Chương 4
Quản lý Bộ nhớ
Memory Management

4.1 Quản lý Bộ nhớ cơ bản
4.2 Swapping
4.3 Bộ nhớ ảo - Virtual memory
4.4 Cách thực hiện
4.5 Phân đoạn - Segmentation

Quản lý Bộ nhớ

Quản lý Bộ nhớ

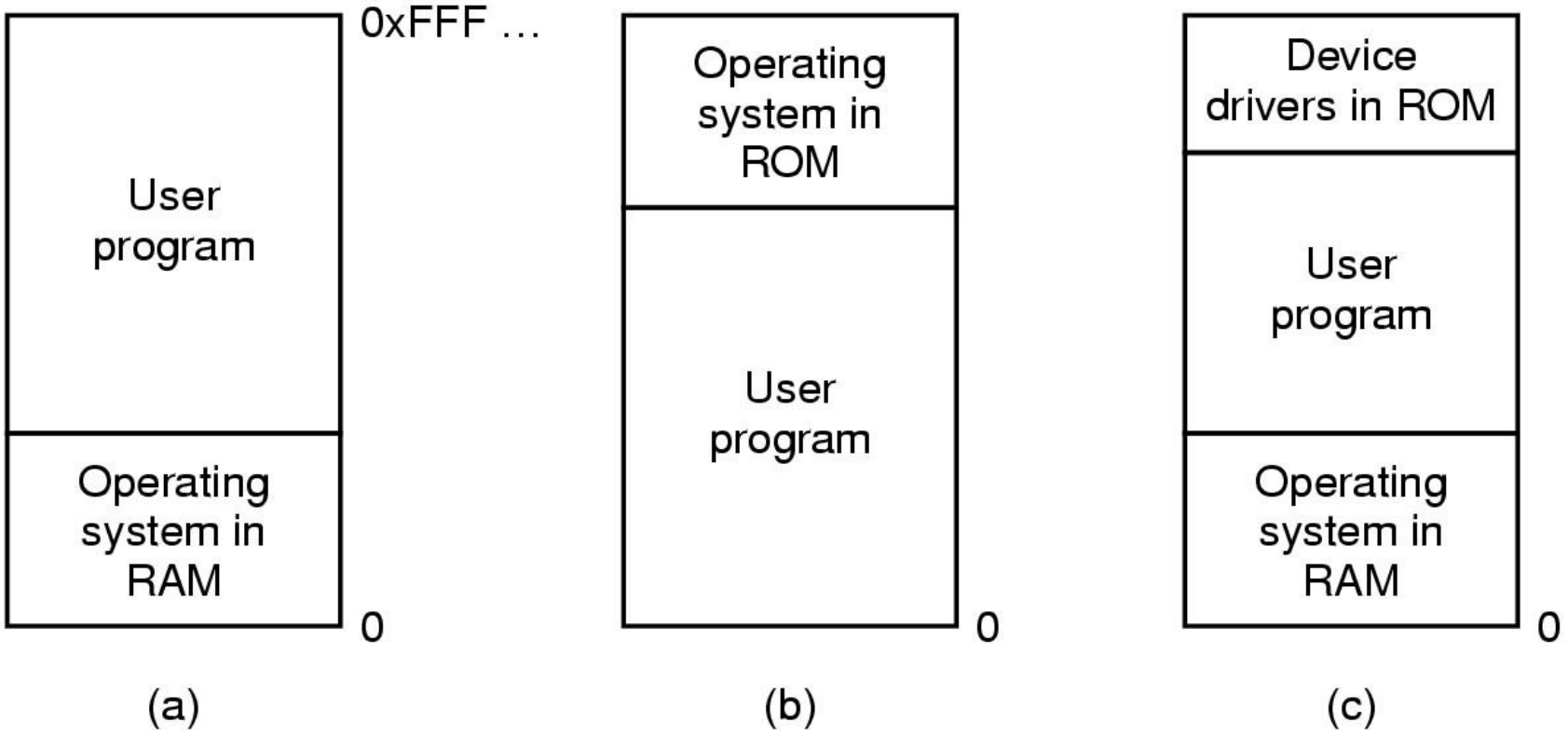
- Mong muốn của người lập trình có bộ nhớ lý tưởng là
  - dung lượng lớn
  - nhanh
  - ổn định
- Phân cấp bộ nhớ
  - small amount of fast, expensive memory – cache
  - some medium-speed, medium price main memory
  - gigabytes of slow, cheap disk storage
- Memory manager handles the memory hierarchy

Quản lý Bộ nhớ cơ bản
Không gian địa chỉ Logical với Không gian địa chỉ vật lý

- Khái niệm về không gian địa chỉ logic bị ràng buộc vào không gian địa chỉ vật lý là vấn đề chính để quản lý bộ nhớ:
  - **Logical address** – được sinh ra bởi CPU; cũng được xem như địa chỉ ảo - **virtual address**
  - **Physical address** – địa chỉ hiển thị bởi bộ nhớ vật lý

Quản lý Bộ nhớ cơ bản

Đơn nhiệm với Swapping hay Paging

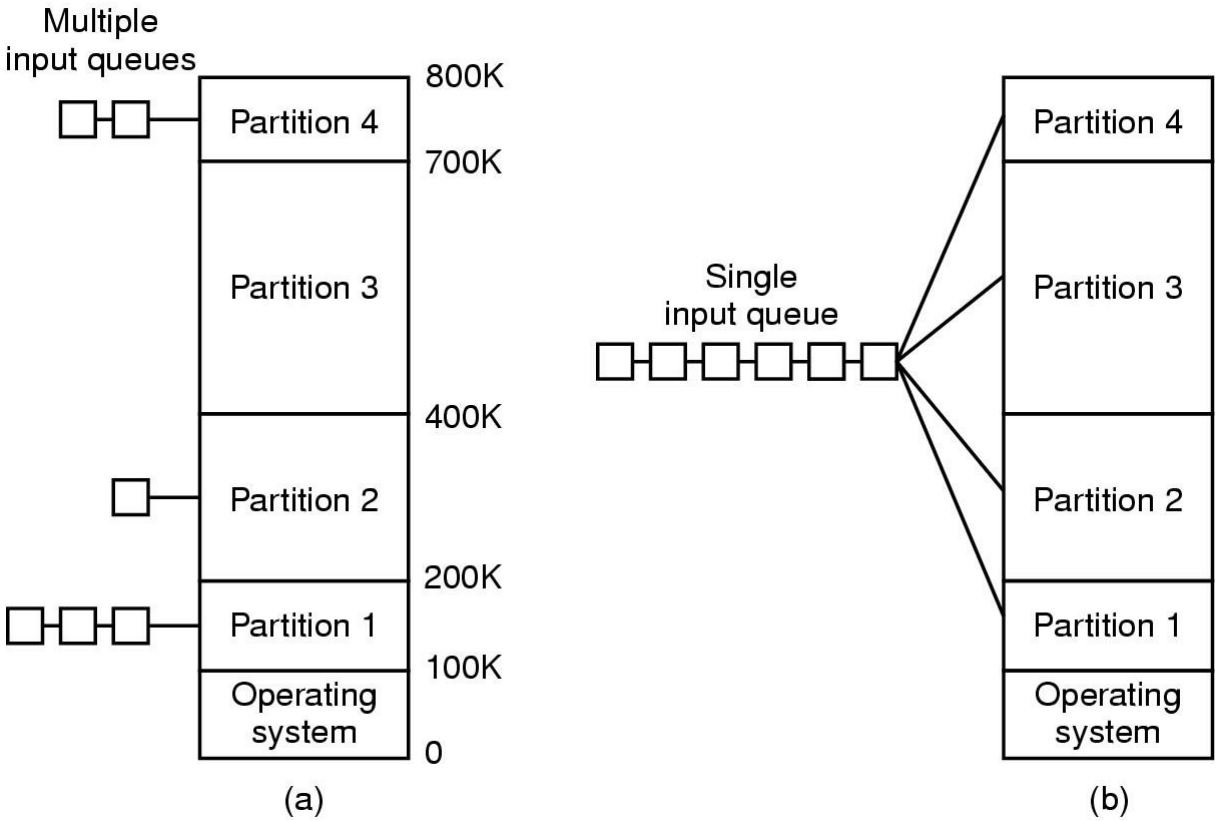


Có ba cách đơn giản để tổ chức bộ nhớ

- một hệ điều hành với một tiến trình user

Quản lý Bộ nhớ cơ bản

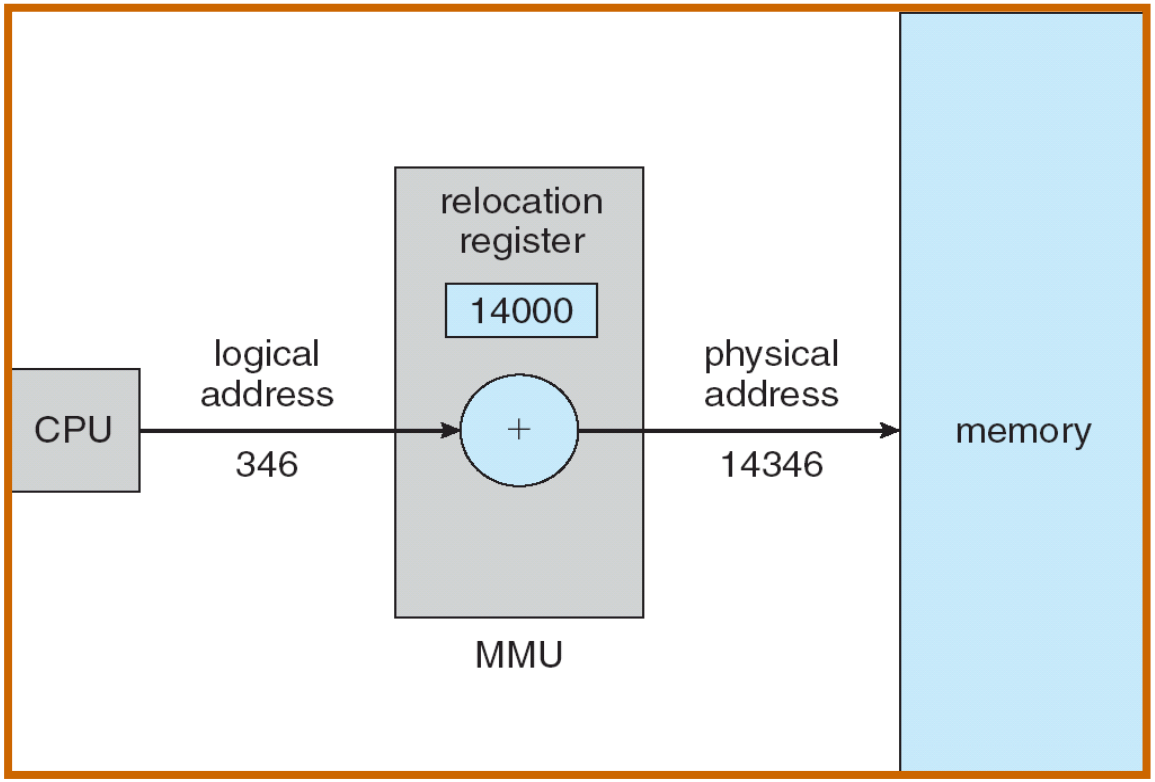
Đa nhiệm với các phân vùng cố định (Fixed Partitions)



- Các phân vùng bộ nhớ cố định
  - (a) nhiều hàng đợi vào
  - (b) đơn hàng đợi vào

Quản lý Bộ nhớ cơ bản

Tái cấp phát động sử dụng thanh ghi Relocation



Quản lý Bộ nhớ cơ bản
Tái cấp phát và Bảo vệ

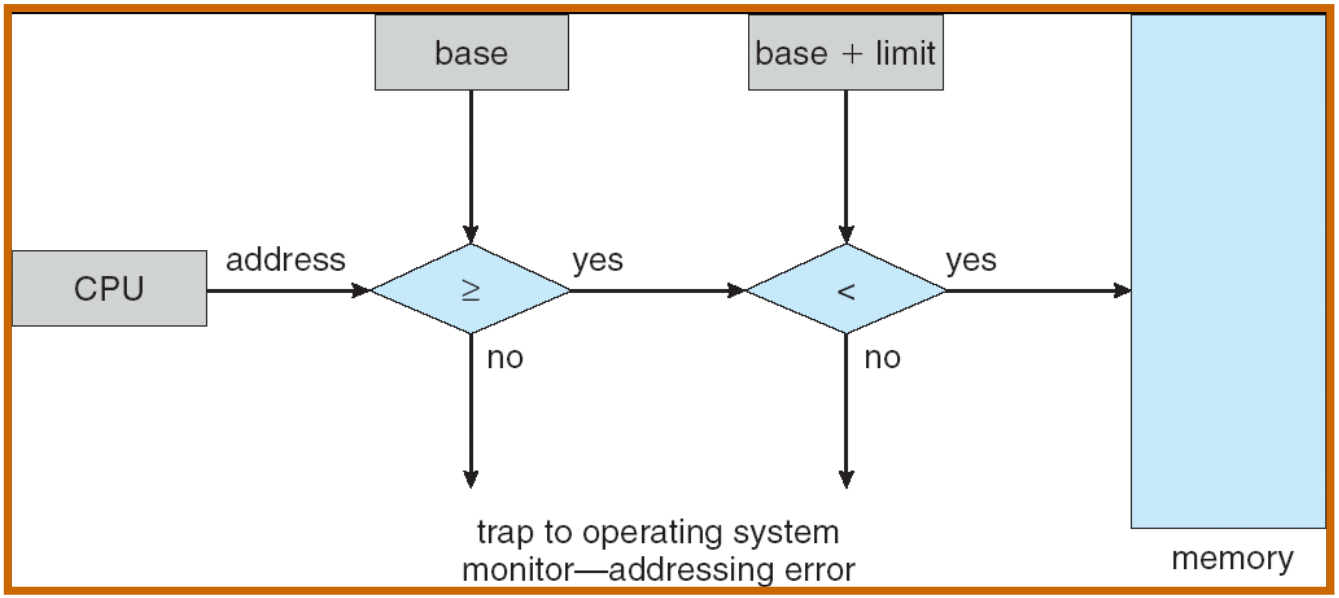
- Không chắc chắn ở đó chương trình được tải vào bộ nhớ
  - địa chỉ cấp phát cho các biến, cho đoạn code có thể thay đổi
  - cần phải giữa cho chương trình nằm ngoài các phân vùng của tiến trình khác
- Sử dụng các giá trị base và limit
  - address locations added to base value to map to physical addr
  - address locations larger than limit value is an error

Quản lý Bộ nhớ cơ bản
Tái cấp phát và Bảo vệ

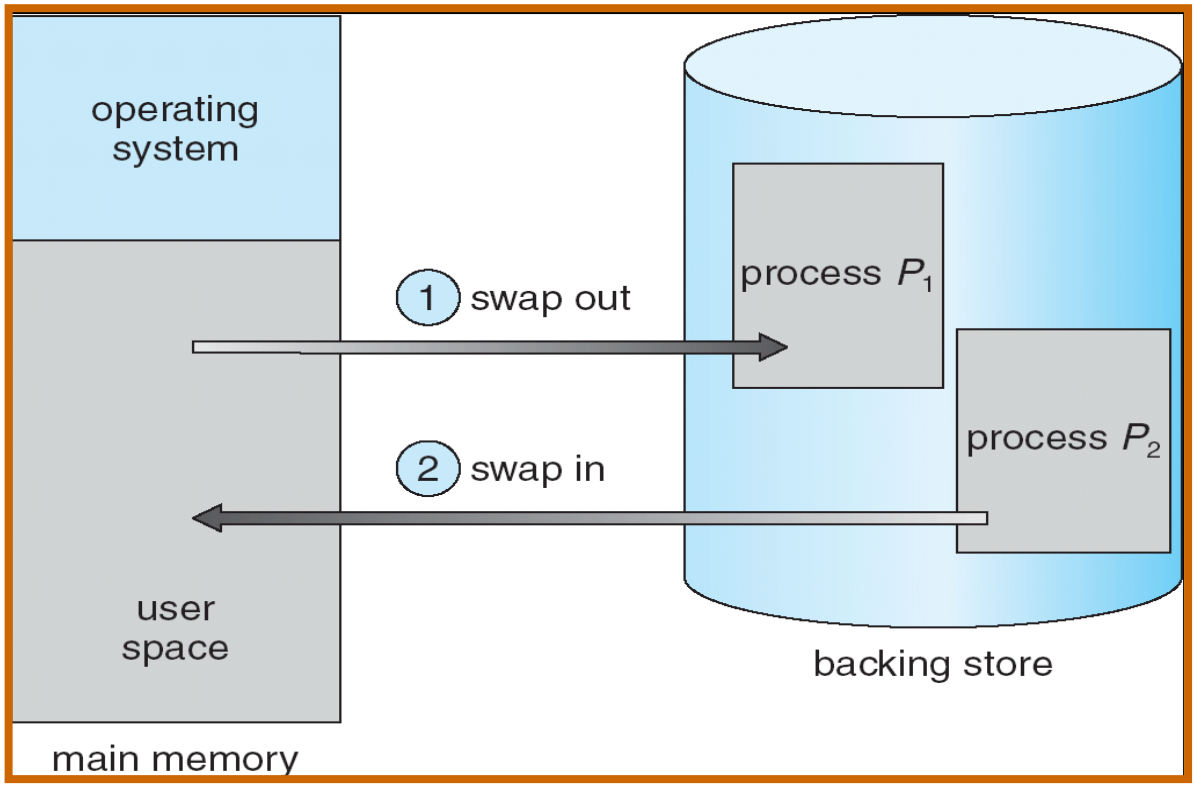
- Relocation registers used to protect user processes from each other, and from changing operating-system code and data
  - Base register contains value of smallest physical address
  - Limit register contains range of logical addresses – each logical address must be less than the limit register
  - MMU maps logical address *dynamically*

Quản lý Bộ nhớ cơ bản

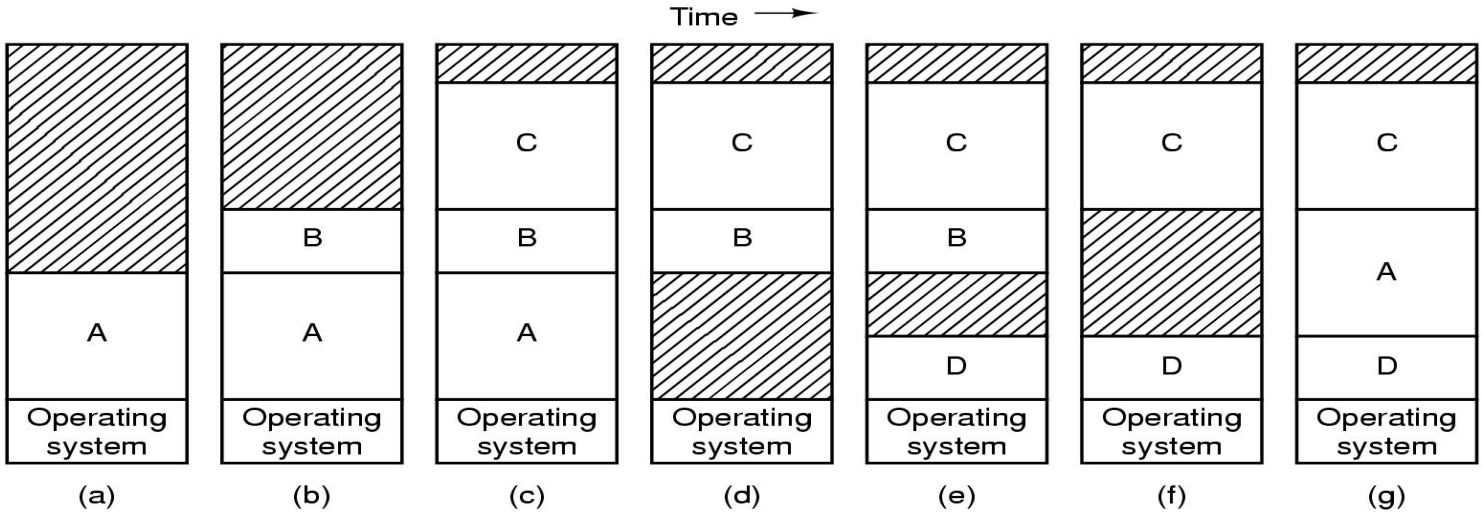
HW address protection with base and limit registers



Swapping (1)
Lược đồ Swapping

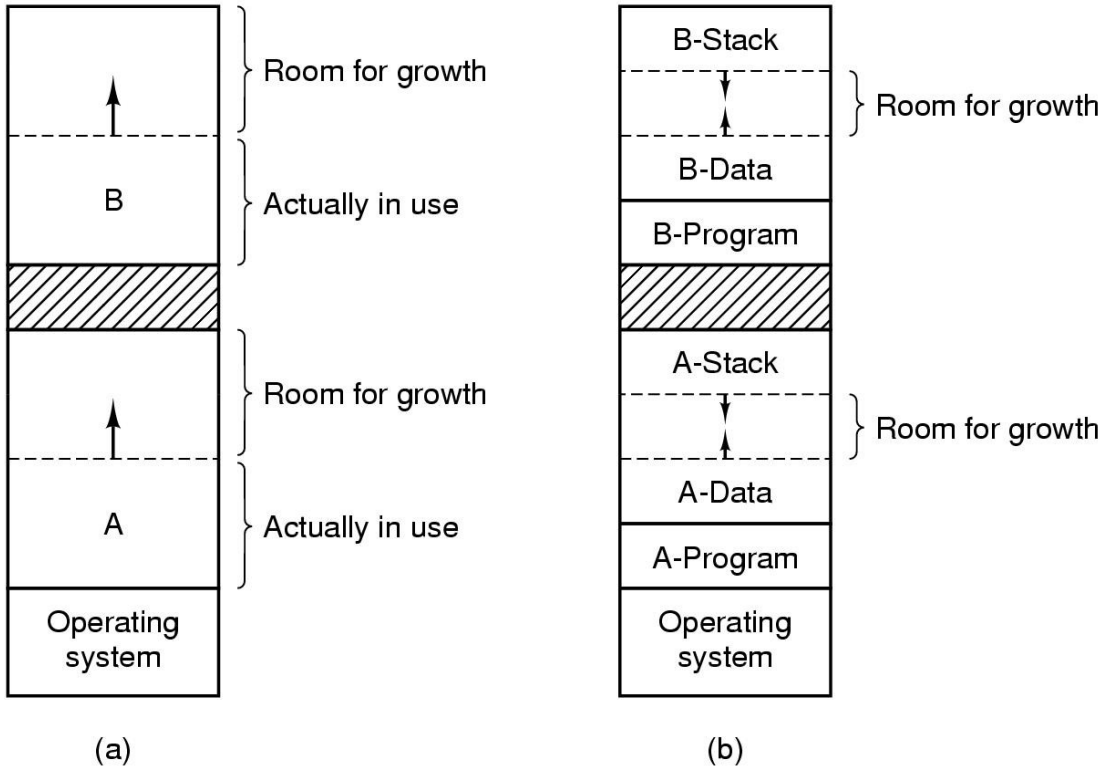


Swapping (2)



- Cấp phát Bộ nhớ có thể thay đổi khi
  - tiến trình đến bộ nhớ
  - tiến trình rời khỏi bộ nhớ
- Các vùng gạch chéo là chưa sử dụng
- **External Fragmentation** – phân mãnh ngoại: tổng không gian trống có thể đủ để cấp thêm cho một tiến trình nhưng lại nằm rãi rác không liên tục
- **Internal Fragmentation** – phân mãnh nội: vùng nhớ cấp phát lớn hơn vùng nhớ yêu cầu; do kích thước của phân vùng và kích thước yêu cầu của tiến trình khác nhau; free nhưng không được dùng

Swapping (3)



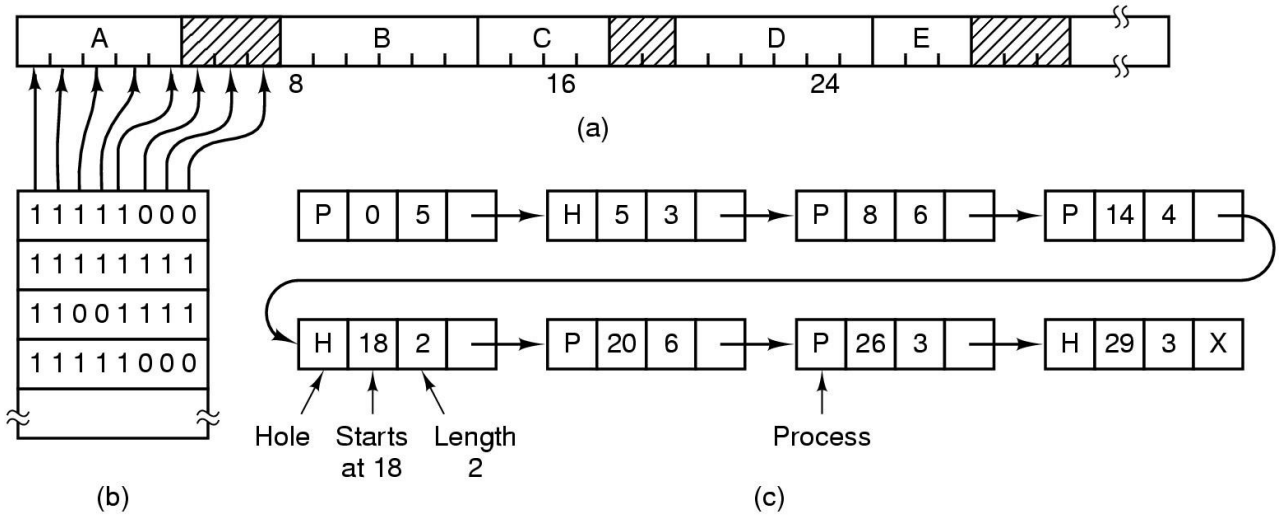
- (a) Cấp phát vùng nhớ để phát triển đoạn data
- (b) Cấp phát vùng nhớ để phát triển đoạn stack & data

Swapping (4)
Đa nhiệm với phân vùng động

- Đa nhiệm với phân vùng động
  - Hole – block trống của bộ nhớ; các vùng hole có kích thước khác nhau và nằm rải rác khắp nơi trên bộ nhớ
  - Khi một tiến trình đến, nó được cấp phát bộ nhớ từ các vùng hole đủ lớn phù hợp
  - Hệ điều hành quản lý:
    a) các phân vùng đã cấp (allocated partitions)
    b) các phân vùng free (free partitions - hole)
  - Có hai cách để quản lý các phân vùng đã cấp:
    - Quản lý Bộ nhớ với Bit Maps
    - Quản lý Bộ nhớ với Linked Lists

Swapping (4)

Đa nhiệm với phân vùng động



Quản lý Bộ nhớ với Bit Maps

- (a) Bộ nhớ với 5 tiến trình, 3 holes
  - tick marks show allocation units
  - shaded regions are free
- (b) Nội dung của bit map
- (c) Nội dung của danh sách liên kết

Swapping (5)
Các vấn đề của cấp phát động Bộ nhớ

Làm thế nào để thỏa mãn một yêu cầu cần kích thước n từ danh sách các vùng trống

- **First-fit**: Cấp phát vùng free đầu tiên đủ lớn
- **Next fit**: Bắt đầu tìm từ vị trí con trỏ hiện hành phân vùng đầu tiên đủ lớn
- **Best-fit**: Cấp phát vùng free nhỏ nhất nhưng đủ lớn; cần phải tìm toàn bộ danh sách
- **Worst-fit**: Cấp phát phân vùng free lớn nhất; cũng phải tìm toàn bộ dach sách

Bộ nhớ ảo
Virtual Memory

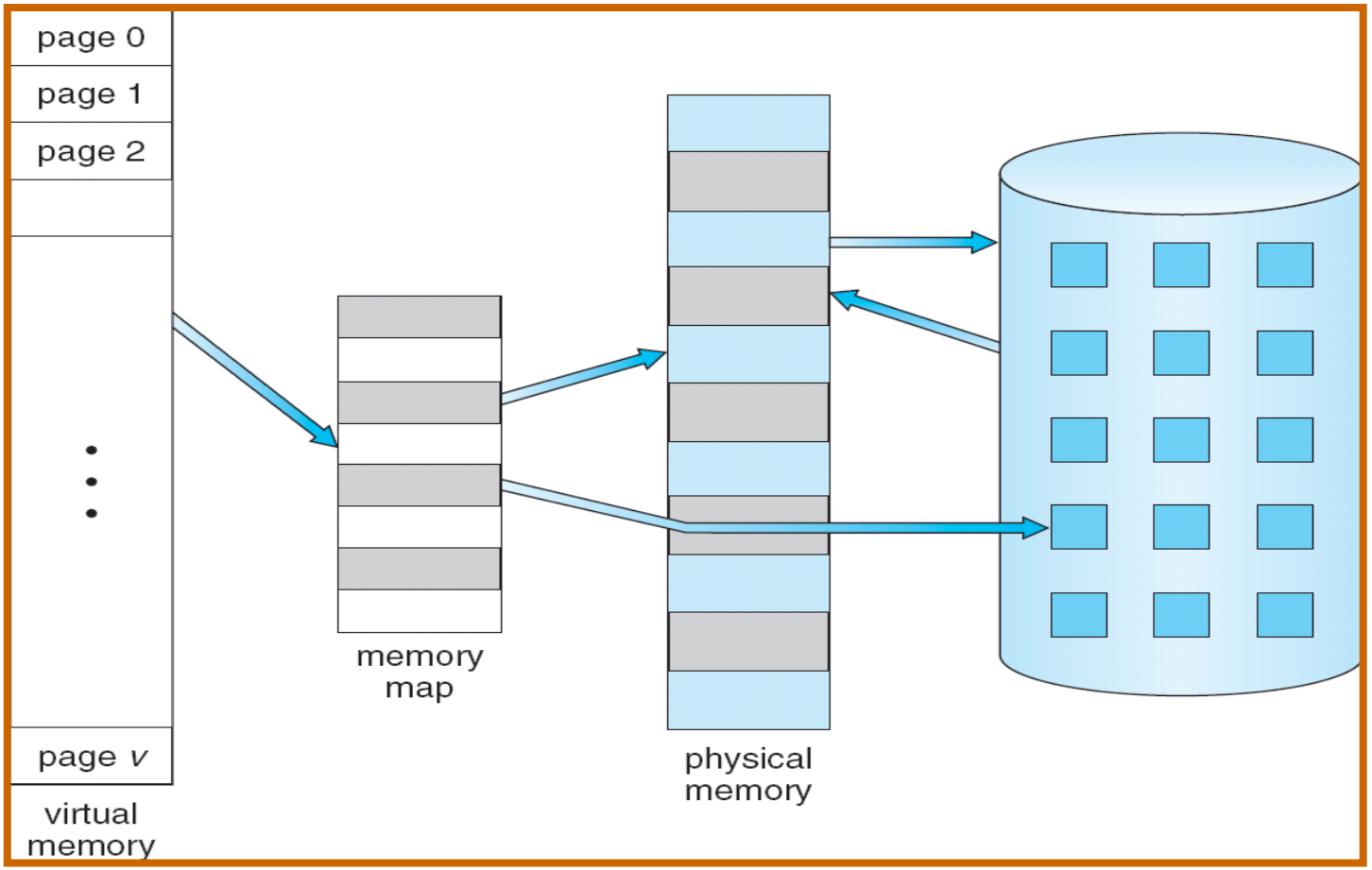
Bộ nhớ ảo

- **Bộ nhớ ảo - Virtual memory** – chia bộ nhớ logic của user thành các phần từ bộ nhớ vật lý.
  - Chỉ một phần của chương trình sẽ được nạp vào bộ nhớ khi thực thi
  - Không gia địa chỉ logic (Logical address space) có thể lớn hơn nhiều so với không gian bộ nhớ vật lý (physical address space)
  - Cho phép không gian bộ nhớ vật lý có thể chia sẻ cho nhiều tiến trình
  - Cho phép tạo ra nhiều tiến trình hơn
- Bộ nhớ ảo có thể thực hiện thông qua:
  - **Phân trang - paging**
  - **Phân đoạn - segmentation**

Bộ nhớ ảo
Phân trang - Paging

- Không gian địa chỉ ảo của tiến trình có thể không liên tục trong bộ nhớ; tiến trình có thể cấp phát bộ nhớ vật lý bất kể ở đâu nếu có vùng free.
- Chia bộ nhớ logic thành các block gọi là **trang - pages**
- Chia bộ nhớ vật lý thành các block có kích thước xác định gọi là **khung trang - Page frames**
- **1 trang = 1 khung trang** (từ 512 bytes đến 8,192 bytes)
- Keep track of all free frames
- Để chạy một chương trình có kích thước ***n*** trang, cần phải tìm *n* khung free và tải chương trình vào
- Thiết lập **bảng trang - page table** để chuyển đổi địa chỉ logic thành địa chỉ vật lý
- Phân mảnh nội

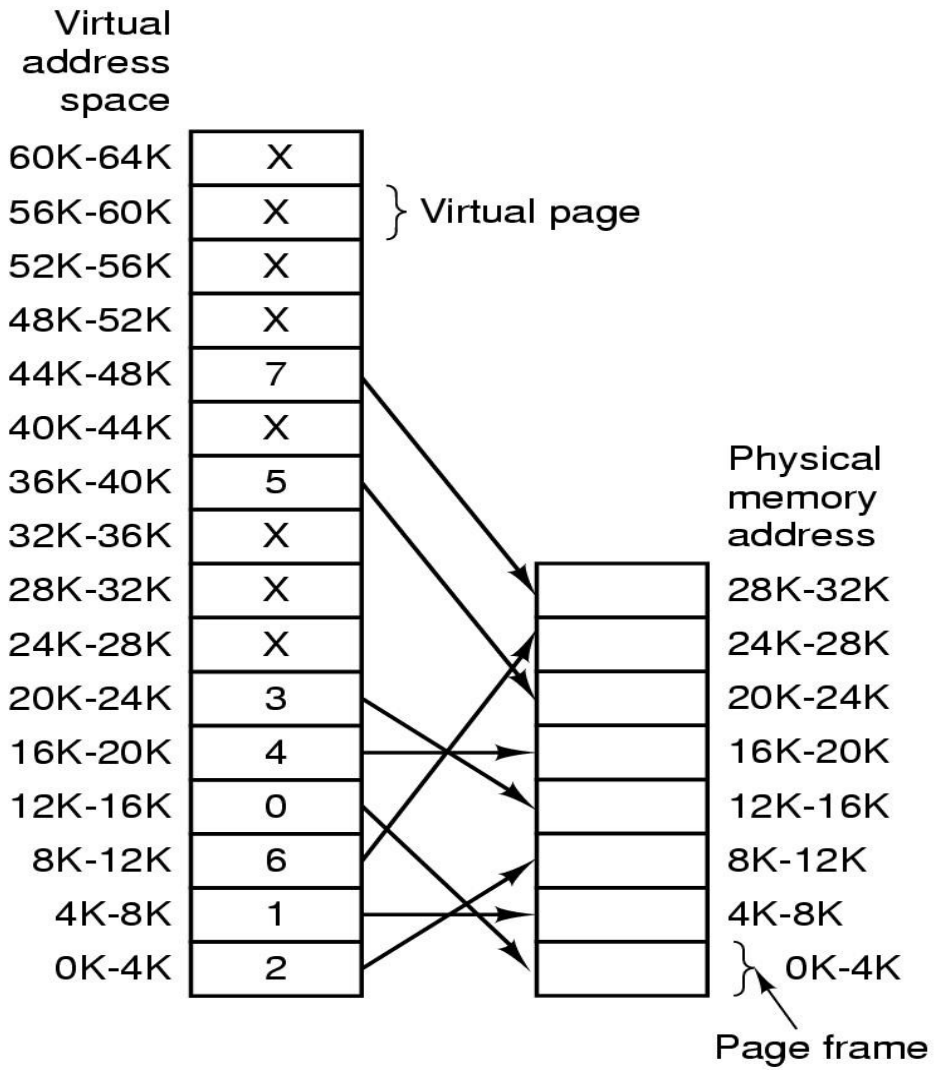
Bộ nhớ ảo
Phân trang - Paging



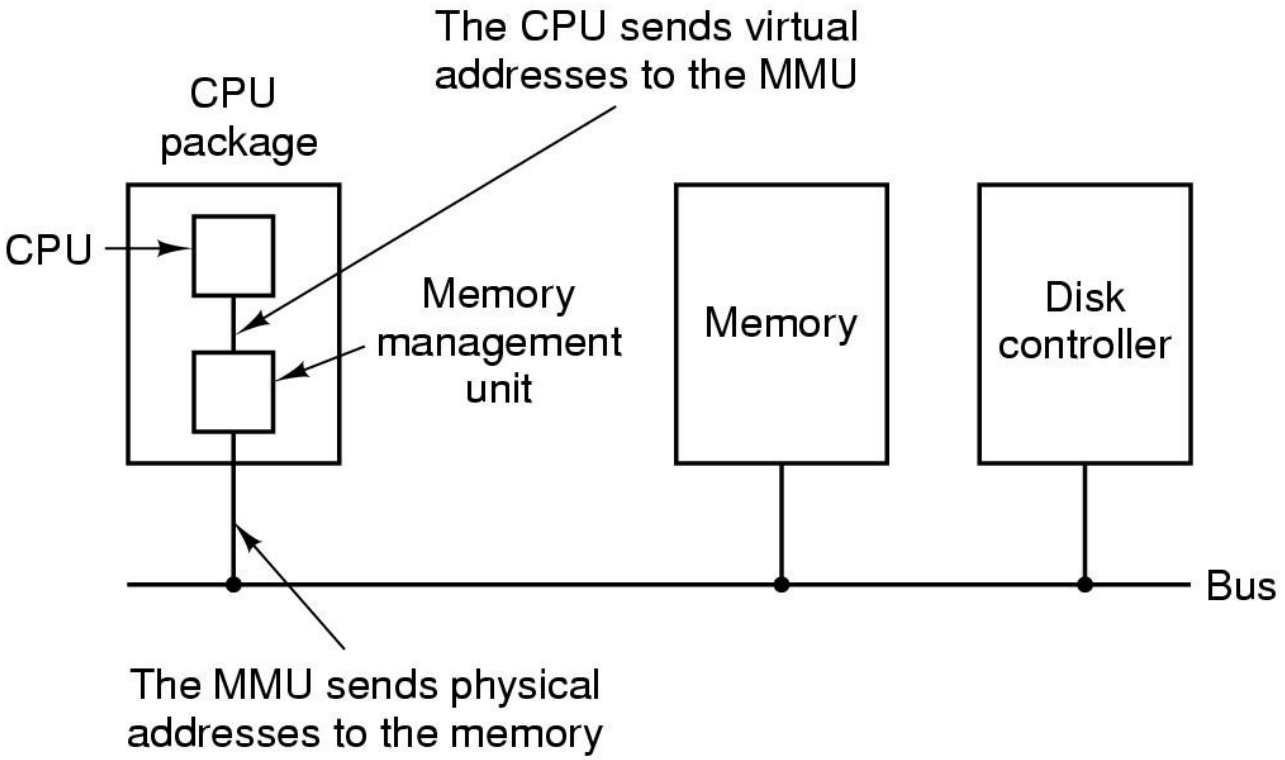
Bộ nhớ ảo

Phân trang: Ví dụ

Liên quan giữa
địa chỉ ảo và
địa chỉ vật lý



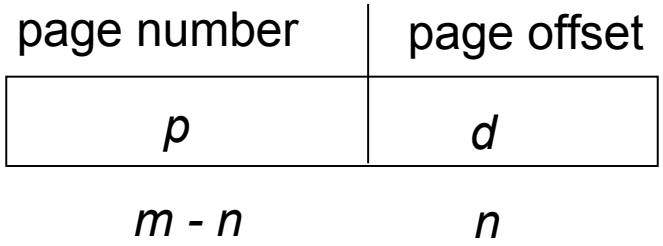
Bộ nhớ ảo
Phân trang - Paging



Vị trí và chức năng của MMU

Bộ nhớ ảo
Lược đồ chuyển đổi địa chỉ

- Địa chỉ sinh ra bởi CPU được chia thành:
  - **Số trang ảo - Page number (*p*)** – được dùng như số thứ tự trong *bảng trang- page table* - chứa địa chỉ cơ sở cho mỗi trang trong bộ nhớ vật lý

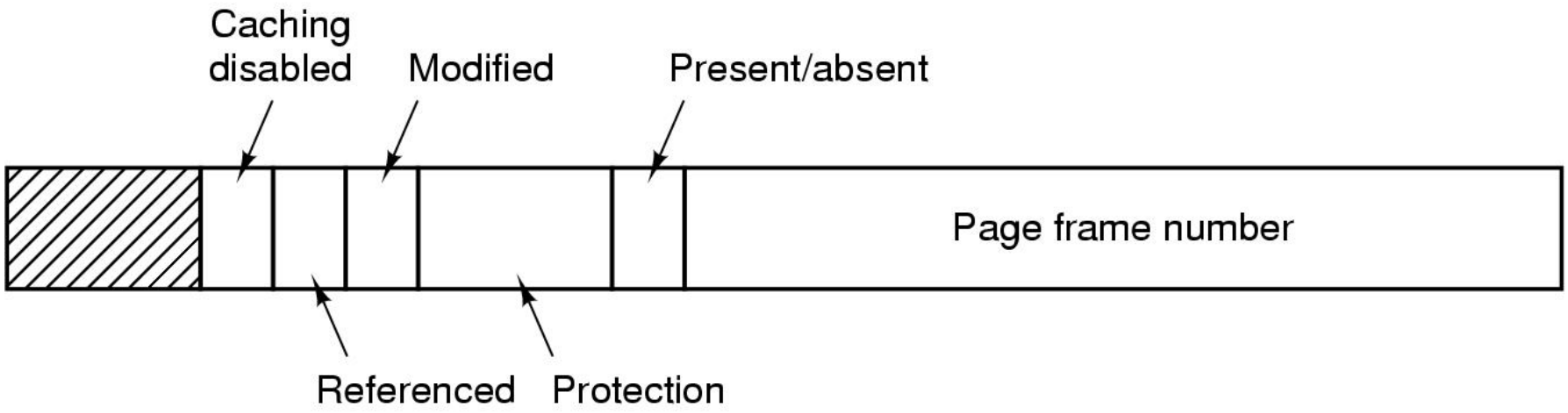


  - **Page offset (d)** – địa chỉ tương đối tính từ đầu trang, ghép với địa chỉ cơ sở để thành địa chỉ vật lý và địa chỉ này được gửi bởi MMU

Địa chỉ logic 2^m và địa chỉ offset 2^n

Bộ nhớ ảo

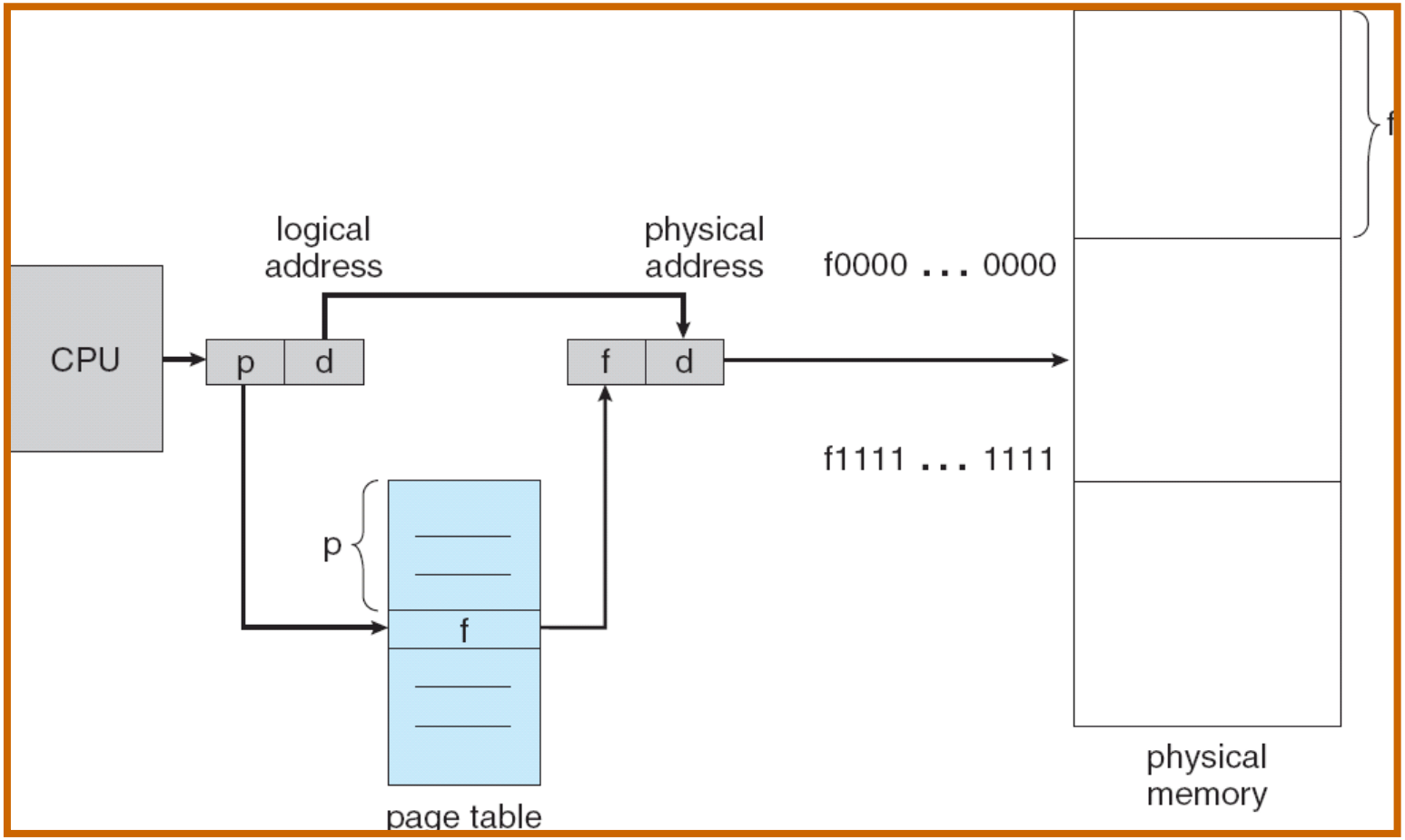
Một entry của Bảng trang



Một hàng của bảng trang

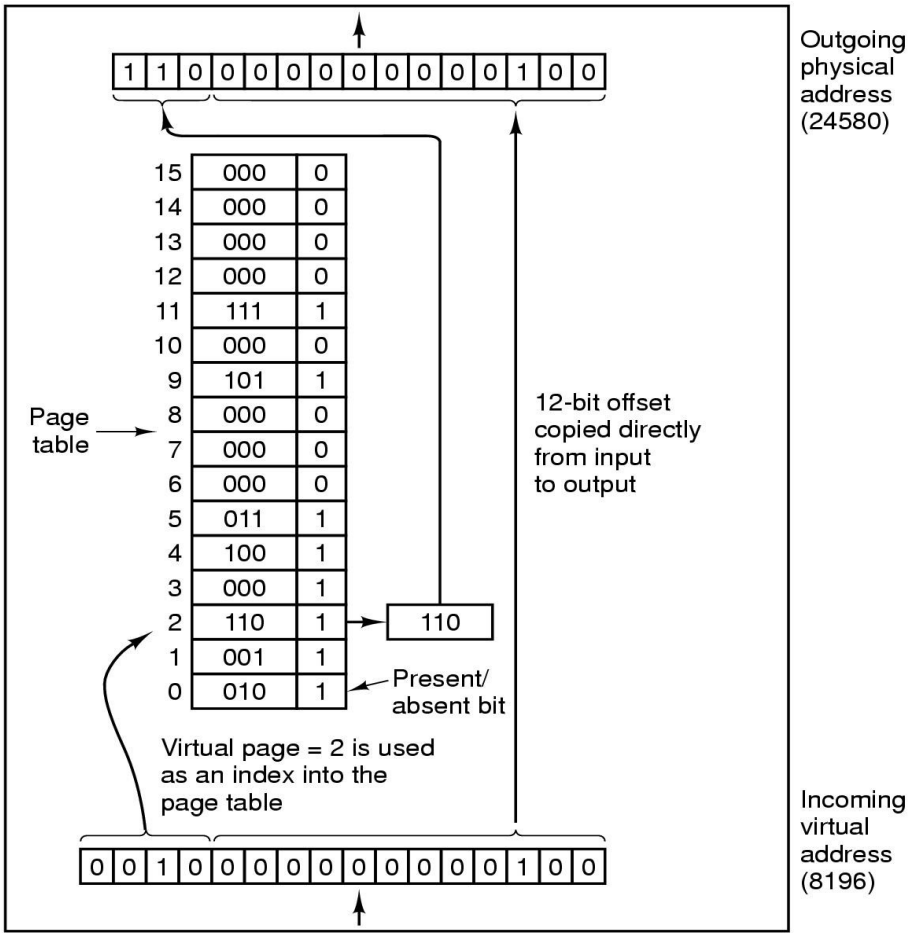
Bộ nhớ ảo

Paging Hardware



Bộ nhớ ảo

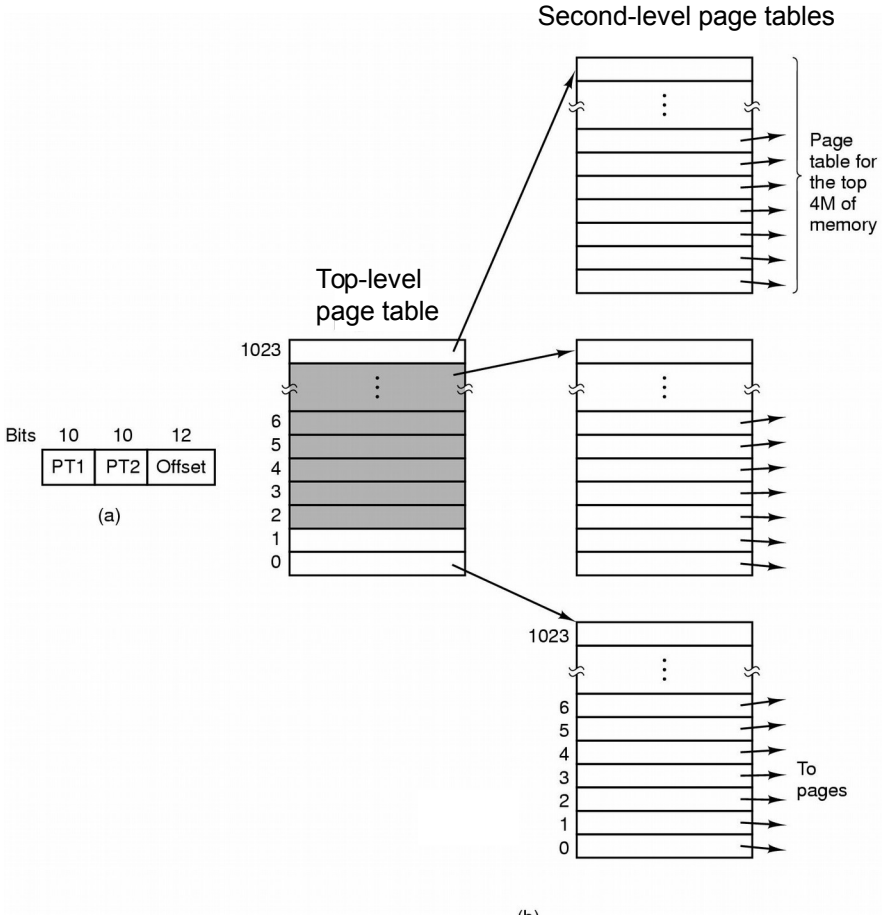
Bảng trang - Page Tables: Ví dụ



Thao tác bên trong của với 16 trang, mỗi trang 4 KB

Bộ nhớ ảo

Bảng trang 2 mức

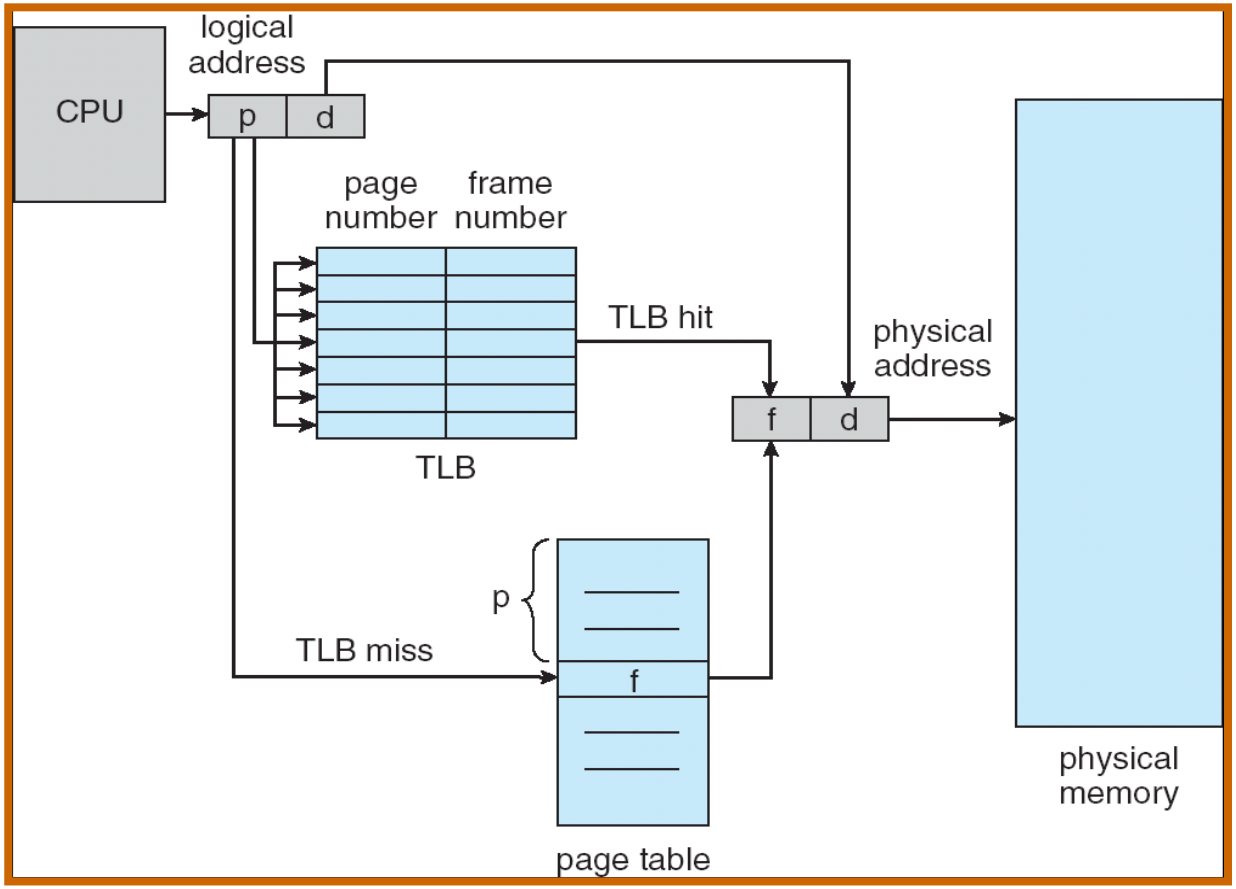


- 32 bit địa chỉ với 2 trường bảng trang
- Bảng trang 2 mức

Bộ nhớ ảo
Thực hiện Bảng trang

- Bảng trang được giữ trong bộ nhớ
- **Page-table base register (PTBR)** trỏ đến bảng trang
- **Page-table length register (PRLR)** chỉ ra kích thước của bảng trang
- Cứ mỗi lần cần truy cập đến data/instruction hệ thống cần 2 lần truy cập đến bộ nhớ. Một lần đến bảng trang và một lần đến data/instruction.
- Vì vậy, làm thế nào để truy cập nhanh, co thể giải quyết bằng cách sử dụng bộ nhớ cache gọi là **bộ nhớ kết hợp-associative memory** hay **translation look-aside buffers (TLBs)**

Bộ nhớ ảo
Phân trang với TLB



Bộ nhớ ảo

TLBs – Translation Lookaside Buffers

Valid	Virtual page	Modified	Protection	Page frame
1	140	1	RW	31
1	20	0	R X	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	R X	50
1	21	0	R X	45
1	860	1	RW	14
1	861	1	RW	75

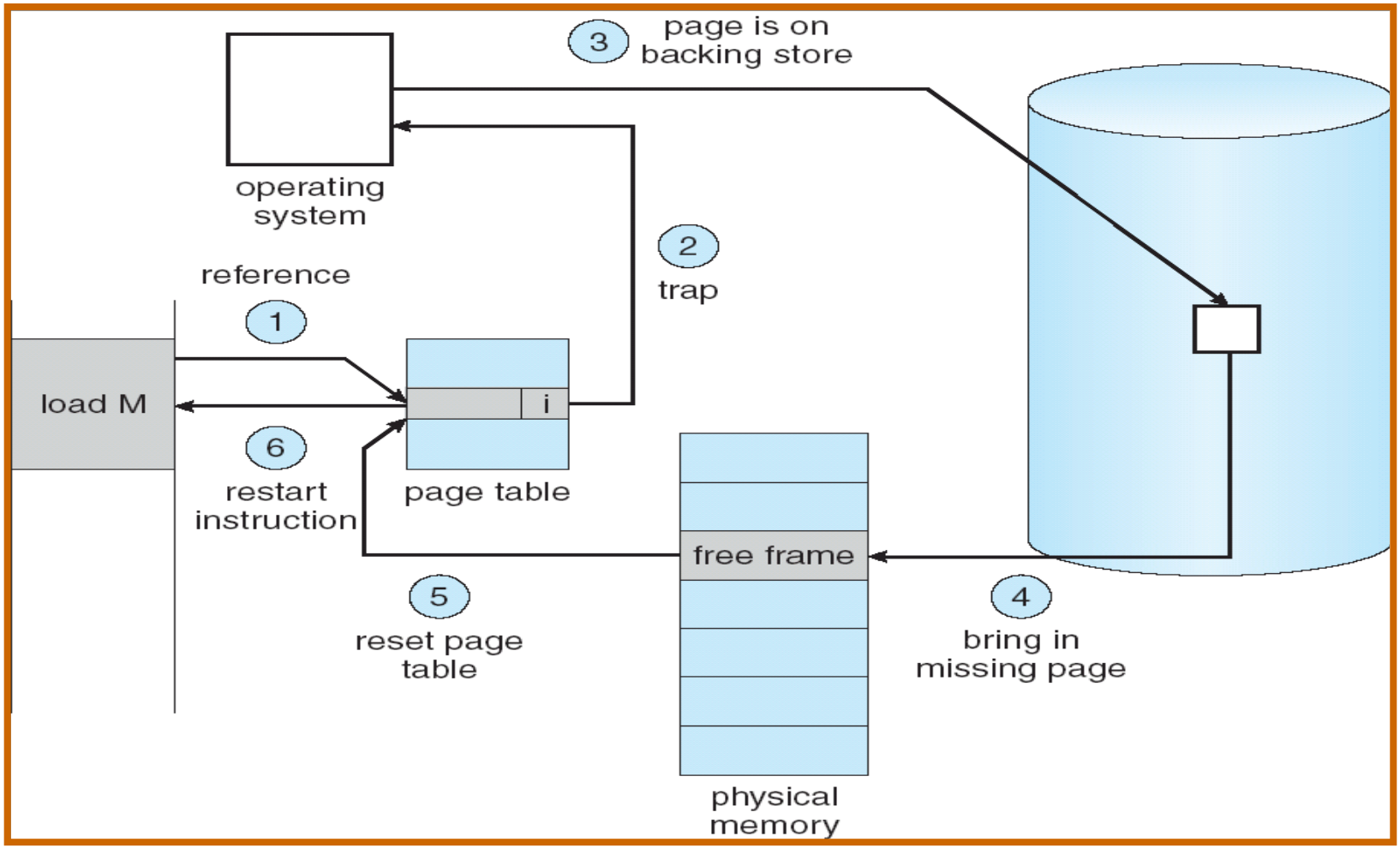
TLB để tăng tốc độ phân trang

Bộ nhớ ảo
Lỗi trang

1. Nếu tham chiếu đến một trang mà trang chưa được nạp vào bộ nhớ gọi là: **lỗi trang**
2. Bẫy lỗi (Trap) đến hệ điều hành:
3. Tạo một khung trang trống, sao lưu trang
4. Swap trang từ đĩa vào khung trang vừa được tự do trong bộ nhớ
5. Chỉnh sửa lại bảng trang, thiết lập bit = **v**
6. Khởi động lại lệnh gây ra lỗi trang

Bộ nhớ ảo

Các bước điều khiển lỗi trang



Bộ nhớ ảo

Các thuật toán thay thế trang

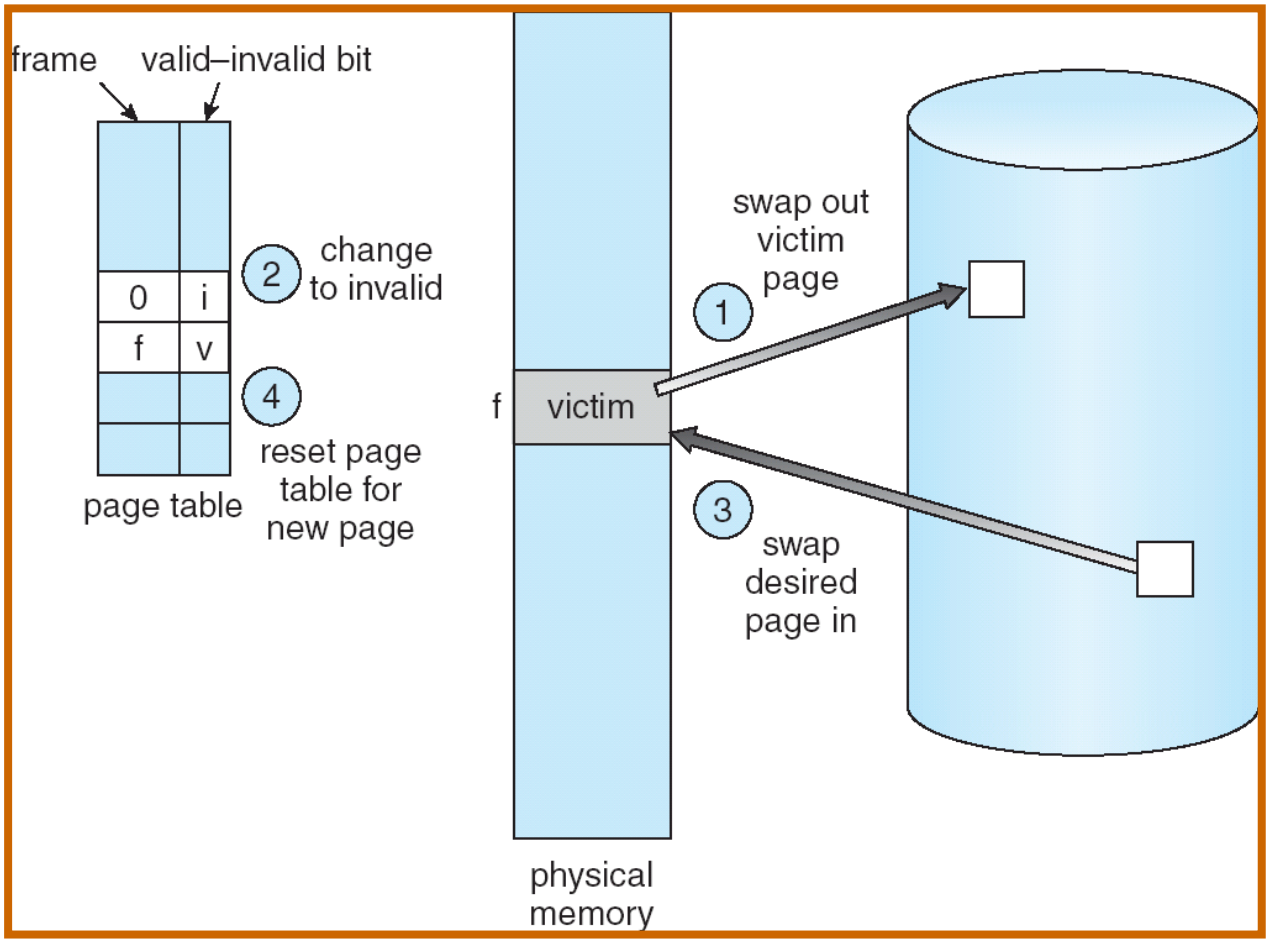
- Điều gì xãy ra nếu không có khung trang free?
- Thay thế trang - tìm vài trang trong bộ nhớ, mà thực sự chưa cần dùng đến, swap trang trở ra đĩa
  - thuật toán
  - hiệu quả – thuật toán thay thế trang sao cho số lần xãy ra lỗi trang càng ít càng tốt
- Cùng một trang có thể mang vào bộ nhớ vài lần

Bộ nhớ ảo

Các bước thay thế trang

1. Tìm vị trí của trang trên đĩa
2. Tìm khung trang free:
   - Nếu có khung trang free, sử dụng khung
   - Nếu không có khung trang free, sử dụng trang
3. Thuật toán thay thế trang chọn một khung nạn nhân (**victim)**
4. Mang trang cần tham chiếu vào khung trang free; cập nhật lại bảng trang
5. Khởi động lại tiến trình

Bộ nhớ ảo
Thay thế trang



Bộ nhớ ảo

Các thuật toán thay thế trang

Algorithm	Comment
Optimal	Not implementable, but useful as a benchmark
NRU (Not Recently Used)	Very crude
FIFO (First-In, First-Out)	Might throw out important pages
Second chance	Big improvement over FIFO
Clock	Realistic
LRU (Least Recently Used)	Excellent, but difficult to implement exactly
NFU (Not Frequently Used)	Fairly crude approximation to LRU
Aging	Efficient algorithm that approximates LRU well
Working set	Somewhat expensive to implement
WSClock	Good efficient algorithm

Bộ nhớ ảo

Cách thực hiện

Hệ điều hành với phân trang

Bốn thời điểm HĐH tham gia vào phân trang

1. Khởi tạo tiến trình
   - xác định kích thước trang
   - khởi tạo bảng trang
2. Tiến trình thực thi
   - MMU thiết lập lại cho tiến trình mới
   - TLB giải phóng
3. Thời điểm lỗi trang
   - xác định địa chỉ ảo gây ra lỗi
   - thay thế trang
4. Thời điểm tiến trình kết thúc
   - giải phóng page table, pages

Bộ nhớ ảo

Cách thực hiện

Điều khiển lỗi trang (1)

1. Phần cứng bẫy lỗi đến kernel
2. Các thanh ghi được lưu lại
3. HĐH xác định trang lỗi
4. HĐH kiểm tra các khung trang có thể thay thế
5. Nếu khung trang có chỉnh sửa, cần phải ghi trang đó ra đĩa để cập nhật dữ liệu

Bộ nhớ ảo

Cách thực hiện

Điều khiển lỗi trang (2)

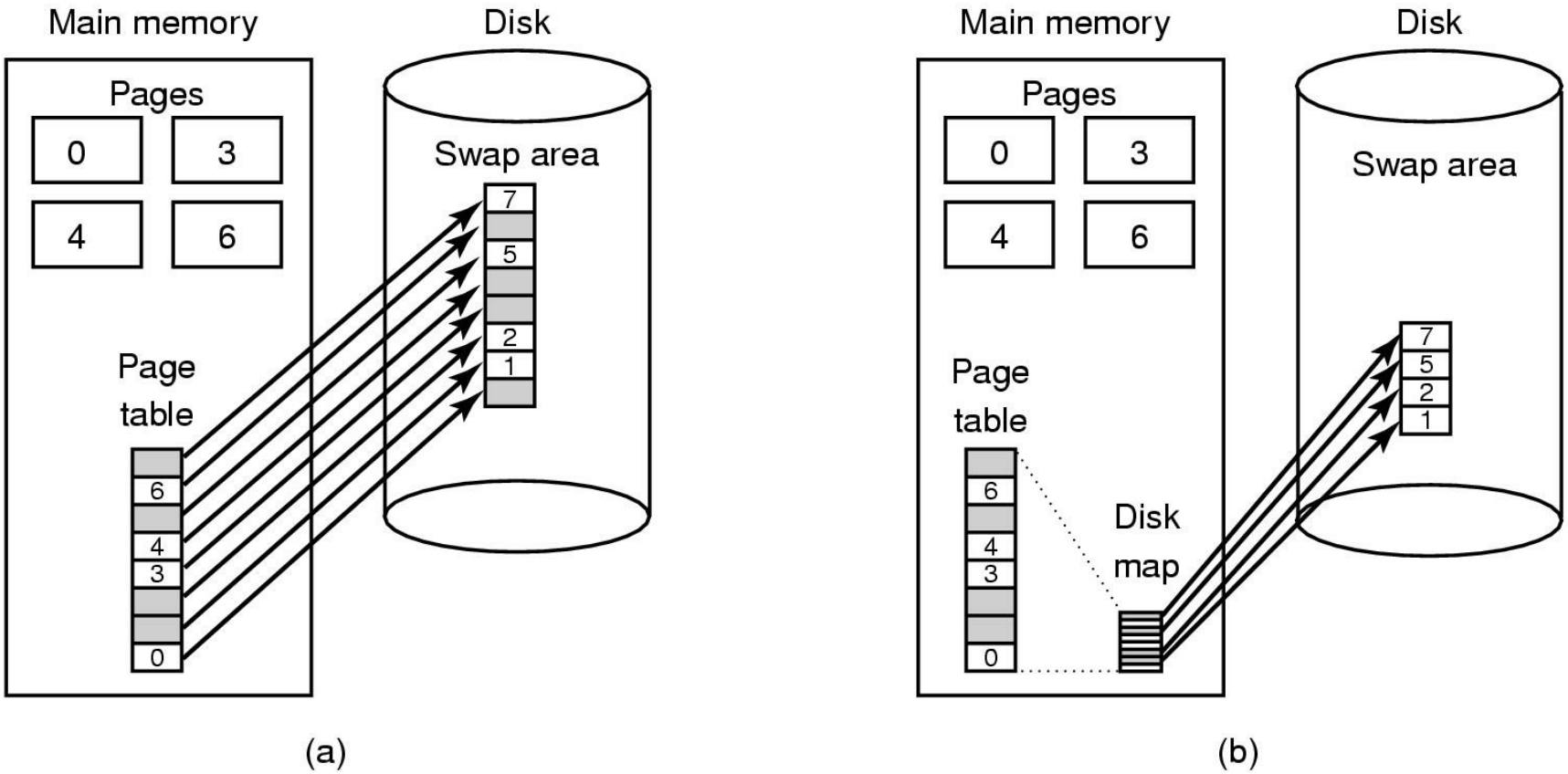
6. HĐH nạp trang mới từ đĩa vào
7. Cập nhật lại bảng trang
8. Khởi động lại lệnh gây ra lỗi
9. Tiến trình lỗi được lên lịch
10. Phục hồi các thanh ghi, Chương trình tiếp tục

Bộ nhớ ảo
Cách thực hiện
Khóa các trang trong bộ nhớ

- Thỉnh thoảng tương tác giữa bộ nhớ ảo và thiết bị Vào/Ra
- Tiến trình yêu cầu đọc (read) từ thiết bị vào bộ đệm (buffer)
  - trong khi chờ I/O, tiến trình khác bắt đầu
  - có lỗi trang
  - buffer cho tiến trình thứ nhất có thể được chọn để đưa ra
- Cần phải chỉ rõ một số trang để khóa
  - được miễn chọn các trang đích

Bộ nhớ ảo

Cách thực hiện: Kho sao lưu - Backing Store



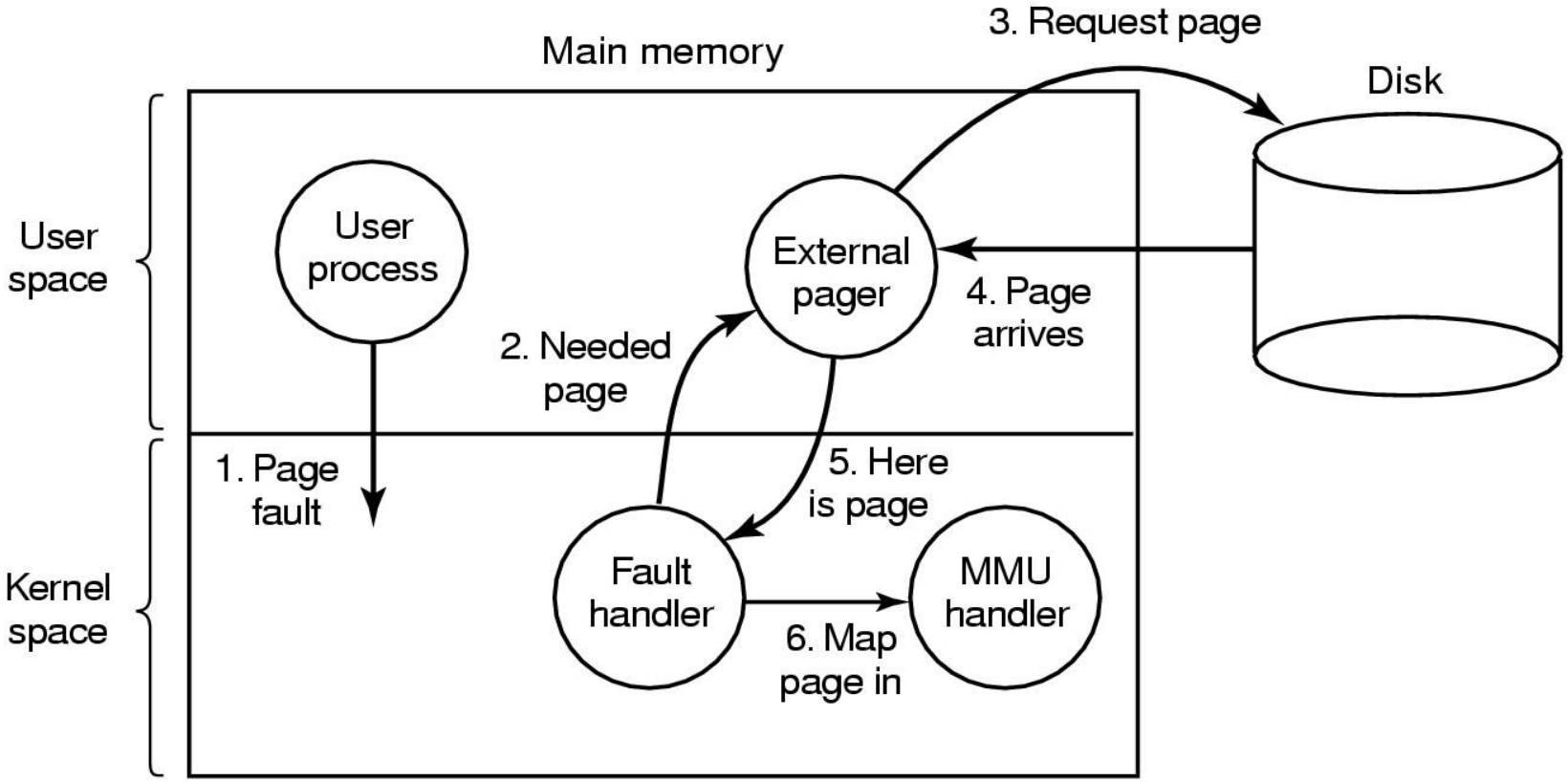
(a) Phân trang đến vùng swap tĩnh

(b) Sao lưu các trang động

Bộ nhớ ảo

Cách thực hiện

Tách biệt Chính sách và Cơ chế

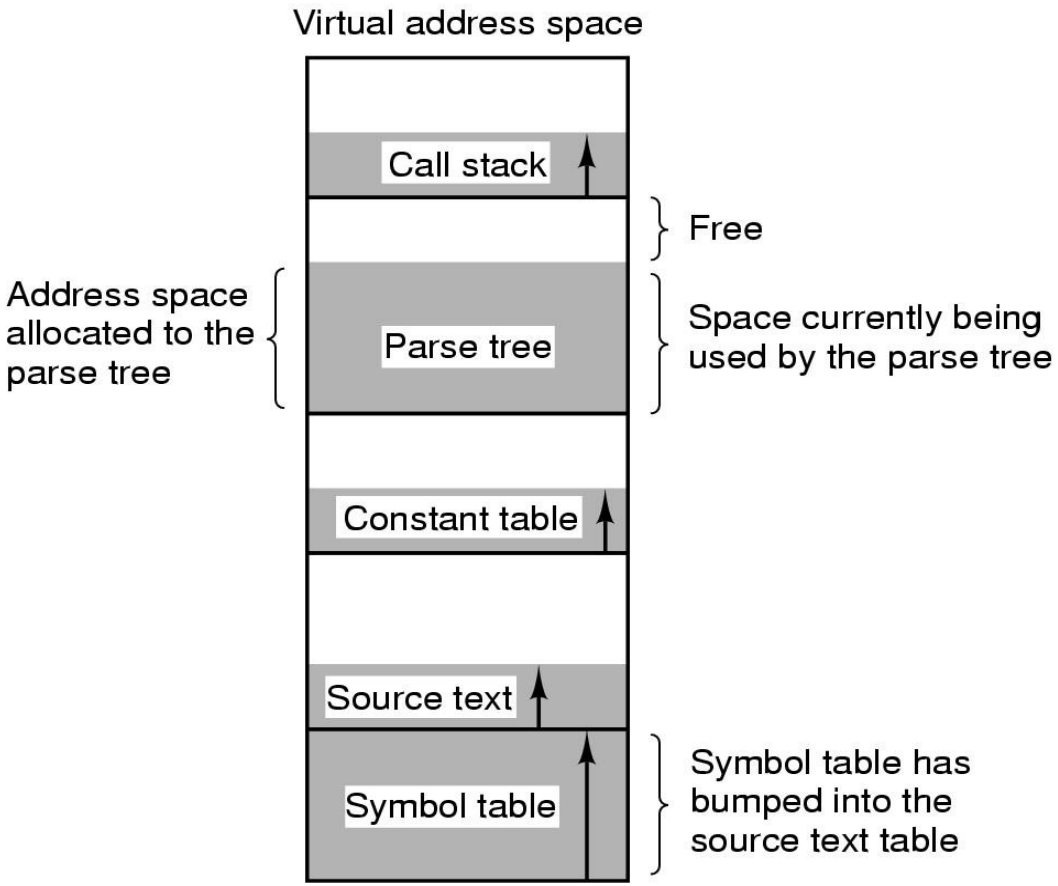


Điều khiển lỗi trang với một external pager

Bộ nhớ ảo
Phân đoạn-Segmentation

Bộ nhớ ảo

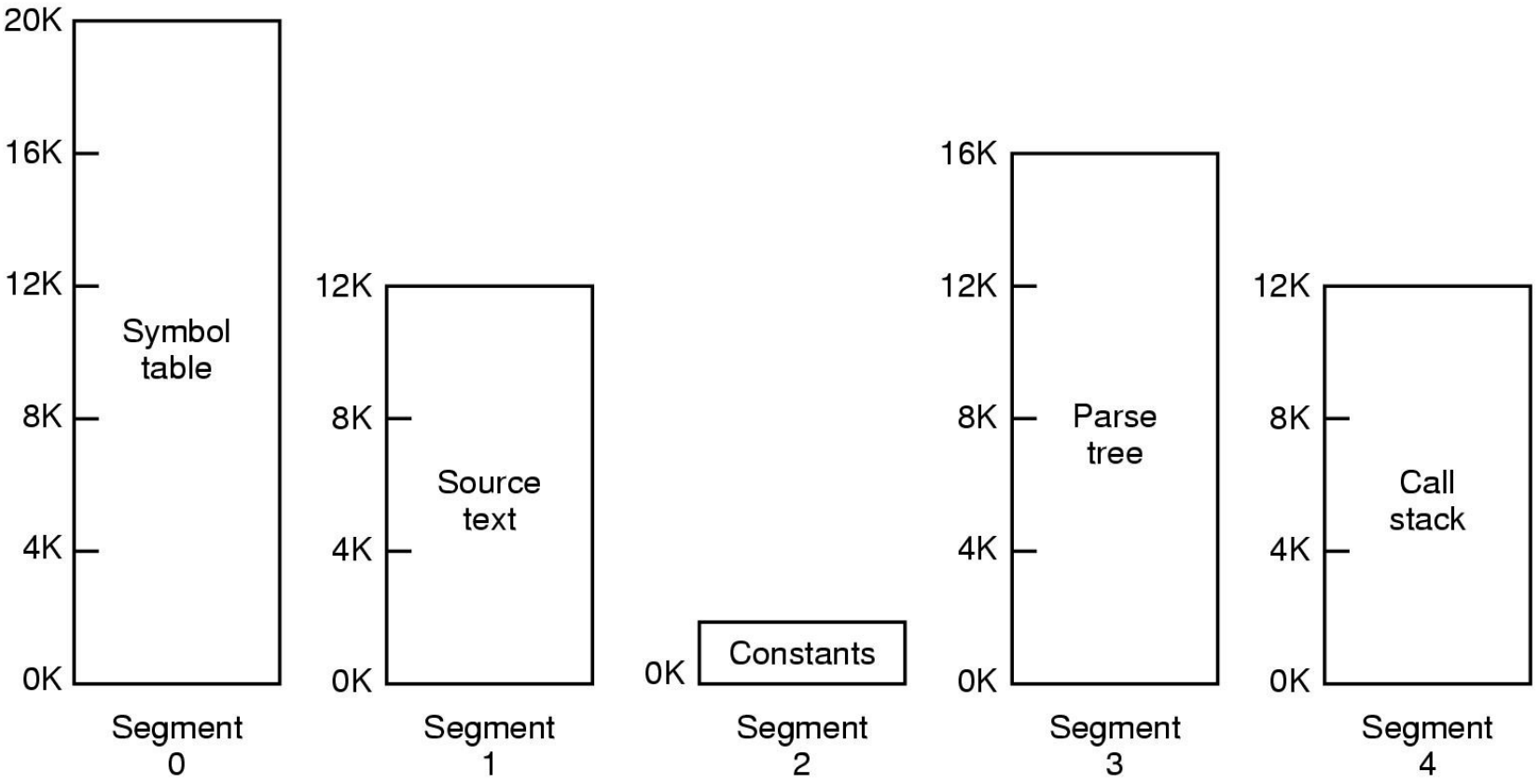
Phân đoạn - Segmentation (1)



- Không gian địa chỉ một chiều với sự phát triển của các bảng
- Một bảng có thể chạm đến bảng khác

Bộ nhớ ảo

Phân đoạn - Segmentation (2)



Cho phép mỗi bảng tăng lên hoặc co lại một cách độc lập

Bộ nhớ ảo

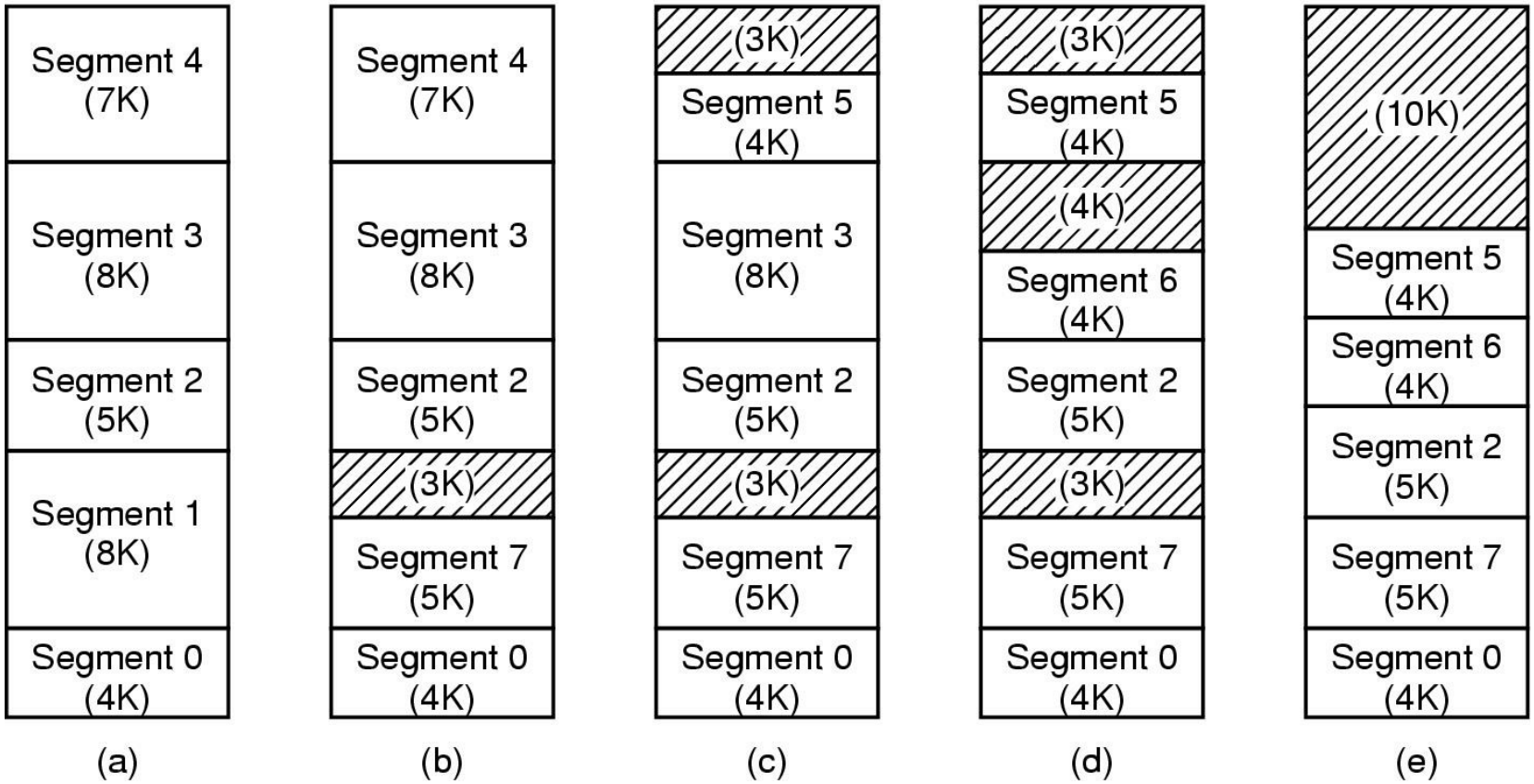
Phân đoạn - Segmentation (3)

Consideration	Paging	Segmentation
Need the programmer be aware that this technique is being used?	No	Yes
How many linear address spaces are there?	1	Many
Can the total address space exceed the size of physical memory?	Yes	Yes
Can procedures and data be distinguished and separately protected?	No	Yes
Can tables whose size fluctuates be accommodated easily?	No	Yes
Is sharing of procedures between users facilitated?	No	Yes
Why was this technique invented?	To get a large linear address space without having to buy more physical memory	To allow programs and data to be broken up into logically independent address spaces and to aid sharing and protection

So sánh giữa phân trang và phân đoạn

Bộ nhớ ảo

Thực hiện phân đoạn (4)

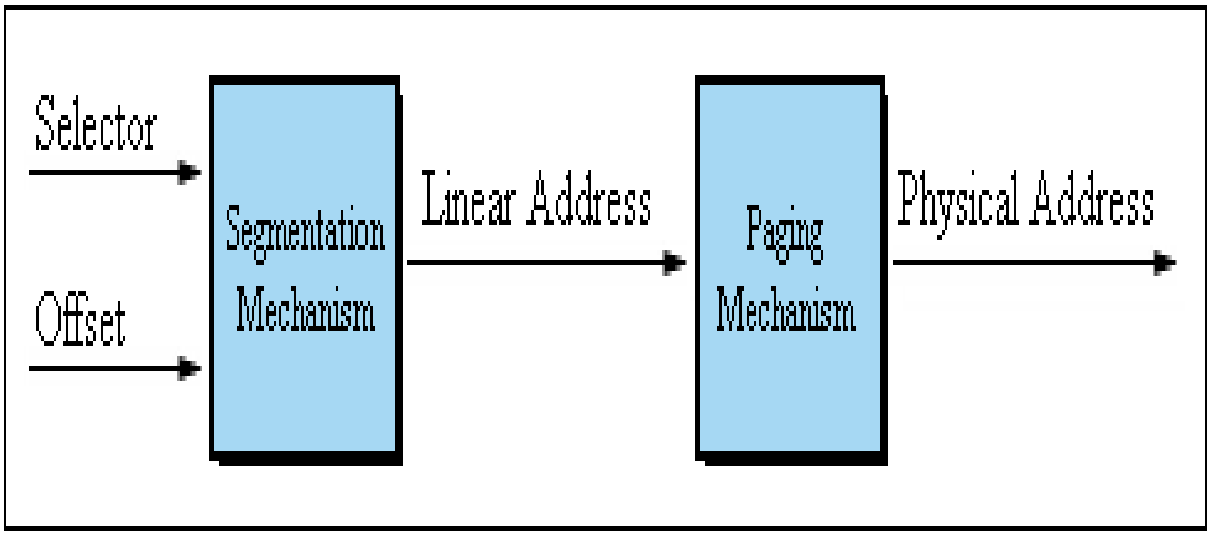


(a)-(d) Quá trình hình thành checkerboarding

(e) Loại bỏ checkerboarding bằng cách nén

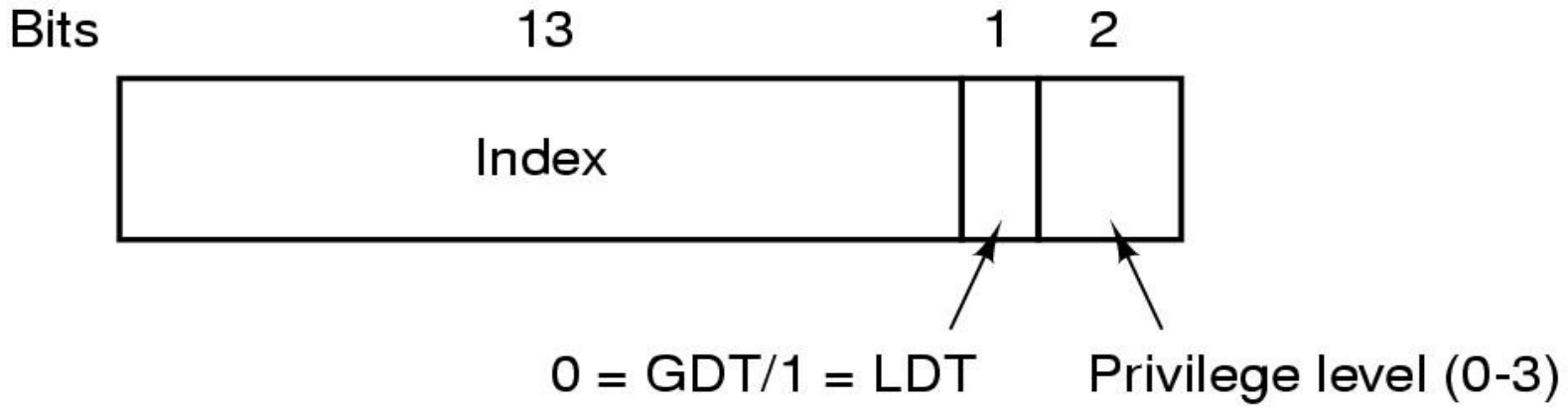
Bộ nhớ ảo

Phân đoạn với phân trang: Pentium (1)



Bộ nhớ ảo

Phân đoạn với phân trang: Pentium (2)

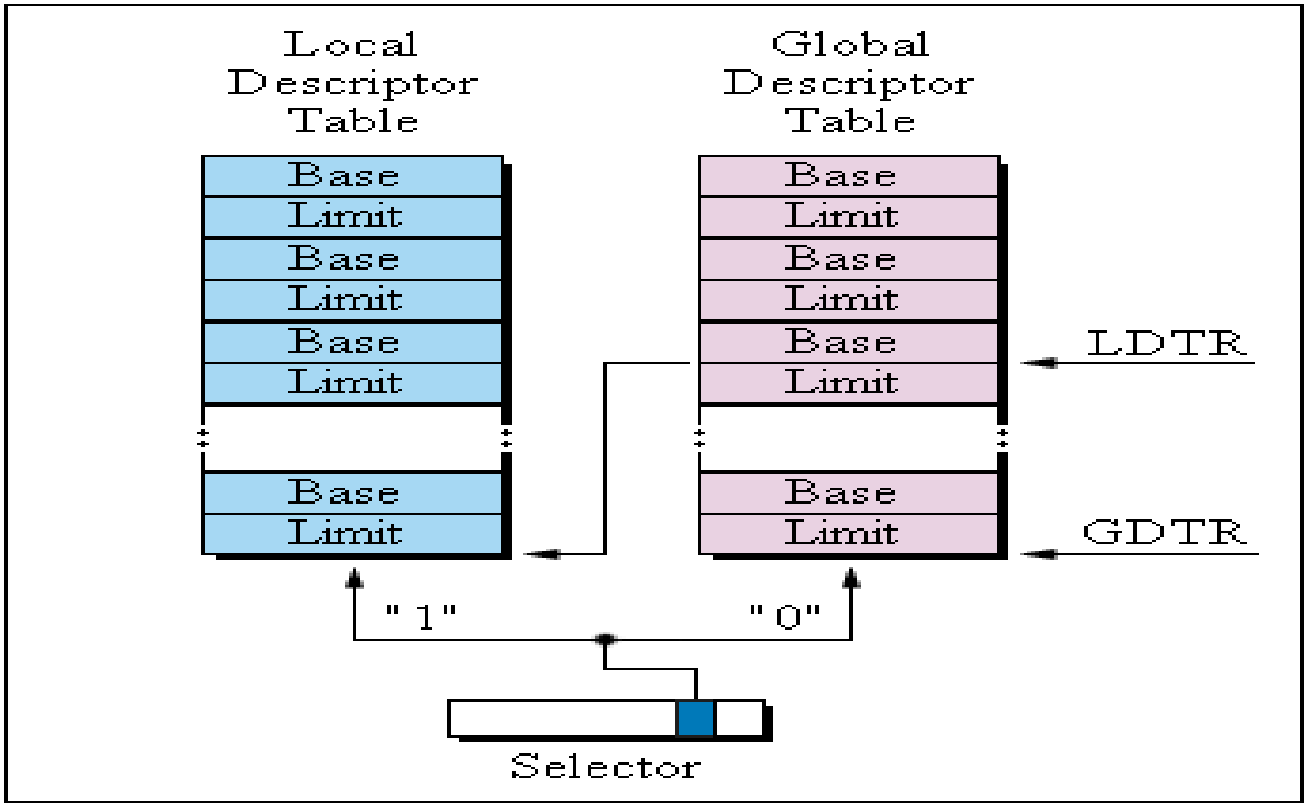


Bộ chọn Pentium (Pentium selector)

GDT (Global Descriptor Table), LDT (Local Descriptor Table)

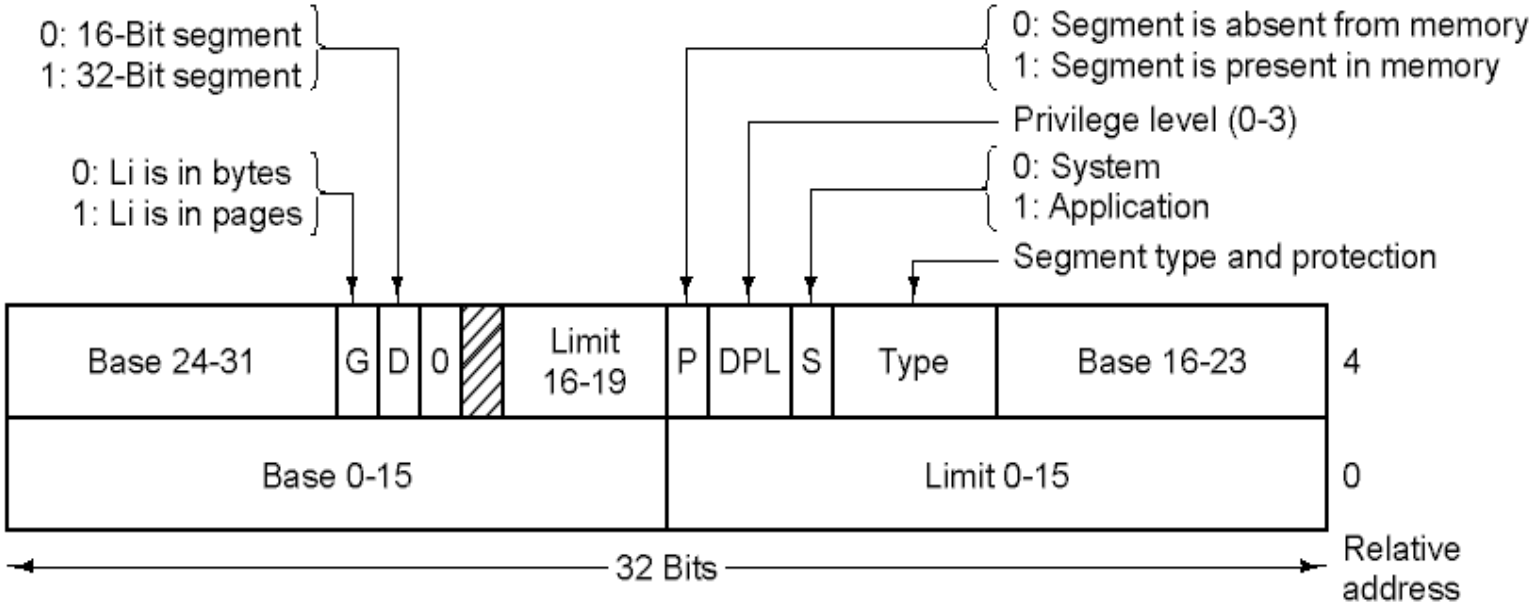
Bộ nhớ ảo

Phân đoạn với phân trang: Pentium (3)



Bộ nhớ ảo

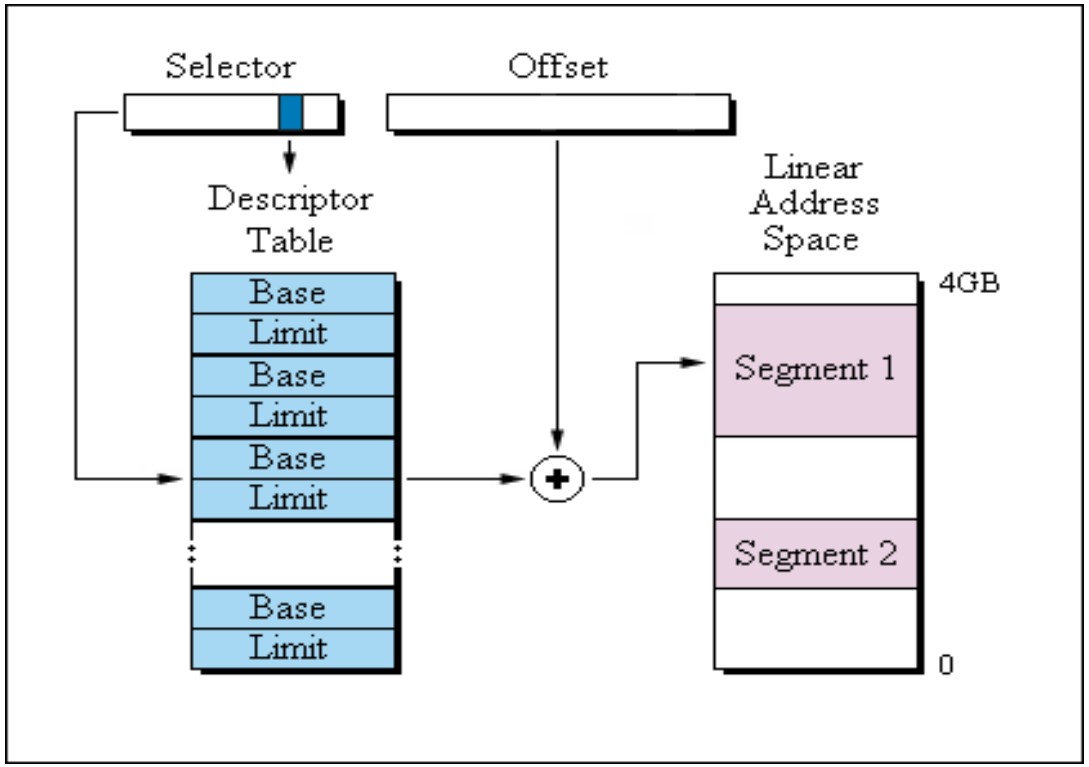
Phân đoạn với phân trang: Pentium (4)



- Bộ mô tả đoạn code chạy trên Pentium
- Đoạn Data hơi khác

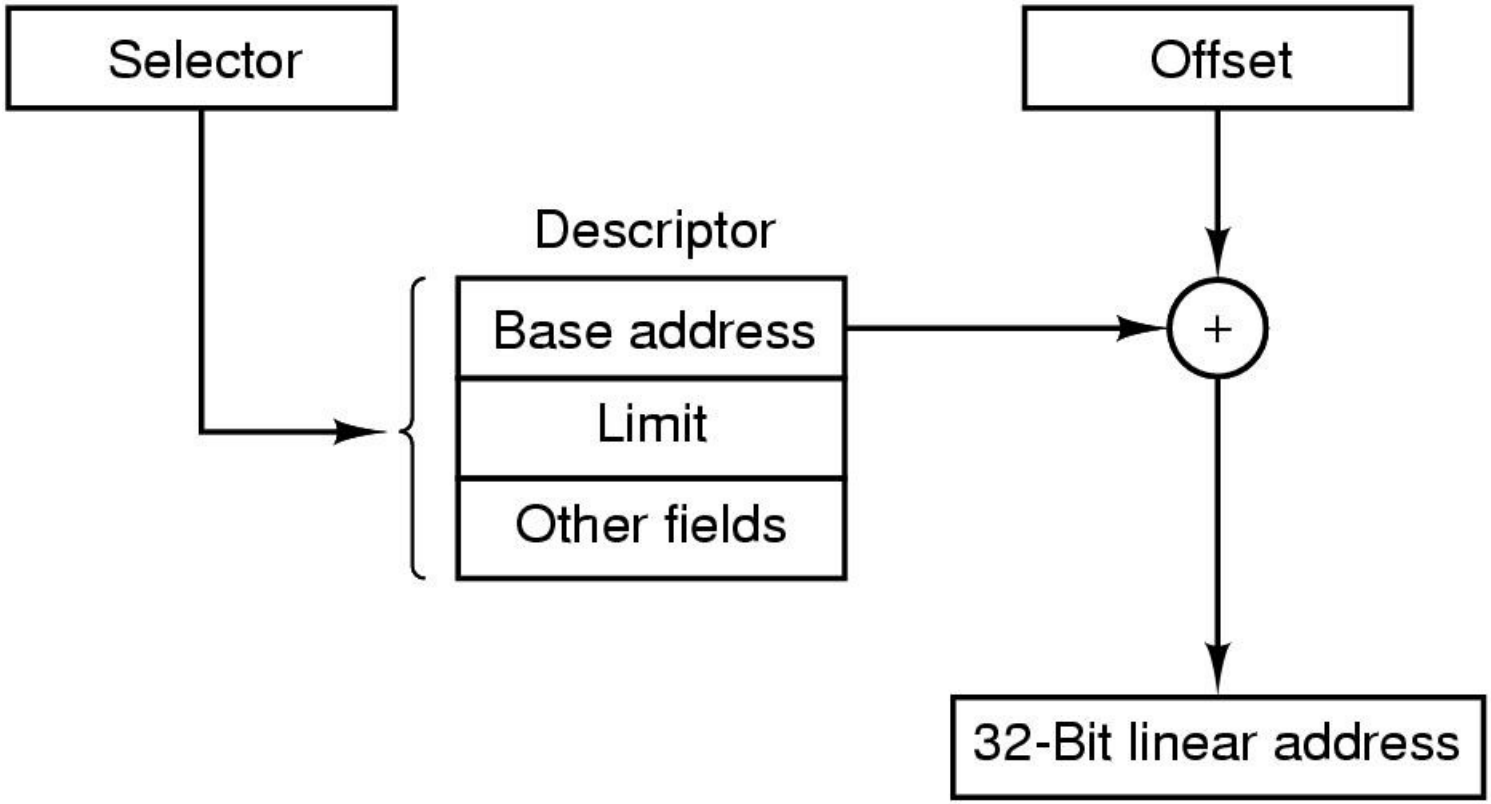
Bộ nhớ ảo

Phân đoạn với phân trang: Pentium (5)



Bộ nhớ ảo

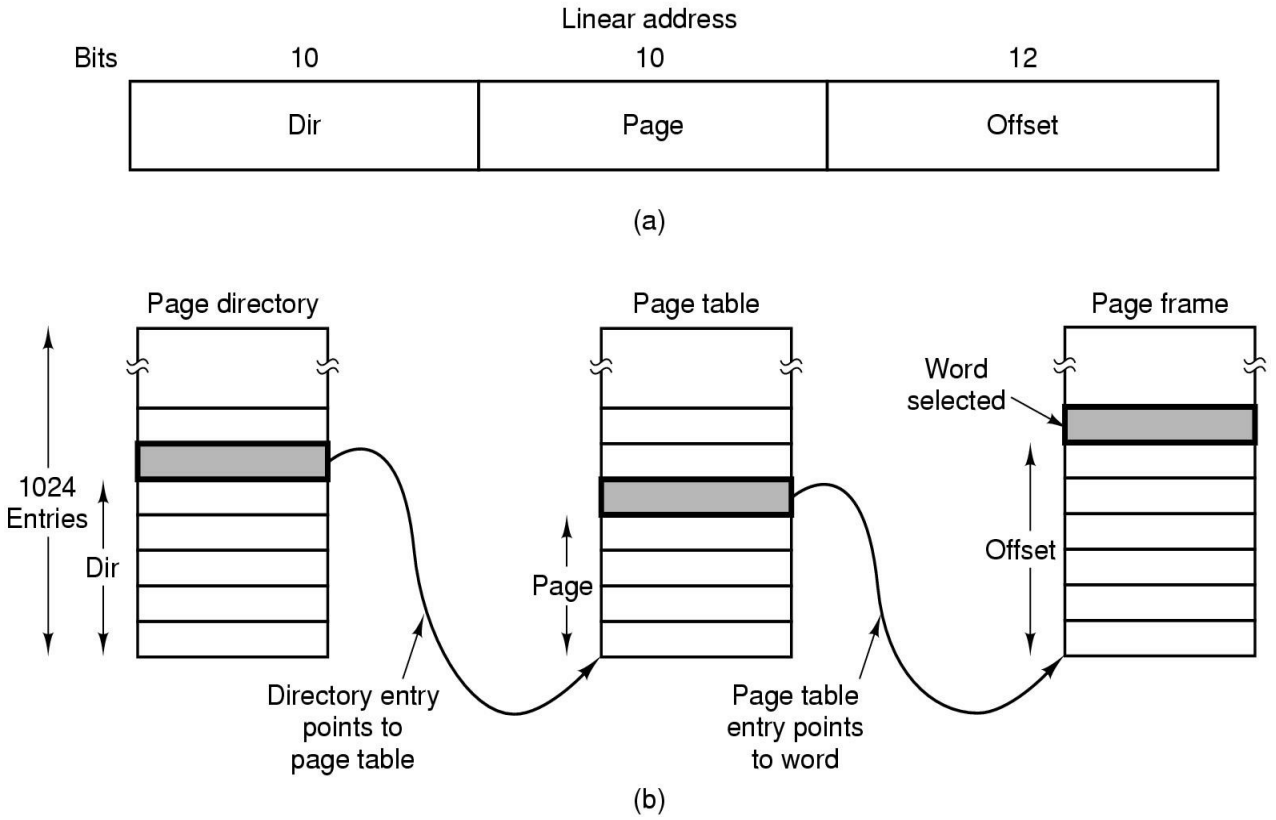
Phân đoạn với phân trang: Pentium (6)



Chuyển đổi một cặp (selector, offset) đến địa chỉ tuyến tính

Bộ nhớ ảo

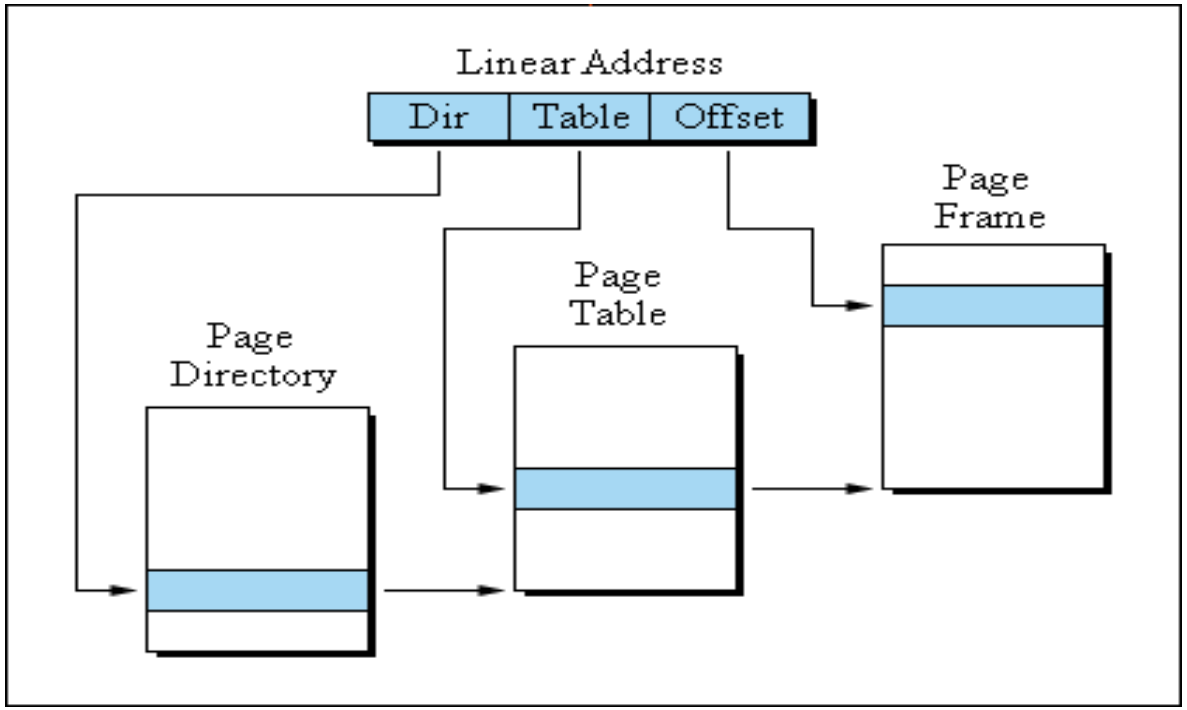
Phân đoạn với phân trang: Pentium (7)



Ánh xạ địa chỉ tuyến tính thành địa chỉ vật lý

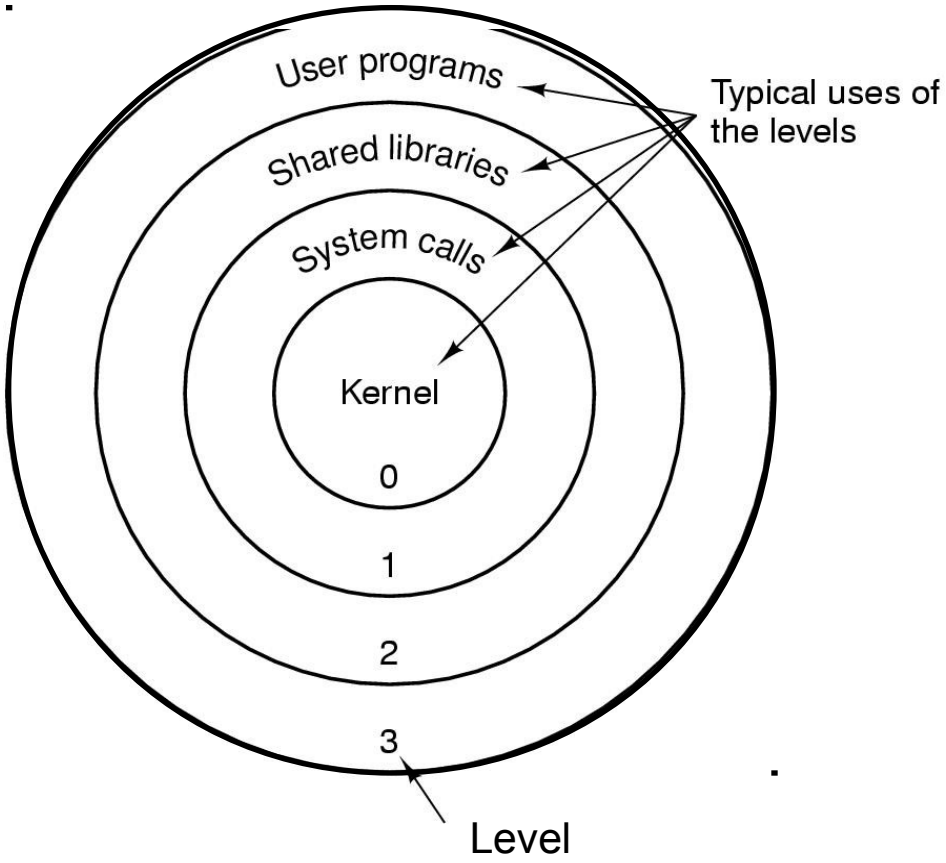
Bộ nhớ ảo

Phân đoạn với phân trang: Pentium (8)



Bộ nhớ ảo

Phân đoạn với phân trang: Pentium (9)



Mức bảo vệ trên Pentium