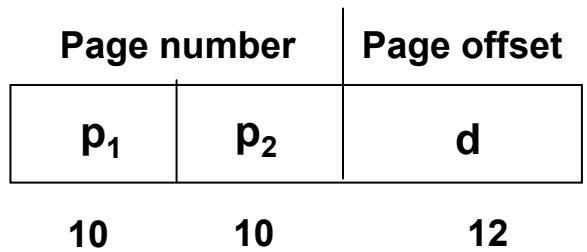


Phân trang (12)
Cấu trúc bảng trang (Page Table Structure)

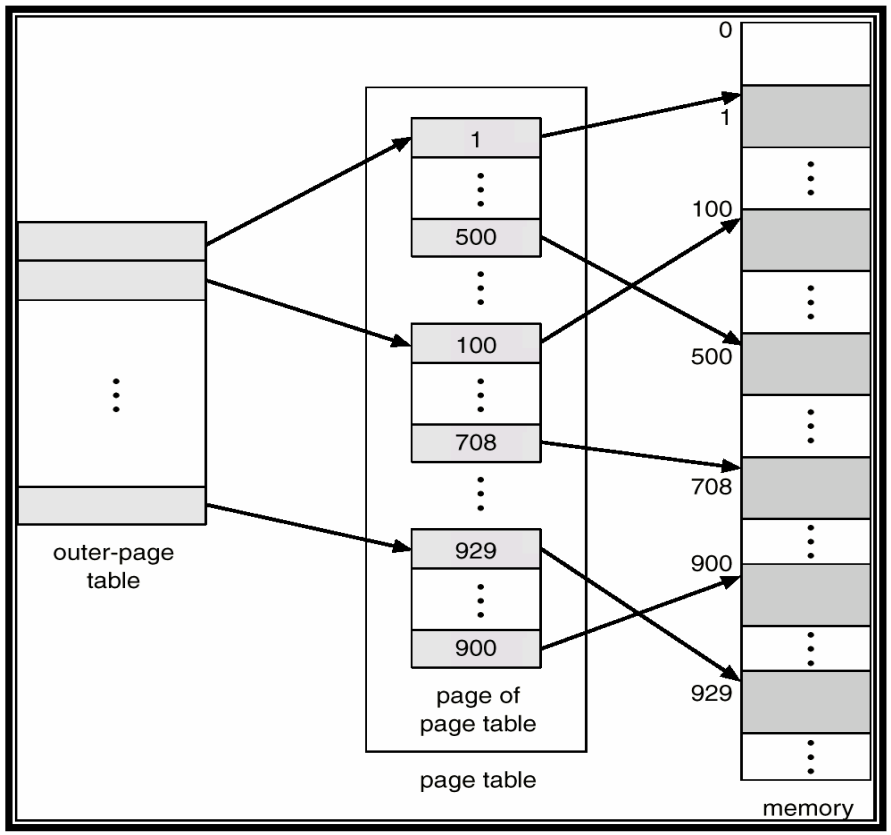
- Bảng trang phân cấp (hierachical page table).
- Bảng trang được băm (hashed page table).
- Bảng trang đảo (inverted page table).

Phân trang (13)
Bảng trang phân cấp (Hierarchical Paging)

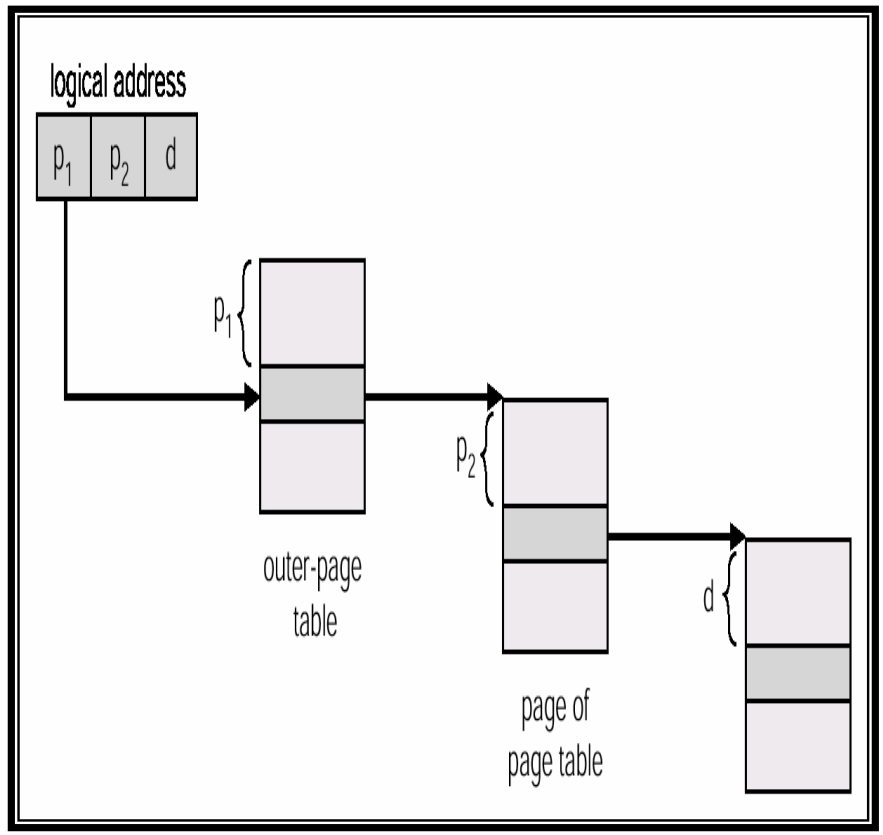
- Phân không gian địa chỉ luận lý vào nhiều bảng trang.
- Một địa chỉ luận lý dài 32 bit. Một trang có kích thước 4K. Địa chỉ luận lý bao gồm:
  - Một số hiệu trang (page number) gồm có 20 bits.
  - Một độ dời (page offset) trong trang có 12 bits.
- Bảng trang 2 mức: bảng trang được phân trang, số hiệu trang lại được chia thành 2 phần:
  - Một số hiệu trang dài 10 bit.
  - Một độ dời trong trang dài 10 bit.
- Do đó, một địa chỉ luận lý có dạng sau:
  - p_1 là chỉ mục đến bảng trang ngoài (outer-page table).
  - p_2 là độ dời bên trong trang của bảng trang ngoài.



Phân trang (14)
Sơ đồ bảng trang hai mức – Dịch địa chỉ



Bảng trang 2 mức

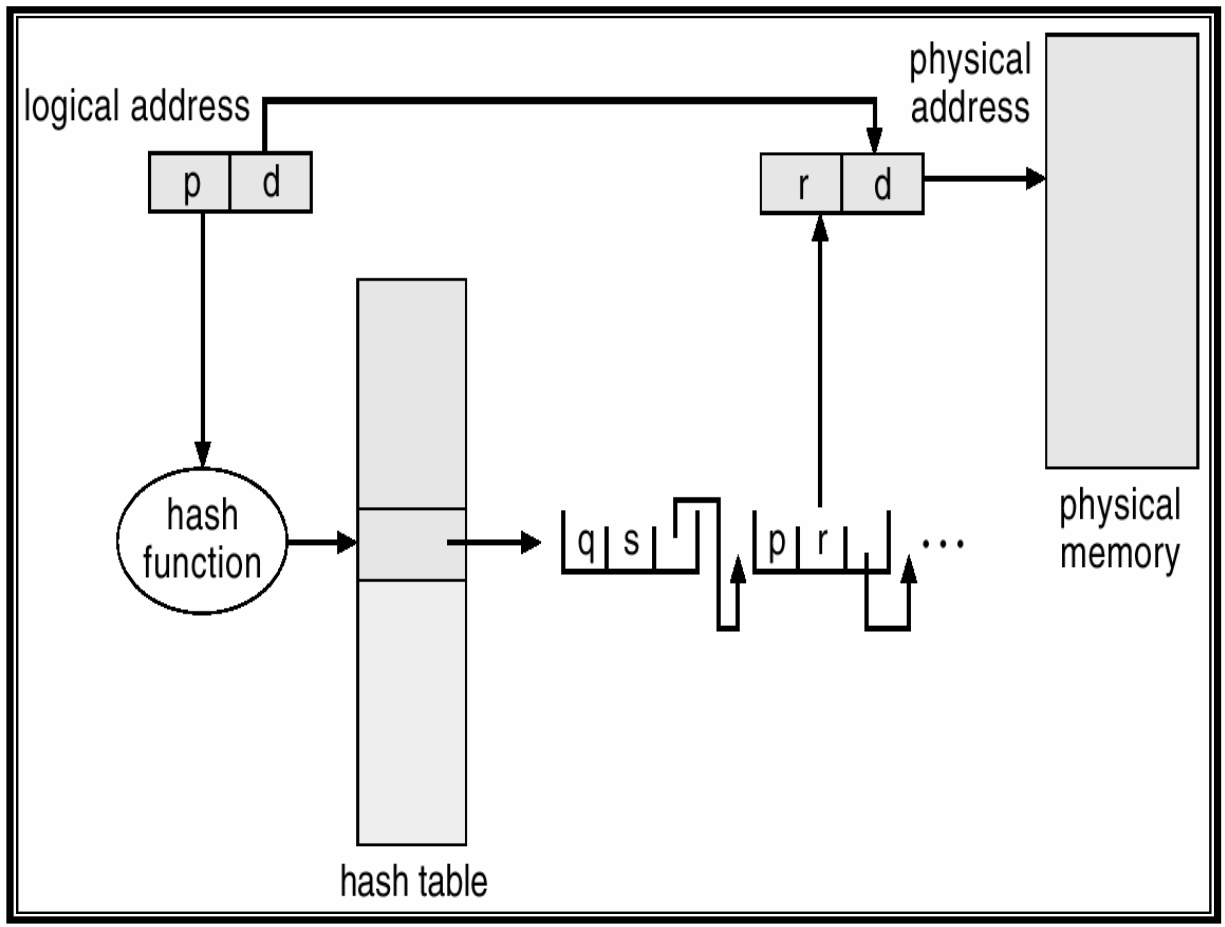


Dịch địa chỉ

Phân trang (15)
Bảng trang được băm (Hashed Page Table)

- Thông dụng trong các không gian địa chỉ > 32 bits.
- Số hiệu của trang ảo được băm vào bảng trang.
- Một mục từ của bảng trang sẽ chỉ đến một danh sách liên kết của các phần tử được băm vào cùng vị trí.
- Một phần tử có ba thông tin chính: số hiệu trang ảo, số hiệu khung trang tương ứng, và con trỏ chỉ đến phần tử kế tiếp trong danh sách liên kết.
- Số hiệu trang ảo được so sánh trong danh sách này để tìm ra một số hiệu trang trùng khớp. Nếu trùng khớp, thì số hiệu khung trang được lấy ra.

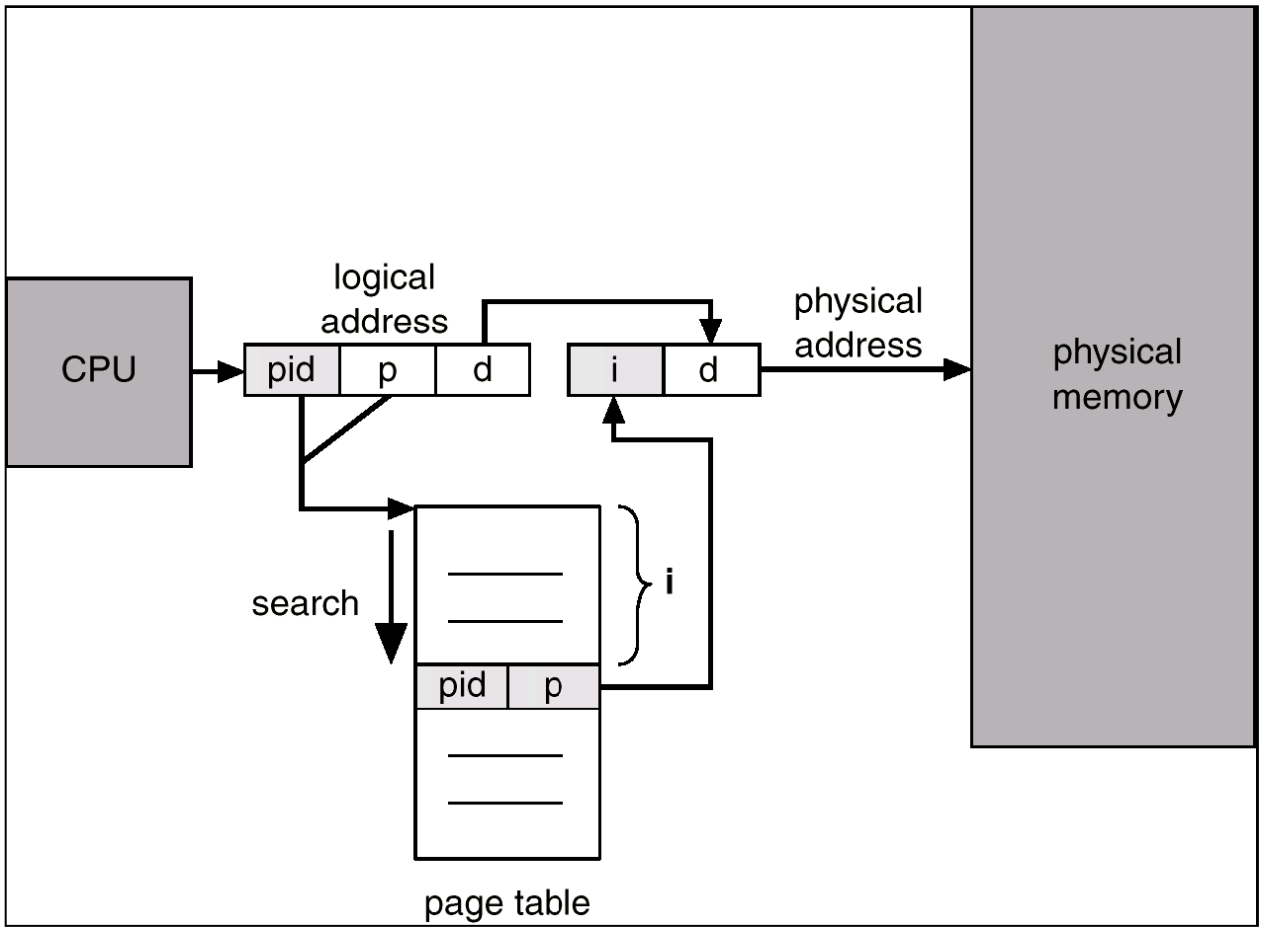
Phân trang (16)
Bảng trang được băm



Phân trang (17)
Bảng trang đảo (Inverted Page Table)

- Mỗi mục từ của bảng trang cho một khung trong bộ nhớ
- Mục từ chứa địa chỉ ảo của trang được chứa trong vị trí bộ nhớ thực tương ứng, với thông tin về quá trình sở hữu trang đó.
- Giảm bộ nhớ cần thiết cho lưu trữ mỗi bảng trang, nhưng tăng thời gian cần thiết để tìm kiếm bảng khi một tham khảo trang xuất hiện.
- Dùng bảng băm để giới hạn tìm kiếm cho một – hoặc nhiều nhất là một ít – các mục từ của bảng trang.

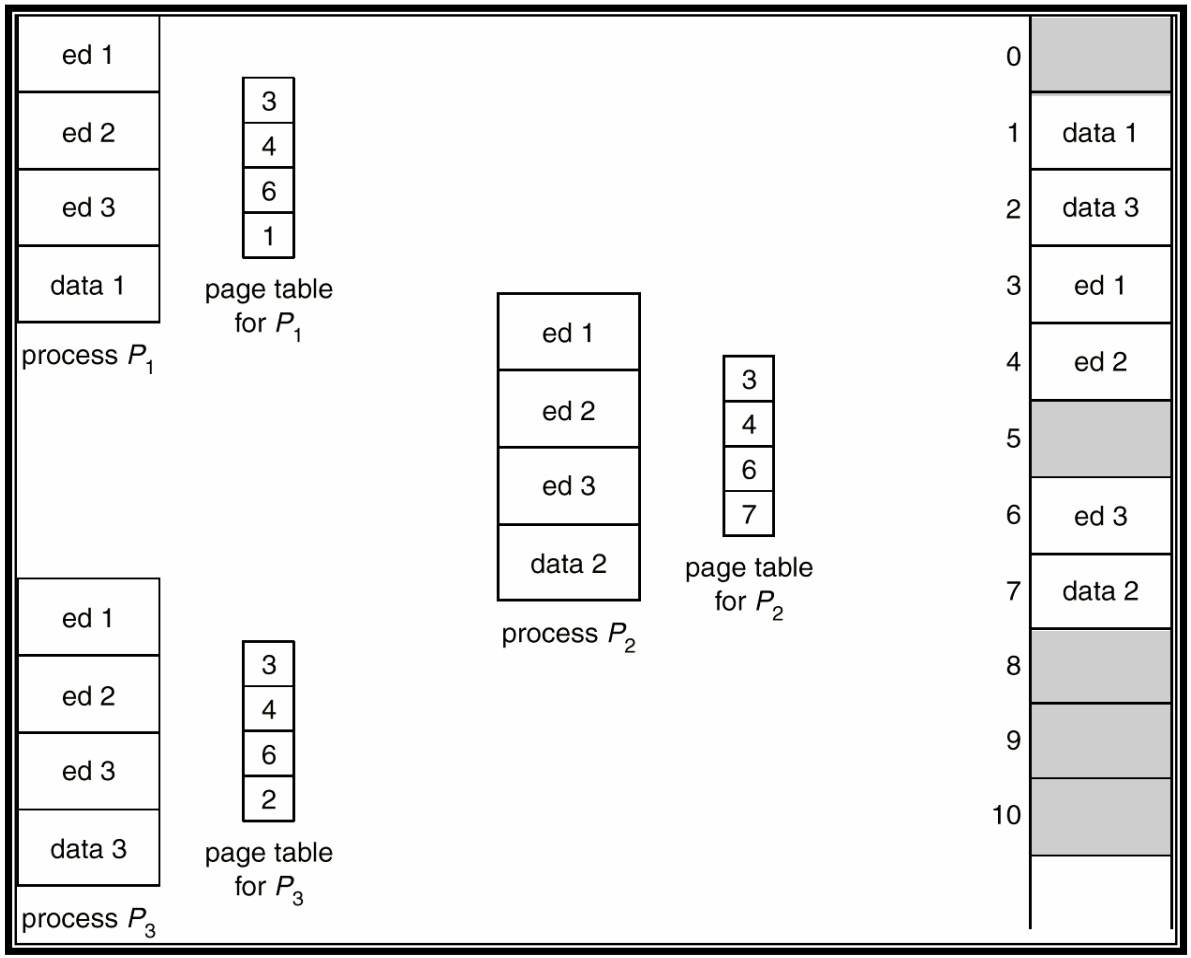
Phân trang (18)
Kiến trúc của bảng trang đảo



Phân trang (19)
Trang được chia sẻ

- Mã lệnh được chia sẻ
  - Một bản sao của mã lệnh chỉ đọc được chia sẻ cho nhiều quá trình (ví dụ text editors, compilers, window systems).
  - Các mã lệnh được chia sẻ phải có cùng vị trí trong không gian địa chỉ ảo của tất cả quá trình.
- Mã lệnh và dữ liệu riêng
  - Mỗi quá trình giữ một bản sao riêng mã lệnh và dữ liệu của mình.
  - Các trang dùng để chứa mã lệnh hoặc dữ liệu riêng có thể nằm ở bất kỳ đâu trong không gian địa chỉ ảo.

Phân trang (20)
Ví dụ về các trang được chia sẻ



Phân đoạn (1)
(Segmentation)

- Sơ đồ quản lý bộ nhớ hỗ trợ việc phân chia bộ nhớ theo góc độ người dùng.
- Một chương trình bao gồm một tập hợp các đoạn (segment). Một đoạn là một đơn vị luận lý, ví dụ như:

  main program,

  procedure,

  function,

  method,

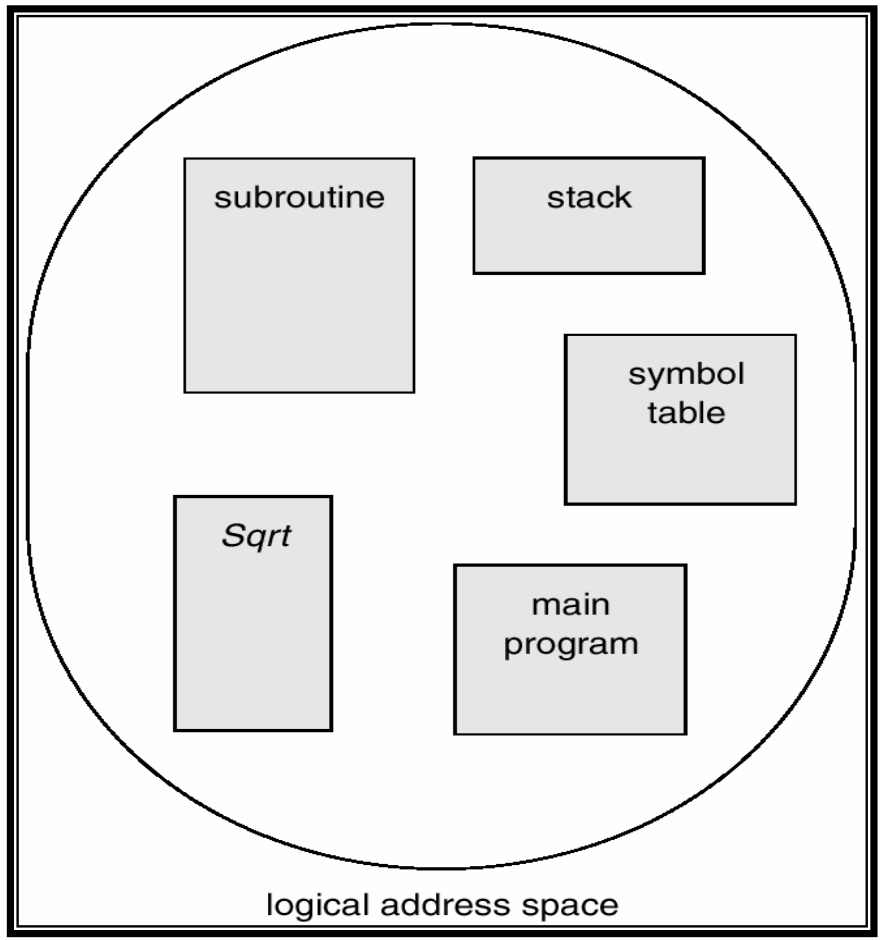
  object,

  local variables, global variables,

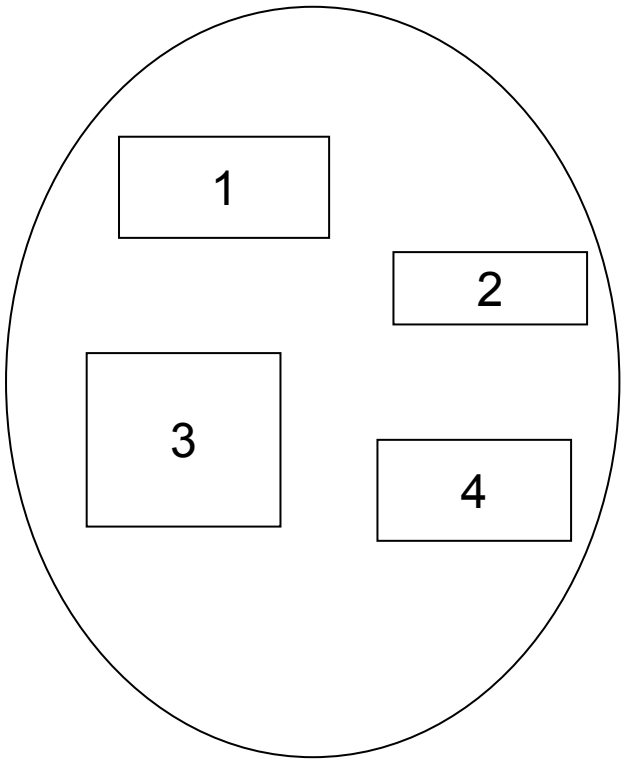
  common block,

  stack,

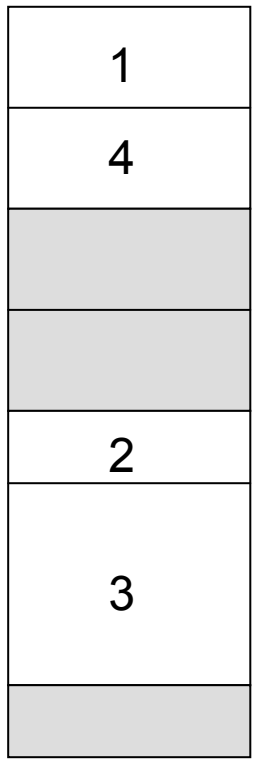
  symbol table, arrays



Phân đoạn (2)
Cái nhìn luận lý về phân đoạn



Không gian người dùng



Không gian bộ nhớ vật lý

Phân đoạn (3)
Kiến trúc hệ thống phân đoạn

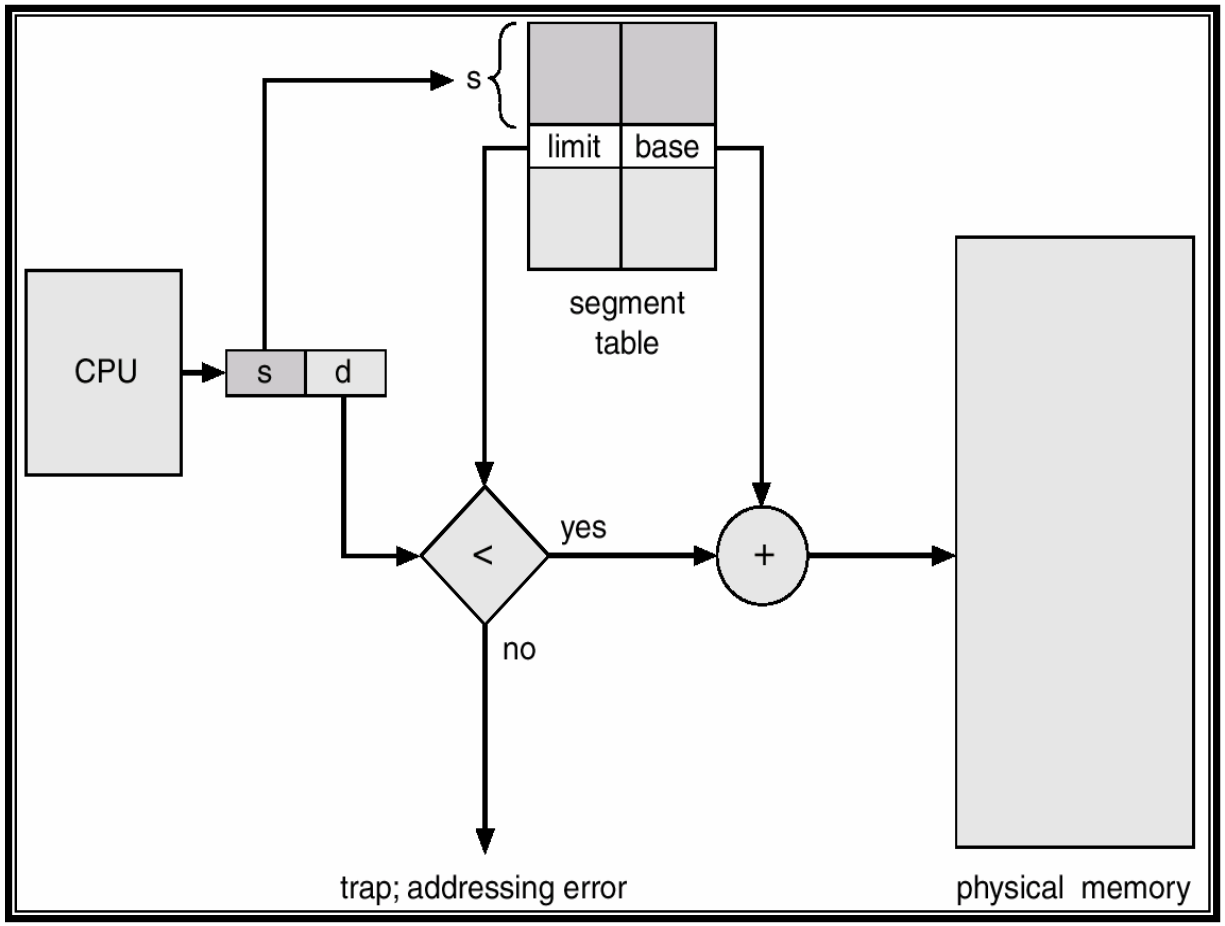
- Địa chỉ luận lý bao gồm một bộ đôi:

  <segment-number, offset>,

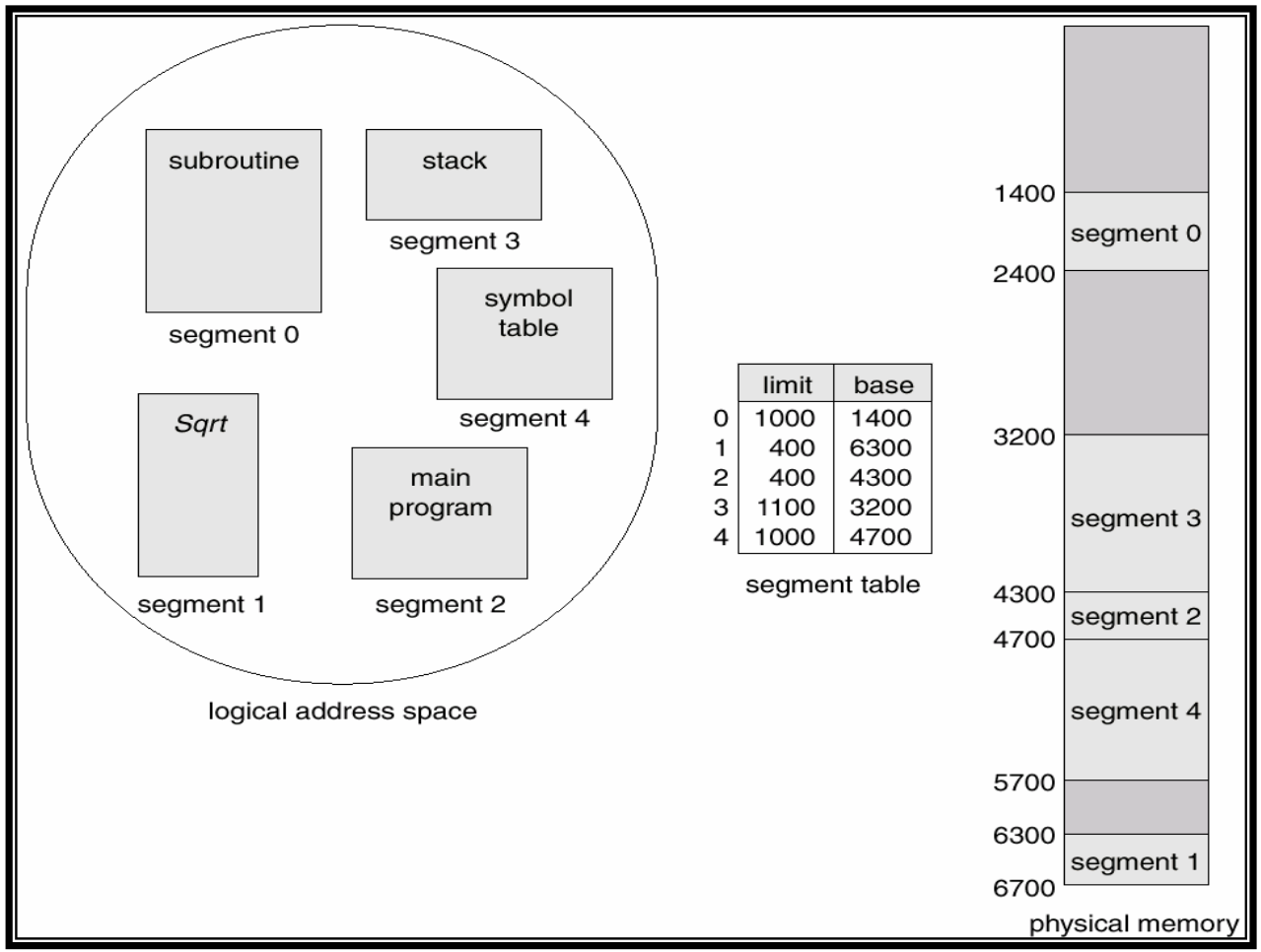
- **Bảng đoạn (Segment table)**: ánh xạ các địa chỉ luận lý 2 chiều định nghĩa bởi người dùng vào các địa chỉ vật lý một chiều.
- Mỗi mục từ trong bảng gồm 2 phần:
  - base – chứa địa chỉ vật lý khởi đầu của đoạn trong bộ nhớ vật lý.
  - limit – chỉ định chiều dài của đoạn.
- **Segment-table base register (STBR)**: là thanh ghi chỉ đến vị trí của bảng 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;

  một đoạn có số hiệu **s** là hợp lệ nếu **s < STLR.**

Phân đoạn (4)
Phần cứng trong hệ thống phân đoạn



Phân đoạn (5)
Ví dụ về phân đoạn



Phân đoạn (6)
Bảo vệ (Protection)

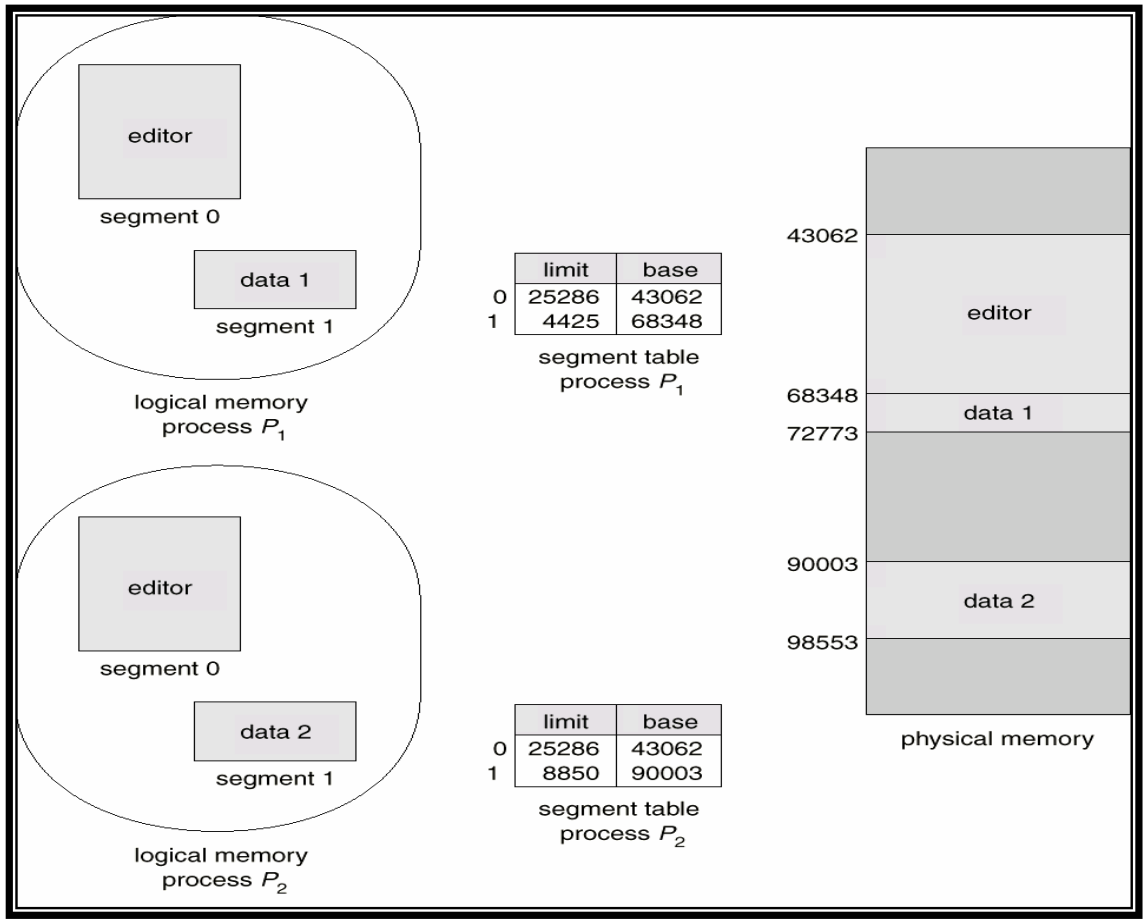
- Cơ chế bảo vệ: Kết hợp với mỗi mục từ trong bảng quản lý đoạn:
  - Bit hợp lệ (validation bit) = 0 $\Rightarrow$ đoạn không hợp lệ
  - Các bit kiểm soát các quyền read/write/execute
- Các bit bảo vệ được kết hợp với các đoạn; việc chia sẻ mã lệnh được thực hiện ở mức đoạn.
- Do các đoạn có độ dài khác nhau, việc cấp phát bộ nhớ là kiểu bài toán cấp phát động.

Phân đoạn (7)
Các đặc điểm khác của hệ thống phân đoạn

- Tái định vị (Relocation):
  - Động.
  - Bằng bảng đoạn.
- Chia sẻ (Sharing):
  - Các đoạn được chia sẻ sẽ có cùng số thứ tự đoạn đối với tất cả các quá trình.
- Cấp phát (Allocation):
  - first fit/best fit.
- Phân mảnh ngoài (External fragmentation)
  - Bởi vì các đoạn có chiều dài biến đổi, việc cấp phát bộ nhớ là vấn đề cấp phát lưu trữ động (dynamic storage-allocation problem).
  - Phân mảnh ngoài xuất hiện khi các khối bộ nhớ trống quá nhỏ để tạo một đoạn.
  - Tránh phân mảnh: cô đặc bộ nhớ để tạo lỗ hổng lớn hơn; tạo các đoạn nhỏ hơn.

Phân đoạn (8)
Chia sẻ các đoạn

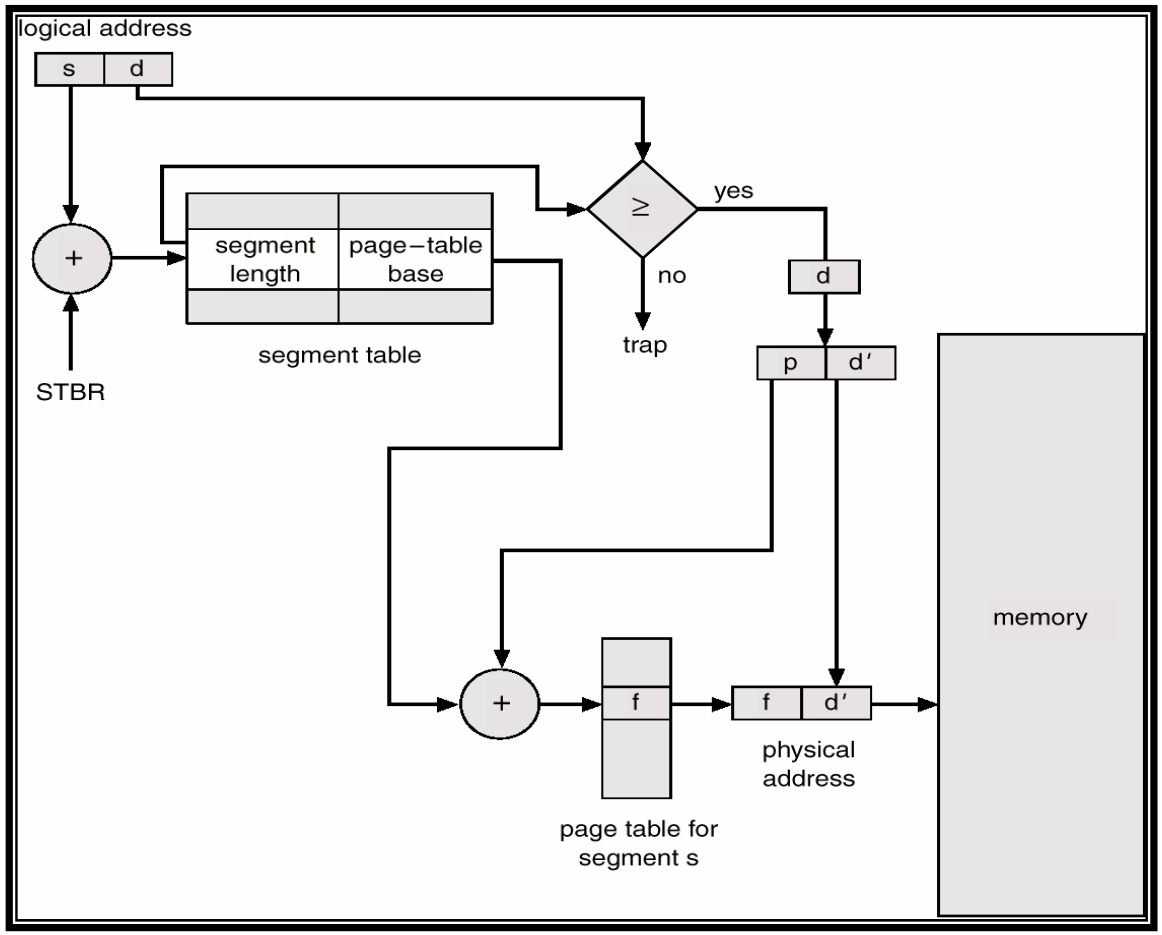
- Dùng text editor trong hệ thống chia thời gian.
- Editor trong đoạn 0 được chia sẻ bởi 2 quá trình.



Phân trang trong phân đoạn (1)
MULTICS

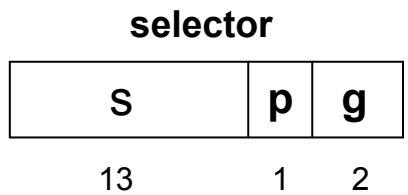
- Hệ thống MULTICS giải quyết các vấn đề phân mảnh ngoài và thời gian tìm kiếm lâu của hệ thống phân đoạn bằng giải pháp phân trang các phân đoạn.
- Giải pháp này khác với giải pháp phân trang thuần túy ở điểm: một mục từ trong bảng quản lý đoạn không phải chứa địa chỉ nền của đoạn mà lại chứa địa chỉ nền của bảng trang của đoạn đó.

Phân trang trong phân đoạn (2)
Sơ đồ dịch địa chỉ trong MULTICS



Phân trang trong phân đoạn (3)
Intel 386

- Intel 386 sử dụng cơ chế quản lý bộ nhớ phân trang trong phân đoạn với sơ đồ phân trang hai mức.
- Selector:
  - s chỉ số hiệu đoạn,
  - g chỉ định đoạn nằm trong bảng mô tả LDT (Local Descriptor Table) hay GDT (Global Descriptor Table)
  - p là các bit bảo vệ



- Segment descriptor chứa giá trị của base và limit.
- Giá trị của base cộng với giá trị offset (32 bit) để tạo ra địa chỉ tuyến tính (linear address) trong đó được tổ chức theo bảng trang 2 mức: directory (10 bit) là độ dời trong bảng trang mức 1, page (10 bit) là độ dời trong bảng trang mức 2, và offset (12 bit) là độ dời trong khung trang.

Phân trang trong phân đoạn (4)
Dịch địa chỉ trong Intel 80386