QUẢN LÝ BỘ NHỚ

- Các vấn đề phát sinh khi quản lý bộ nhớ
- Các mô hình cấp phát bộ nhớ
- Bộ nhớ ảo

CÁC VẤN ĐỀ PHÁT SINH KHI QUẢN LÝ BỘ NHỚ

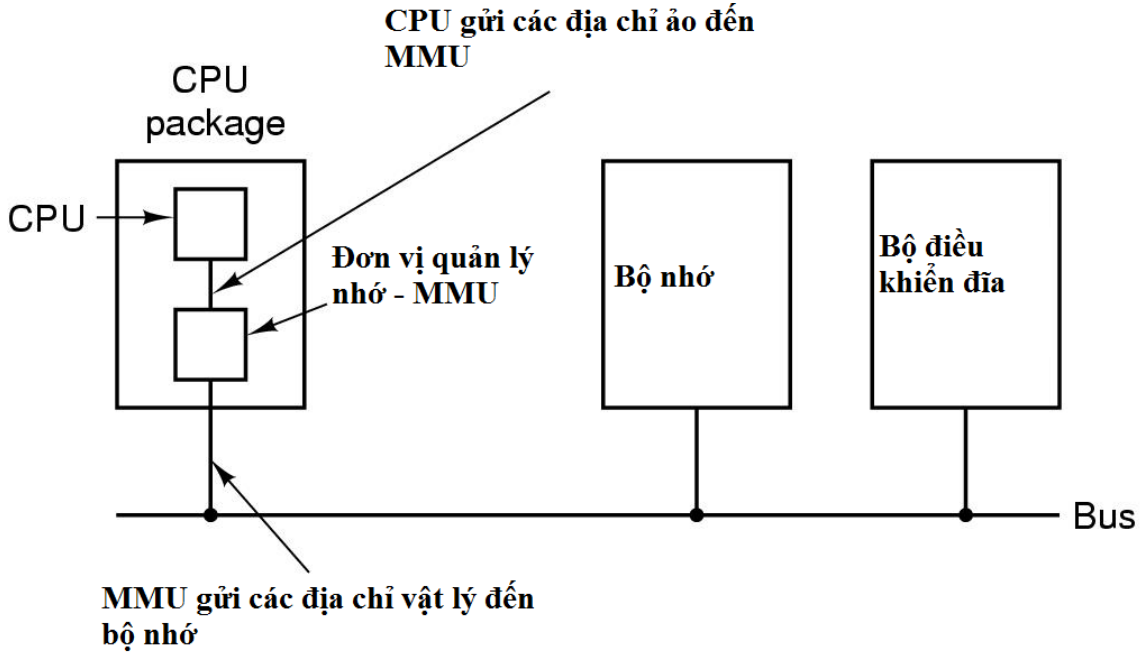
- Chuyển đổi địa chỉ tương đối trong chương trình thành địa chỉ thực trong bộ nhớ chính.
- Quản lý bộ nhớ đã cấp phát và chưa cấp phát.
- Các kỹ thuật cấp phát bộ nhớ sao cho:
  - Ngăn chặn các tiến trình xâm phạm đến vùng nhớ đã được cấp phát cho tiến trình khác.
  - Cho phép nhiều tiến trình có thể dùng chung một phần bộ nhớ của nhau.
  - Mở rộng bộ nhớ để có thể lưu trữ được nhiều tiến trình đồng thời.

Chuyển đổi địa chỉ tương đối sang tuyệt đối

- ***Thời điểm biên dịch (compile time)***
- ***Thời điểm nạp (load time)***
- ***Thời điểm xử lý (execution time)***

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

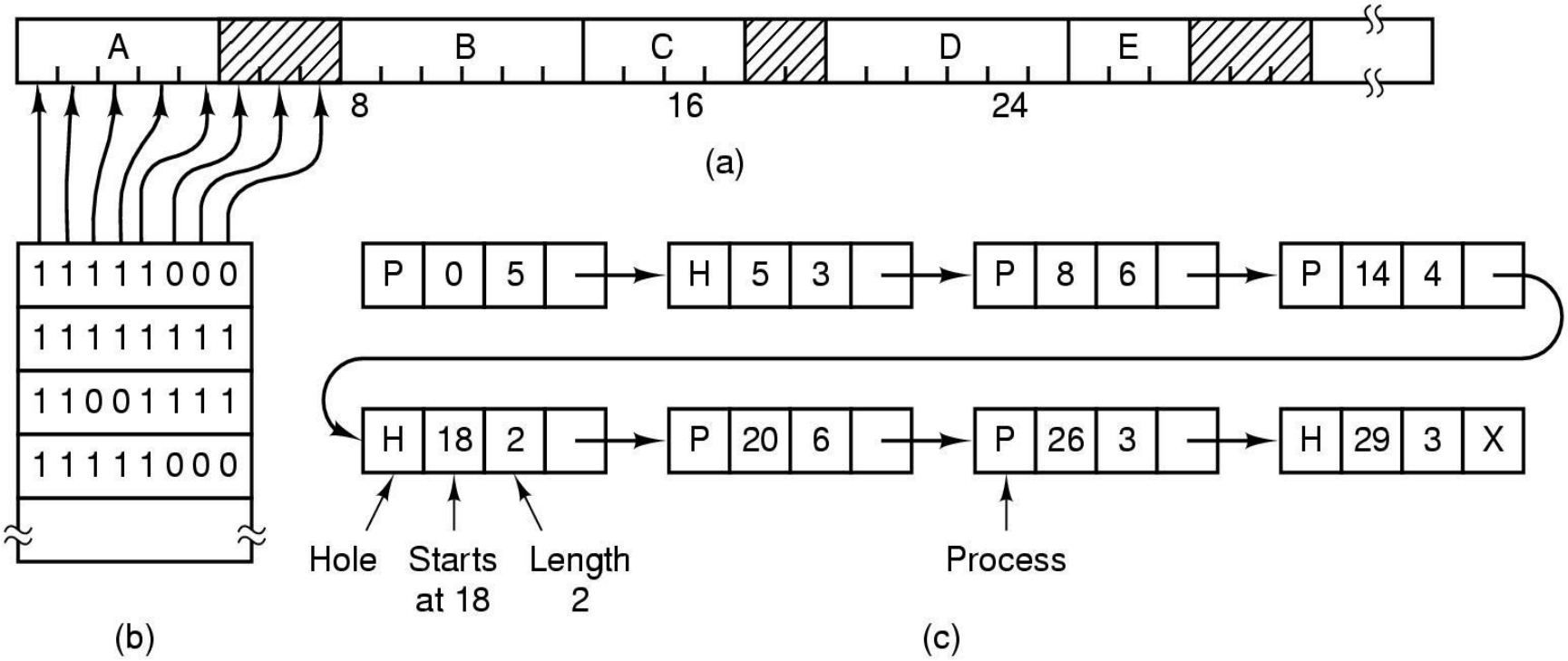
- ***Địa chỉ ảo (địa chỉ logic):*** là địa chỉ do CPU tạo ra.
- ***Địa chỉ vật lý (địa chỉ physic):*** là địa chỉ thực trong bộ nhớ chính.
- ***Không gian địa chỉ ảo của tiến trình:*** là tập hợp tất cả các địa chỉ ảo của một tiến trình.
- ***Không gian địa chỉ vật lý của tiến trình:*** là tập hợp tất cả các địa chỉ vật lý tương ứng với các địa chỉ ảo.



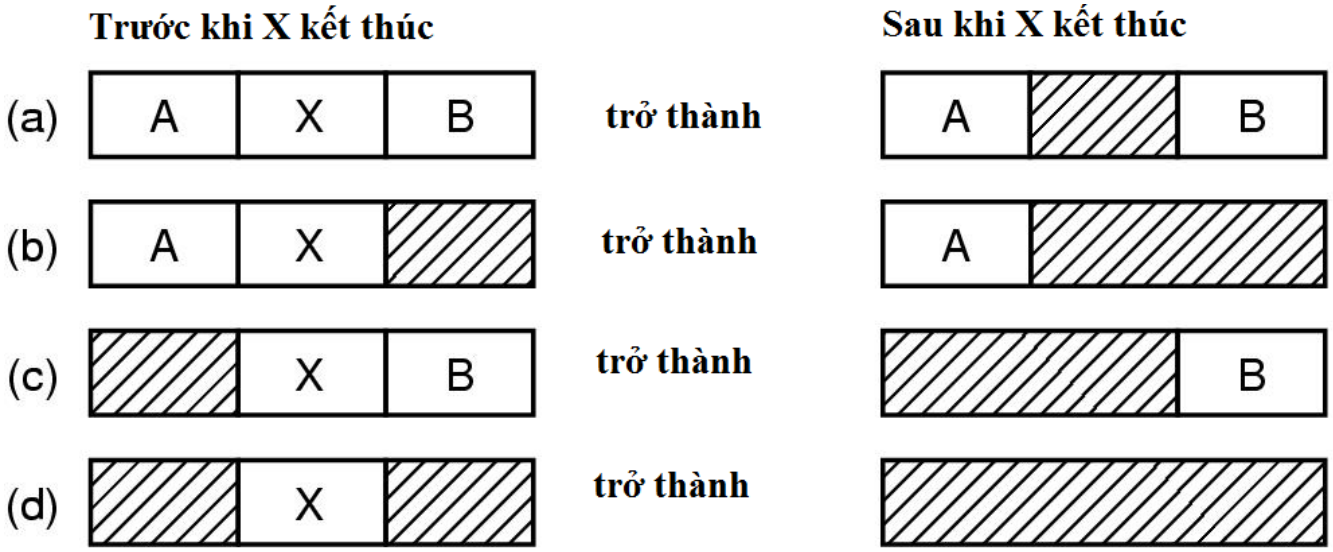
Quản lý bộ nhớ đã cấp phát và chưa cấp phát

- **Các phương pháp quản lý việc cấp phát bộ nhớ**
  - ***Sử dụng dãy bit*** : bít thứ i bằng 1 là khối thứ i đã cấp phát, bằng 0 là chưa cấp phát.
  - ***Sử dụng danh sách liên kết***: mỗi nút của danh sách liên kết lưu thông tin một vùng nhớ chứa tiến trình (P) hay vùng nhớ trống giữa hai tiến trình (H).
- **Các thuật toán chọn một đoạn trống:**
  - ***First-fit***: chọn đoạn trống đầu tiên đủ lớn.
  - ***Best-fit***: chọn đoạn trống nhỏ nhất nhưng đủ lớn để thỏa mãn nhu cầu.
  - ***Worst-fit*** : chọn đoạn trống lớn nhất.

Quản lý việc cấp phát bộ nhớ bằng dãy bit hoặc danh sách liên kết



Quản lý việc cấp phát bộ nhớ



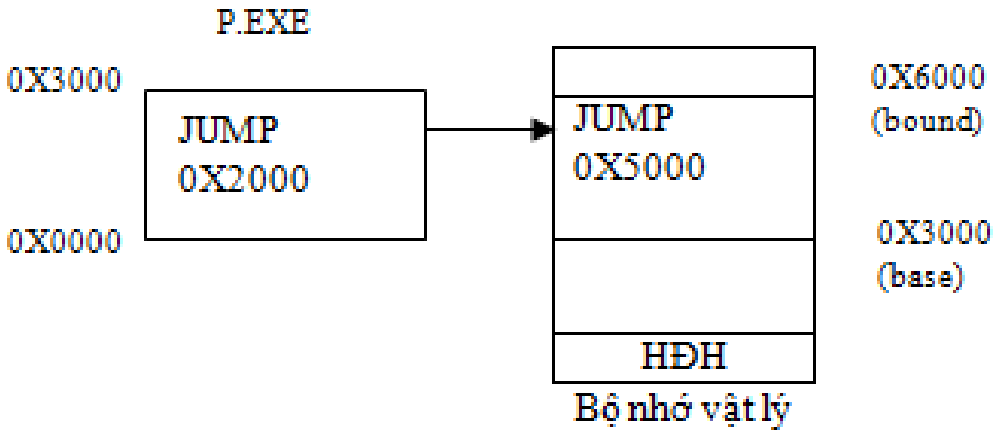
Các trường hợp có thể xảy ra trước và sau khi tiến trình X kết thúc

CÁC MÔ HÌNH CẤP PHÁT BỘ NHỚ

- Cấp phát liên tục: tiến trình được nạp vào một vùng nhớ liên tục.
  - Linker-Loader
  - Base & Limit
- Cấp phát không liên tục: tiến trình được nạp vào một vùng nhớ không liên tục
  - Mô hình phân đoạn
  - Mô hình phân trang
  - Mô hình phân đoạn kết hợp phân trang

Mô hình Linker_Loader

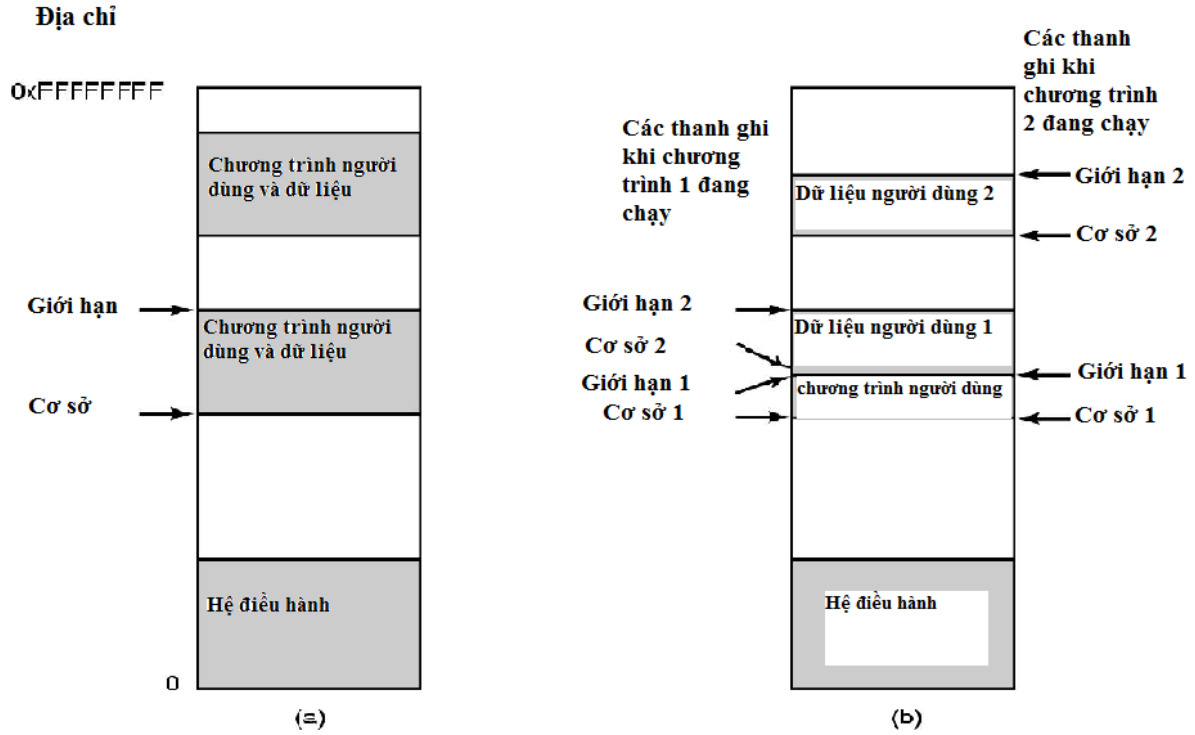
- địa chỉ tuyệt đối = địa chỉ bắt đầu nạp tiến trình + địa chỉ tương đối.



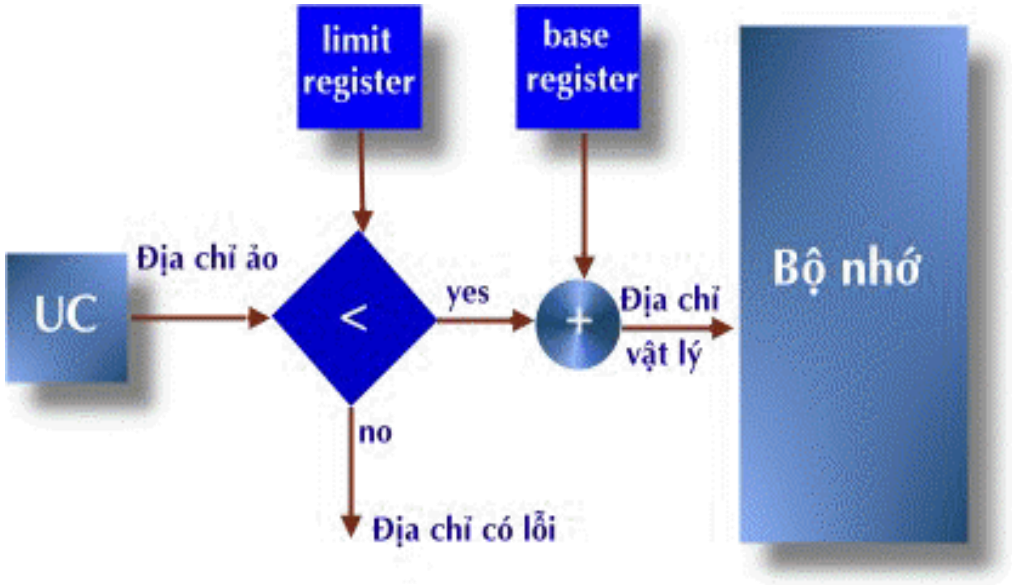
- Không thể di chuyển tiến trình trong bộ nhớ
- Không thể bảo vệ một tiến trình bị một tiến trình khác truy xuất bộ nhớ của tiến trình một cách trái phép

Mô hình Base & Limit

- Thanh ghi nền (base register): giữ địa chỉ bắt đầu của vùng nhớ cấp phát cho tiến trình
- Thanh ghi giới hạn (limit register): giữ kích thước của tiến trình.



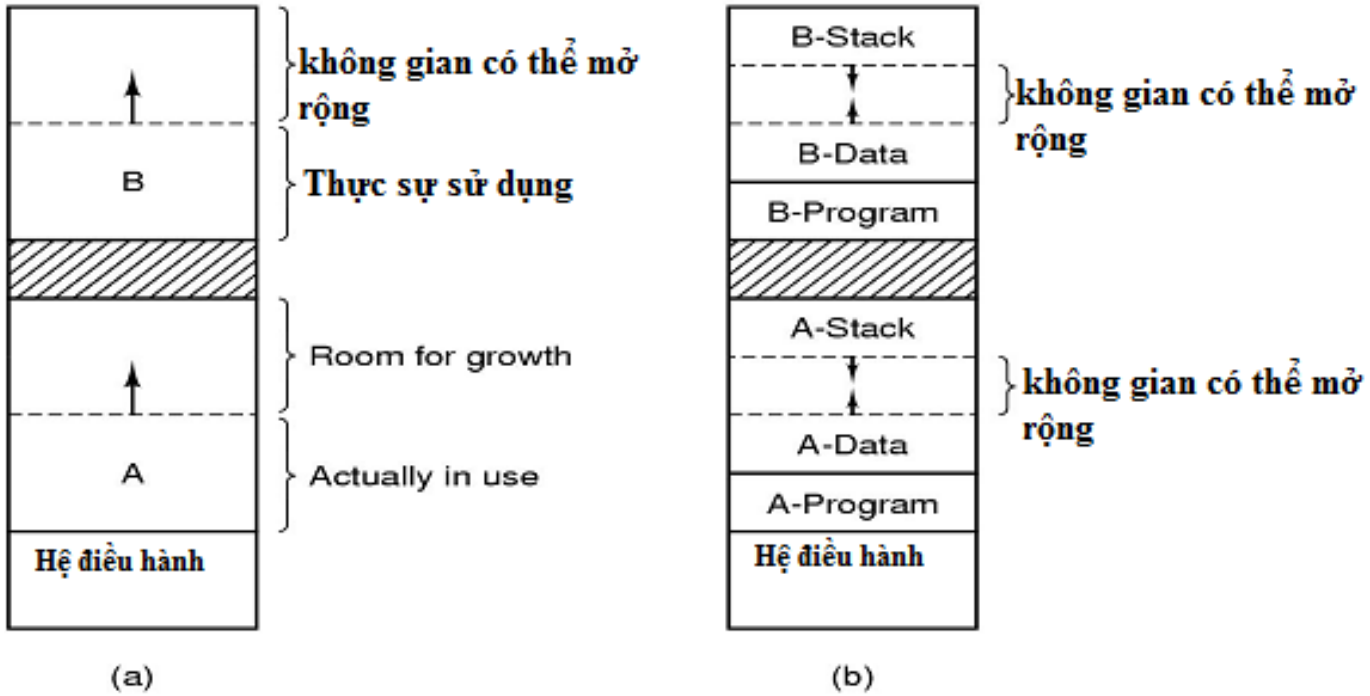
Cơ chế MMU trong mô hình *Base & Limit*



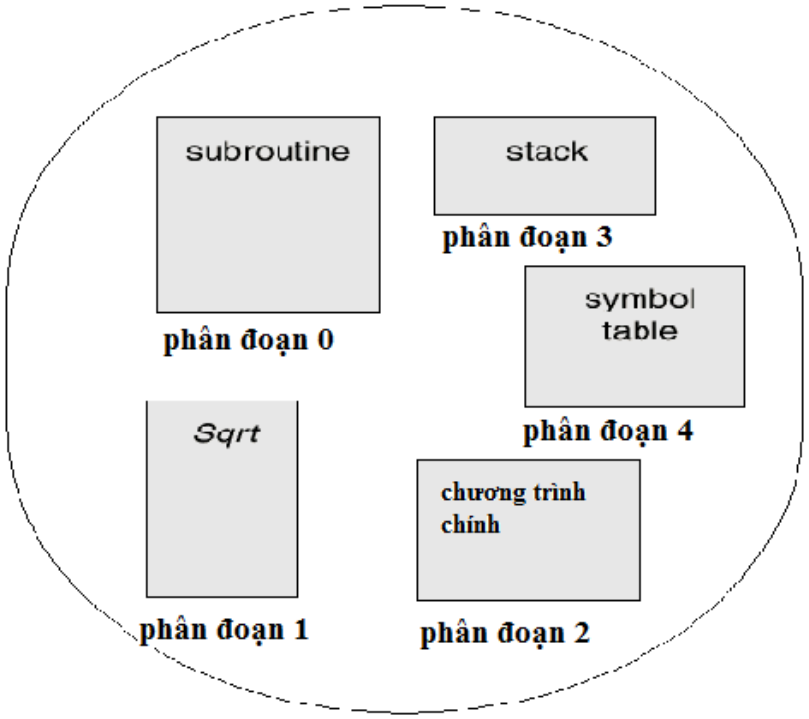
- Có thể di chuyển các chương trình trong bộ nhớ.
- Có thể có hiện tượng phân mảnh ngoại vi (external fragmentation)

Mô hình Base & Limit – Khi kích thước của tiến trình tăng trưởng trong qúa trình xử lý

- ***Dời chỗ tiến trình.***
- ***Cấp phát dư vùng nhớ cho tiến trình***
- ***Swapping***



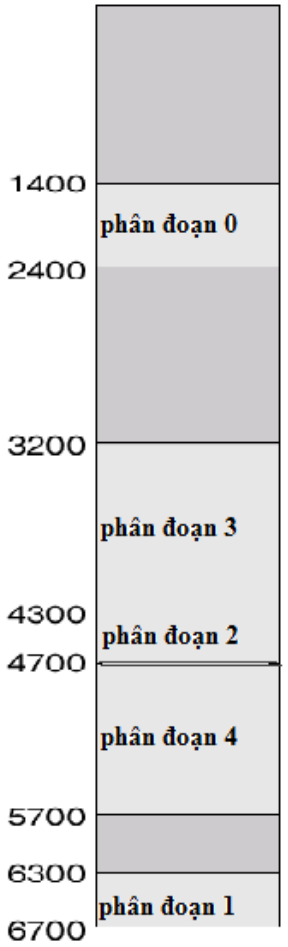
Mô hình phân đoạn (Segmentation)



không gian địa chỉ logic

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

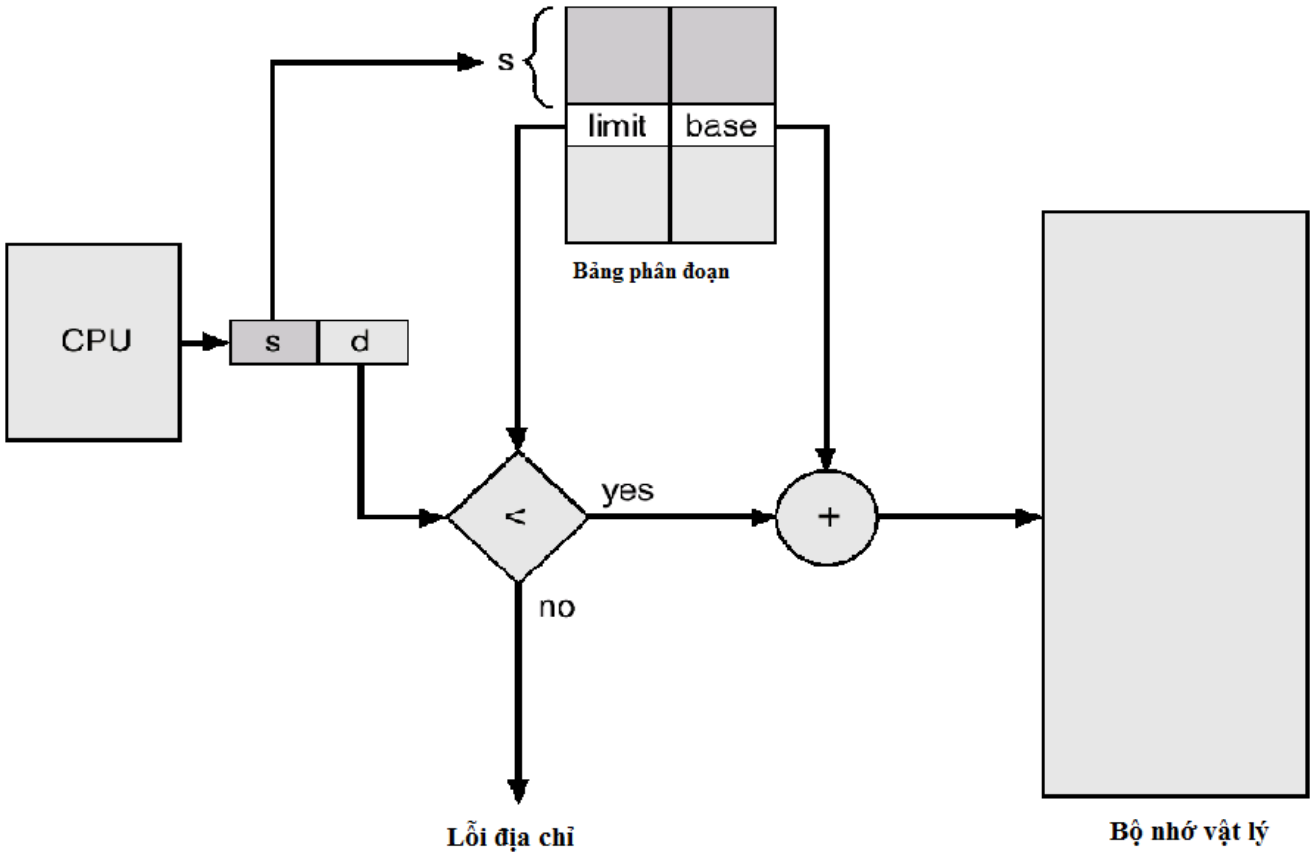
Bảng phân đoạn



Bộ nhớ vật lý

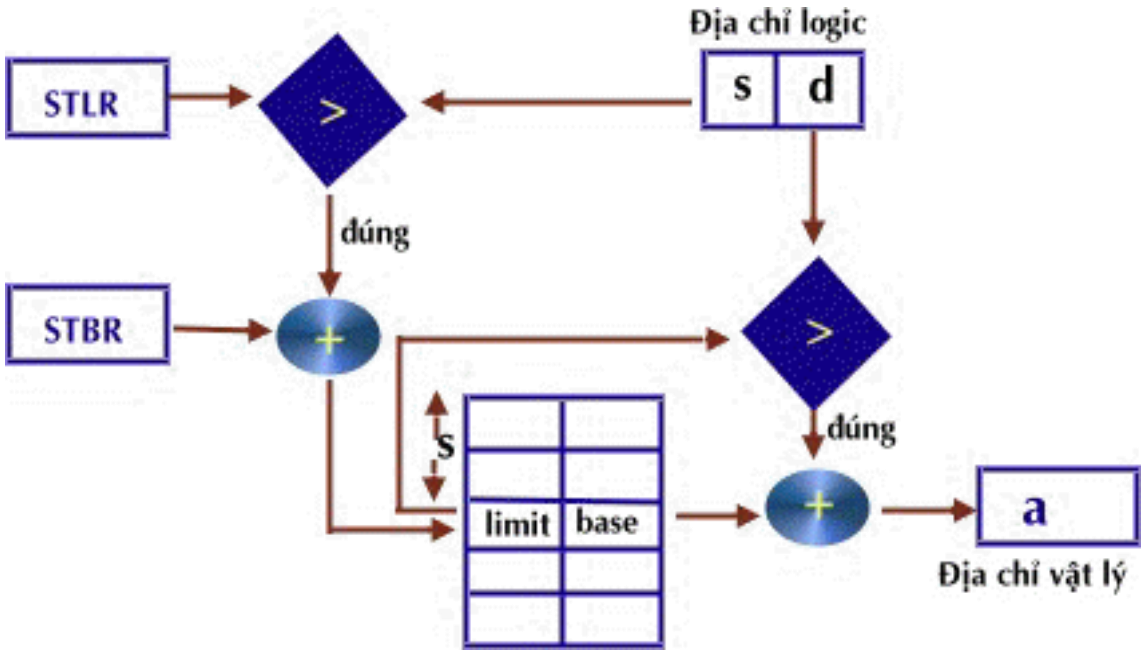
Cơ chế MMU trong kỹ thuật phân đoạn

Địa chỉ vật lý =base + d

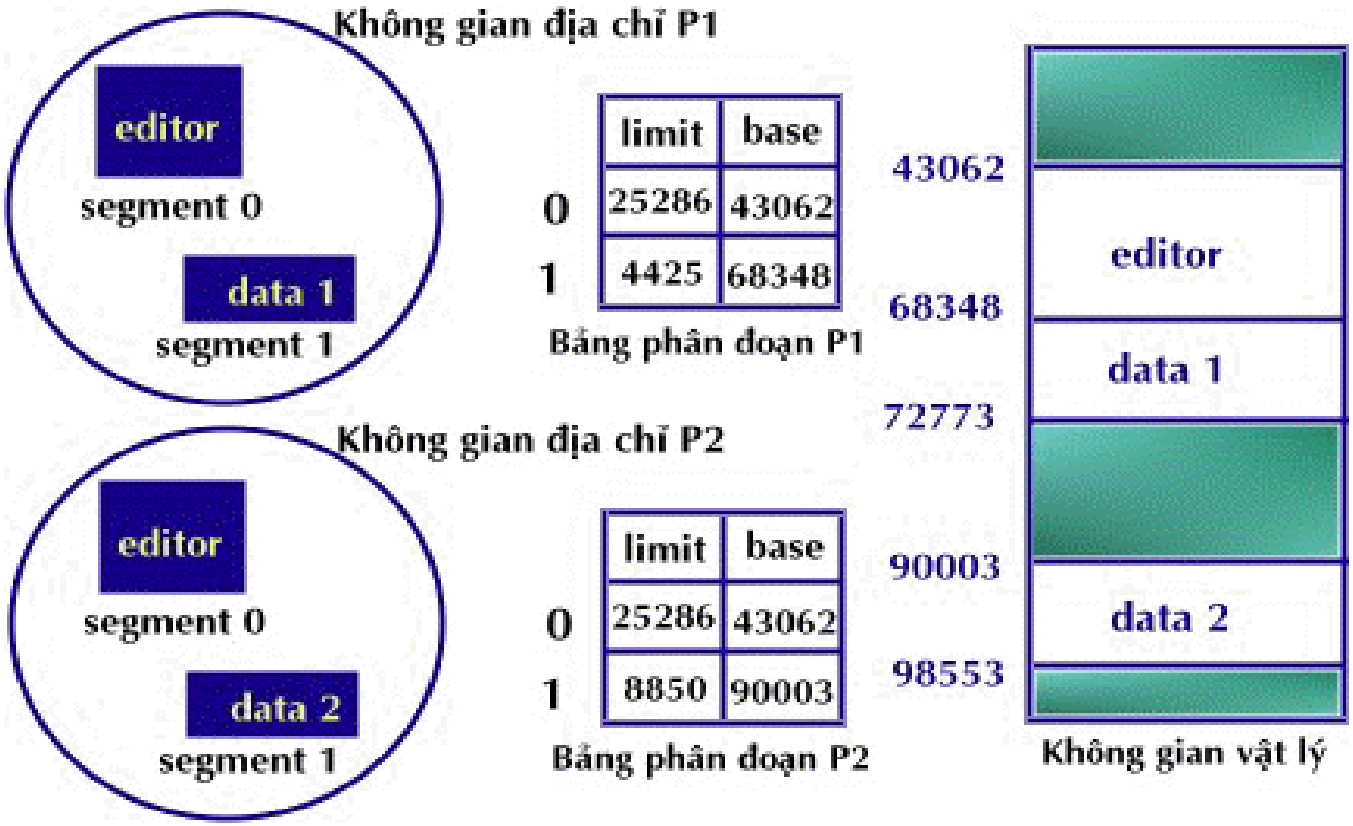


Cài đặt bảng phân đoạn

- STBR (Segment Table Base Register) để lưu địa chỉ bắt đầu của bảng phân đoạn.
- STLR (Segment Table Limit Register) lưu số phân đoạn.



Chia sẻ phân đoạn



MMU gán hai phần tử trong hai bảng phân đoạn của hai tiến trình cùng giá trị

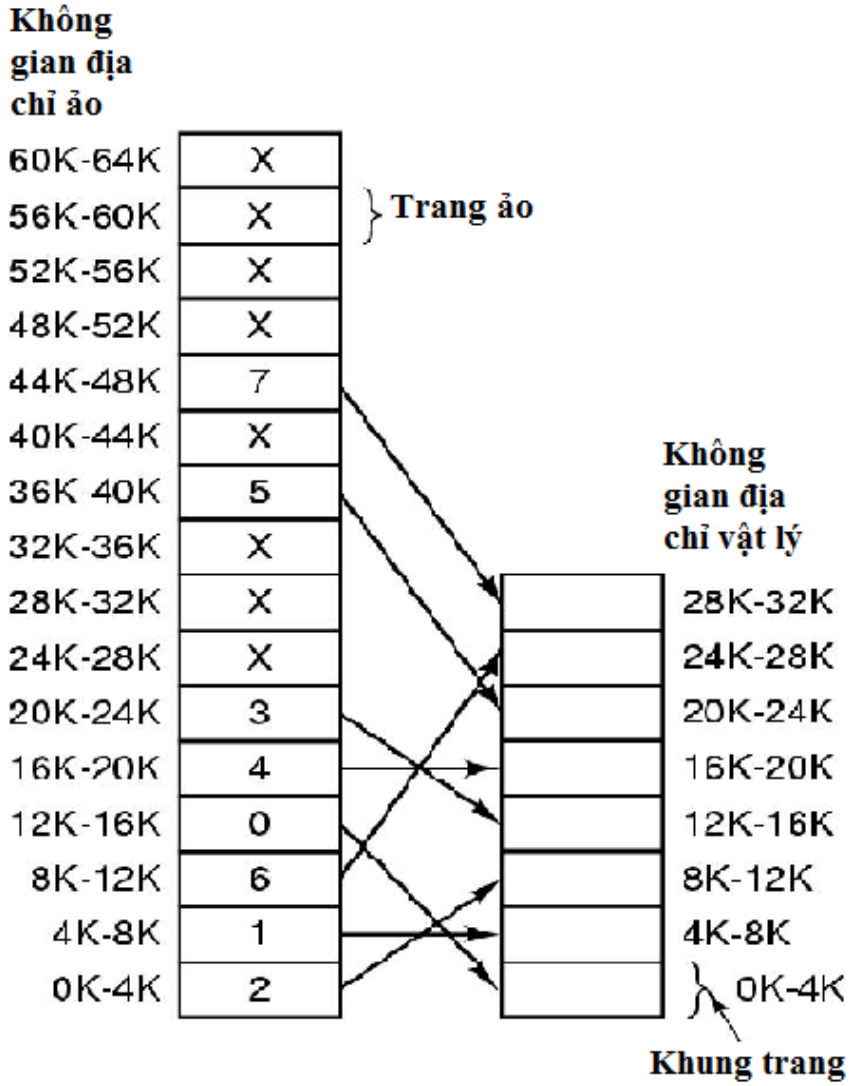
Bảo vệ phân đoạn

- Attribute: R (chỉ đọc), X (thực thi), W (ghi),…

Nhận xét về kỹ thuật phân đoạn:

- ❑ Hiện tượng phân mảnh ngoại vi vẫn xảy ra.
- ❑ Ưu điểm: mã chương trình và dữ liệu được tách riêng -> dễ dàng bảo vệ mã chương trình và dễ dàng dùng chung dữ liệu hoặc hàm

Mô hình phân trang (Paging)



Page 3
Page 2
Page 1
Page 0

Không gian địa chỉ ảo

7	Page 1	7168
6		6144
5	Page 0	5120
4		4096
3		3072
2	Page 3	2048
1		1024
0	Page 2	0000

Không gian địa chỉ vật lý

3	2
2	0
1	7
0	5

Bảng trang

Bộ nhớ vật lý: khung trang (page frame).
Không gian địa chỉ ảo: trang (page).

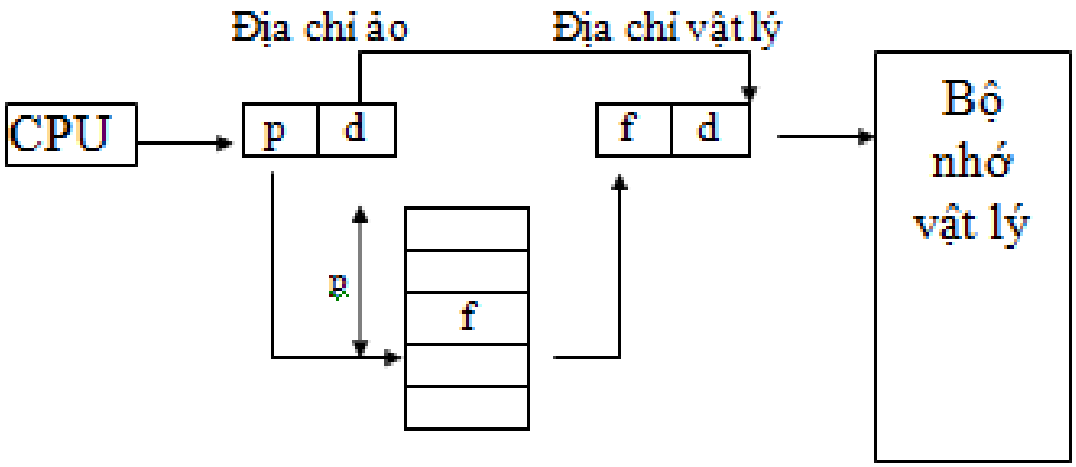
Cấu trúc địa chỉ ảo

- Kích thước của trang là lũy thừa của 2^n (9<=n<= 13)
- Kích thước của không gian địa chỉ ảo là 2^m (CPU dùng địa chỉ ảo m bít)
  - m-n bit cao của địa chỉ ảo sẽ biểu diễn số hiệu trang, và n bit thấp biểu diễn địa chỉ tương đối trong trang

địa chỉ ảo dạng (p,d) có m bit	
p là số hiệu trang và chiếm m-n bit cao	d là địa chỉ tương đối trong trang và chiếm n bít thấp

Cơ chế MMU trong mô hình phân trang

Địa chỉ vật lý = vị trí bắt đầu của khung trang f + d

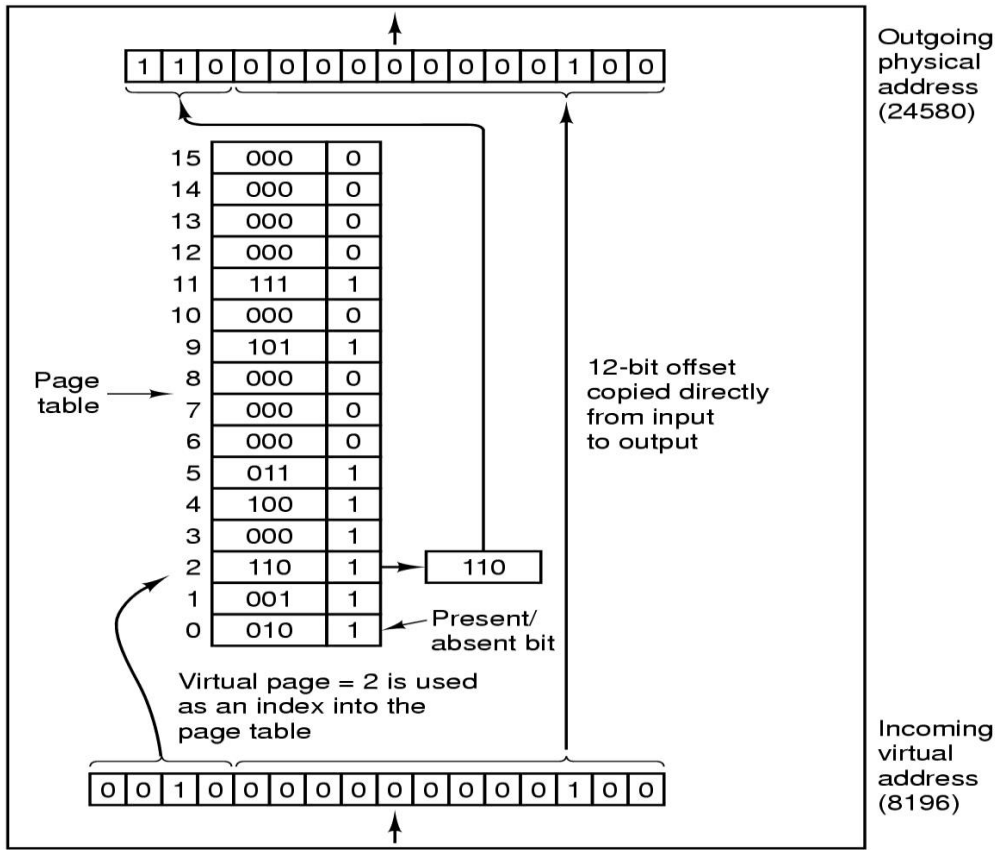


Giả sử tiến trình truy xuất địa chỉ ảo (p,d) = (3,500)
-> địa chỉ vật lý là 2048+500 = 2548.

Cơ chế chuyển đổi địa chỉ của MMU

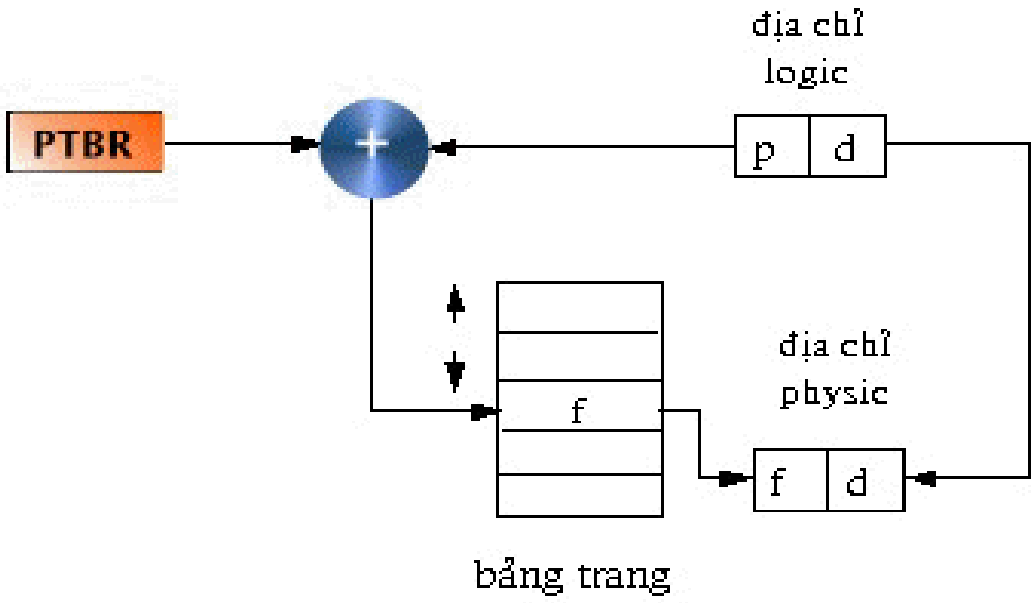
- Cách tính địa chỉ vật lý của MMU:
  - chép d vào n bit thấp của địa chỉ vật lý.
  - chép f vào (m-n) bit cao của địa chỉ vật lý

Ví dụ: Một hệ thống có địa chỉ ảo 16 bit dạng (p,d) với p có 4 bít, d có 12 bít (hệ thống có 16 trang, mỗi trang 4 KB) . Bít Present/absent =1 nghĩa là trang hiện ở trong bộ nhớ và =0 là ở bộ nhớ phụ.

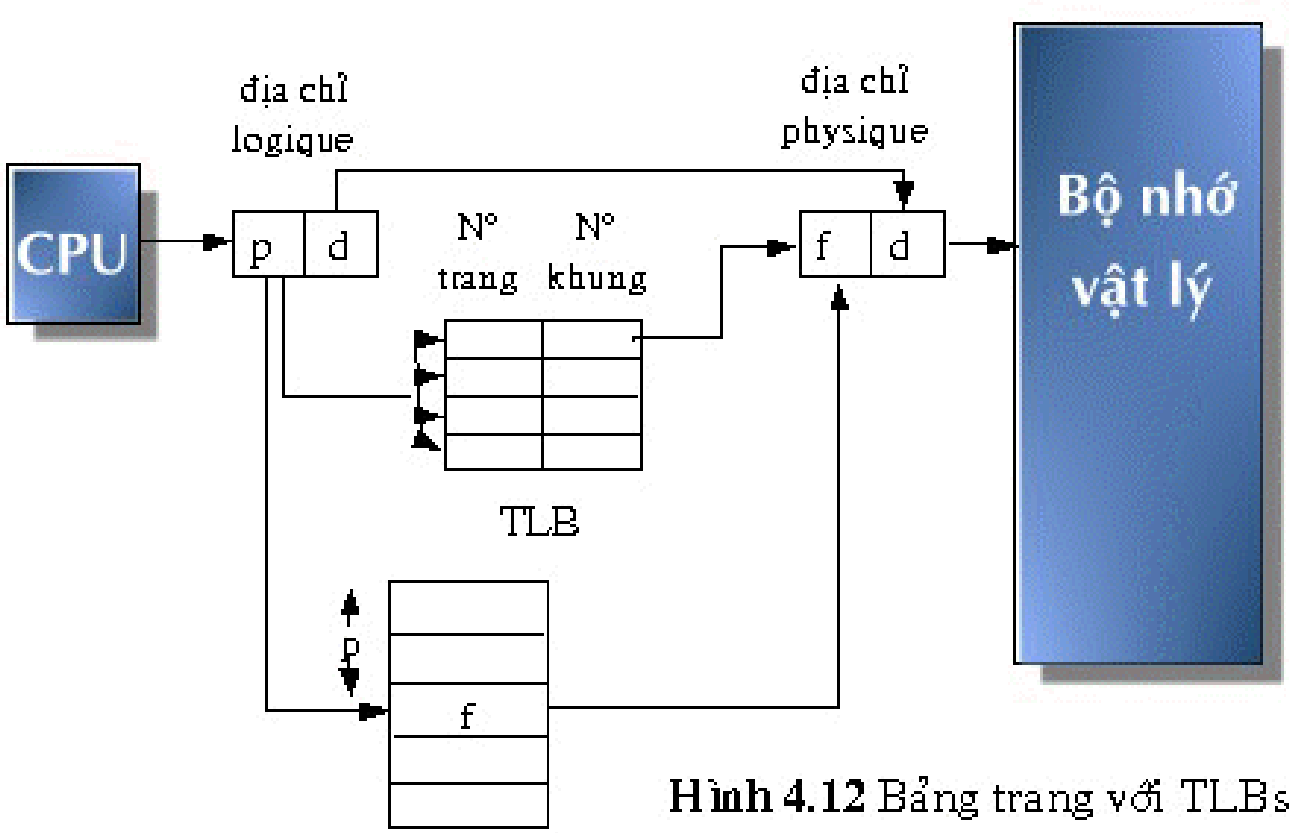


Cài đặt bảng trang

- Thanh ghi PTBR (Page Table Base Register): lưu địa chỉ bắt đầu của bảng trang.
- Thanh ghi PTLR (Page Table Limit Register): lưu số phần tử trong bảng trang.



Bộ nhớ kết hợp (Translation Lookaside Buffers - TLBs)



Hình 4.12 Bảng trang với TLBs

Mô hình phân trang – Ví dụ

- Một hệ thống máy tính 32 bit:
  - kích thước 1 khung trang là 4K.
  - Hỏi hệ thống quản lý được tiến trình kích thước tối đa là bao nhiêu?

- Máy tính 32 bit => địa chỉ ảo (p,d) có 32 bit => số bít của p + số bít của d = 32, mà 1 trang $4K=2^{12}$ bytes => d có 12 bit =>p có 20 bit => 1 bảng trang có 2^{20} phần tử.
- => Hệ thống quản lý được tiến trình có tối đa 2^{20} trang => kích thước tiến trình lớn nhất là $2^{20} \times 2^{12}$ byte = 2^{32} byte =4 GB.

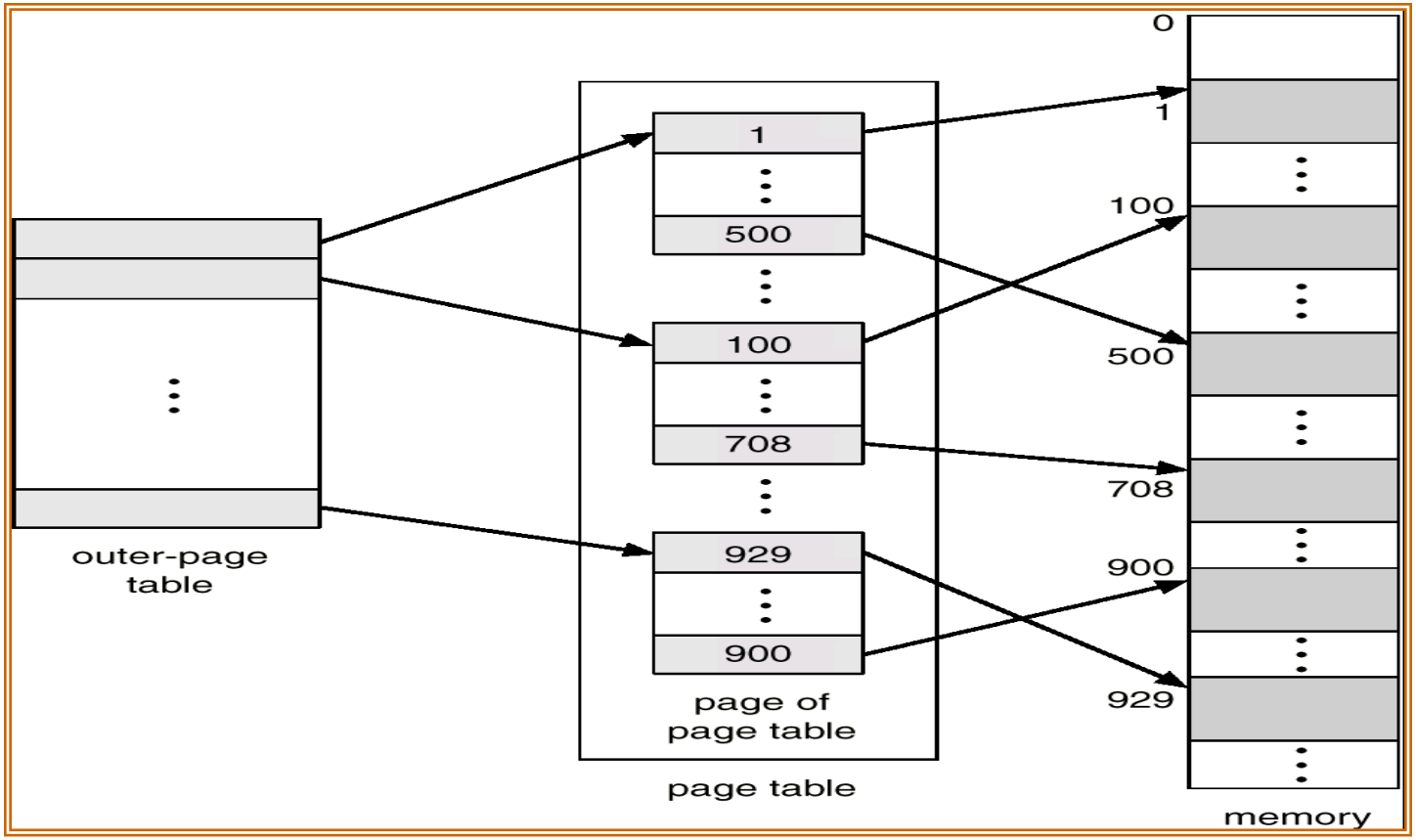
- Nhận xét: **Máy tính n bit quản lý được tiến trình kích thước lớn nhất là 2^n byte.**

Tổ chức bảng trang

- Phân trang đa cấp.
- Bảng trang băm.
- Bảng trang nghịch đảo

Để giải quyết bài toán tốn bộ nhớ khi tiến trình cần bảng trang lớn

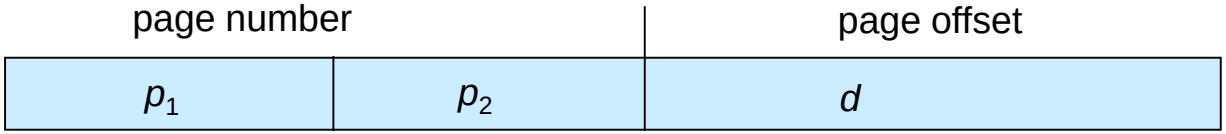
Phân trang đa cấp



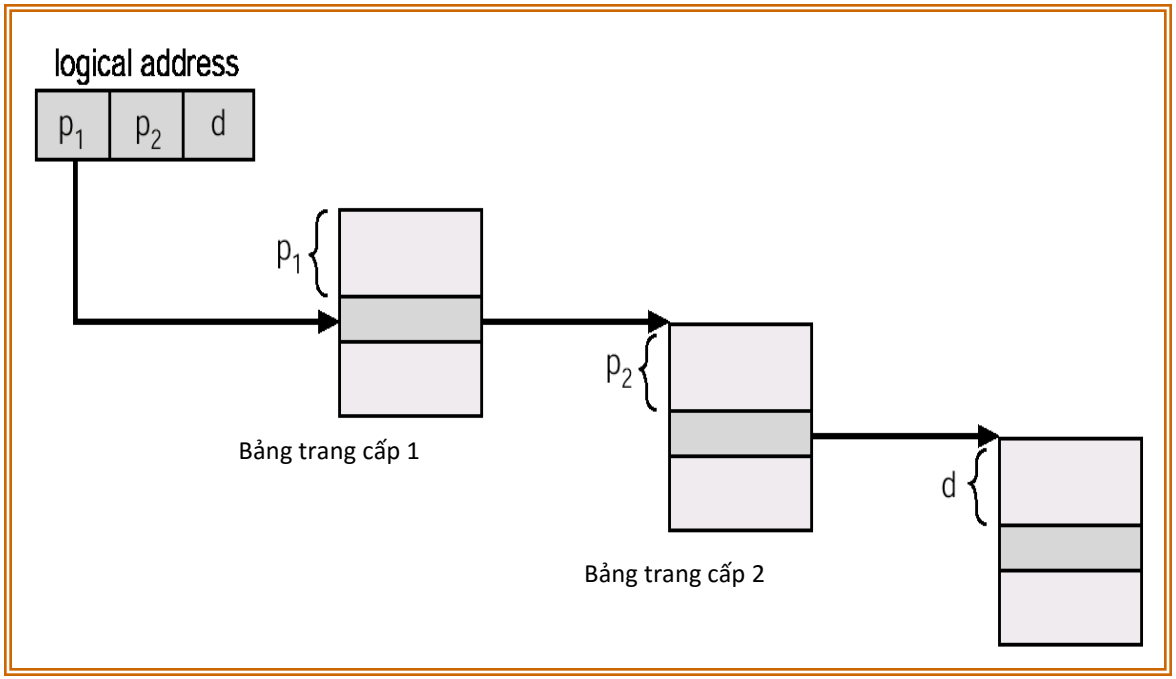
Bảng trang cấp 1

Các bảng trang cấp 2

Phân trang đa cấp



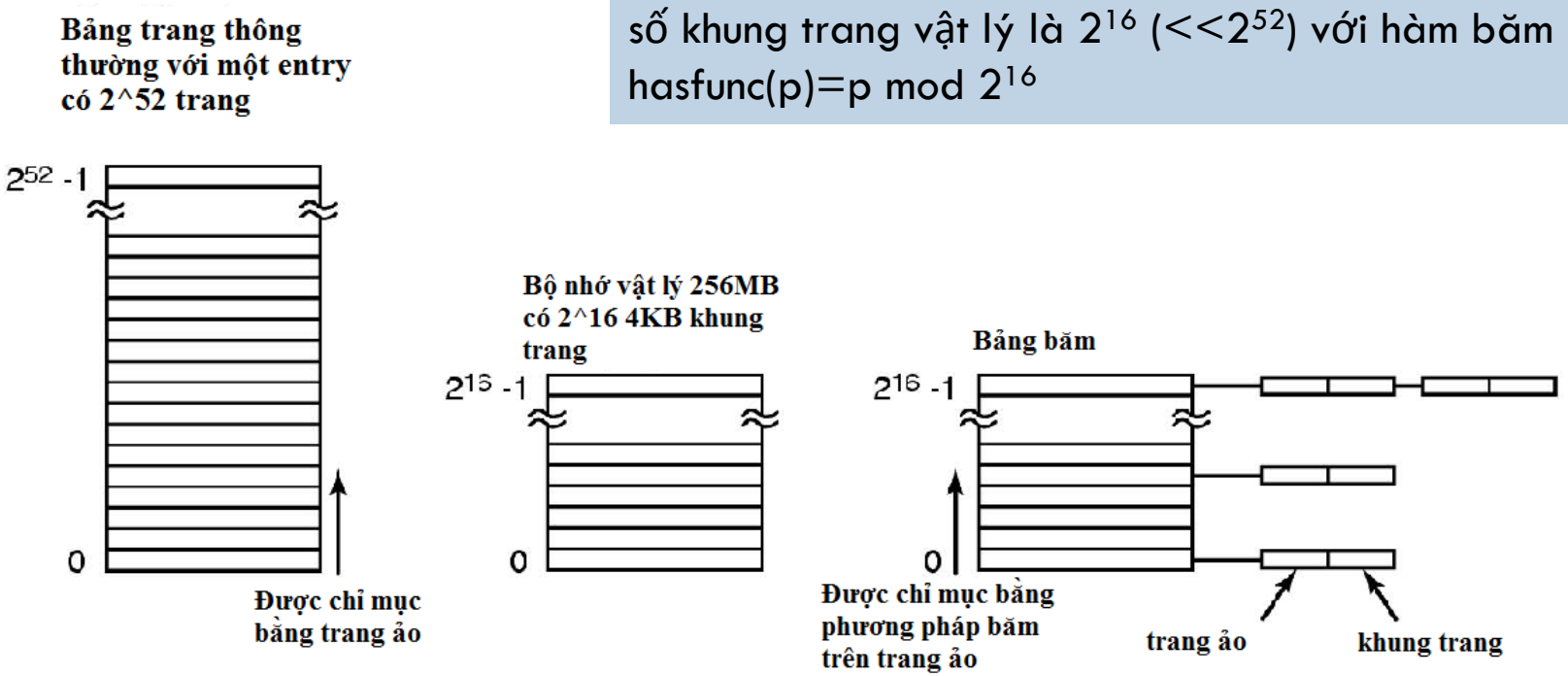
p1 chỉ mục của bảng trang cấp một. p2 chỉ mục của bảng trang cấp 2



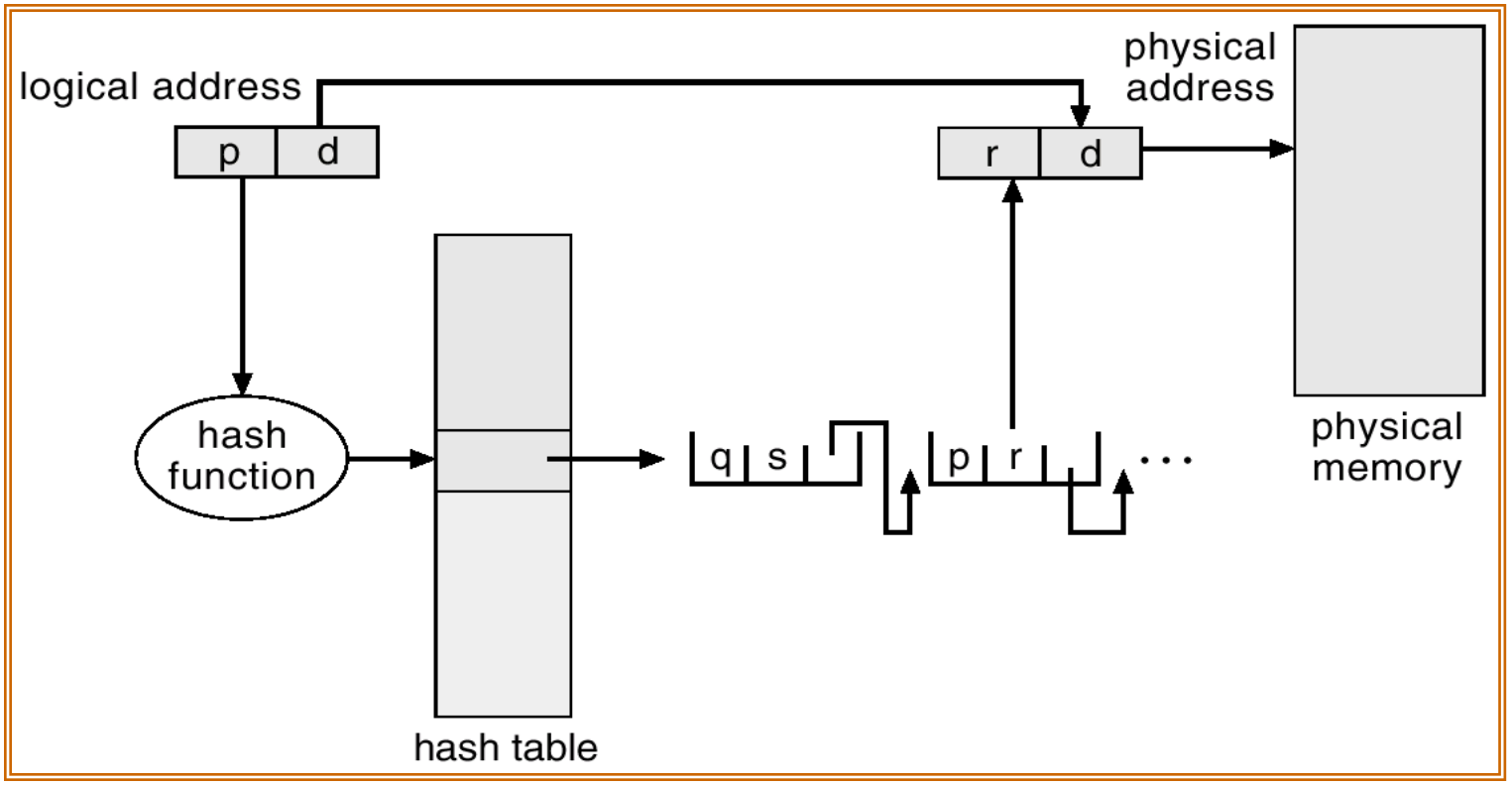
Bảng trang băm

- Khi không gian địa chỉ ảo > 32 bít)

❑ Một máy tính 64 bít, có RAM 256MB, kích thước 1 khung trang là 4KB.
❑ Bảng trang thông thường phải có 2^{52} mục, nếu dùng bảng trang băm có thể sử dụng bảng có số mục bằng số khung trang vật lý là 2^{16} ($<<2^{52}$) với hàm băm là hasfunc(p)=p mod 2^{16}

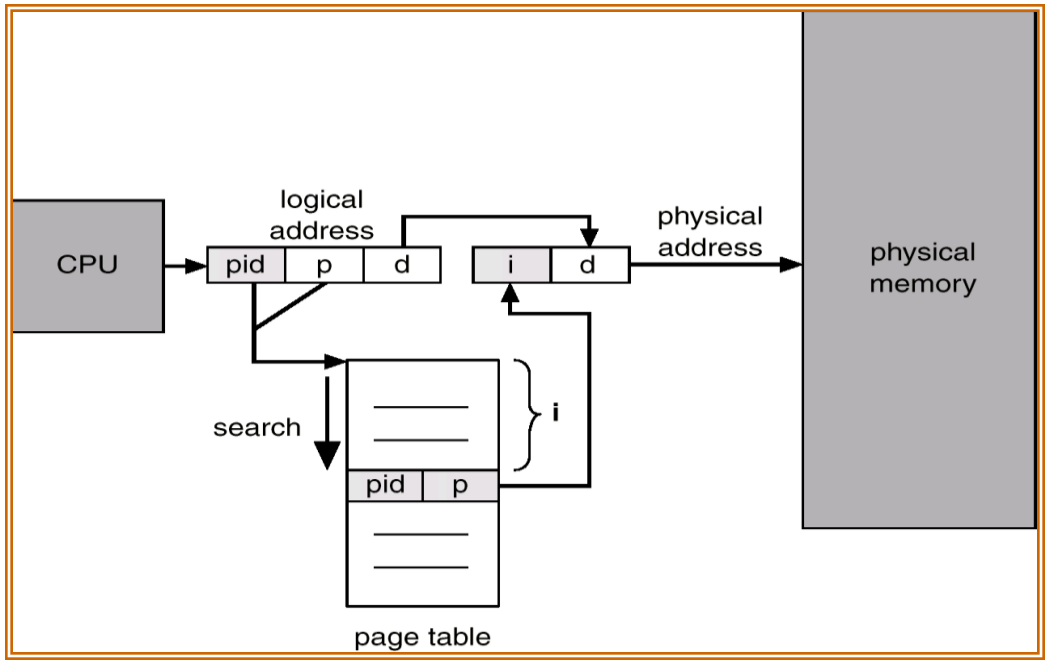


Cơ chế chuyển đổi địa chỉ khi sử dụng bảng trang băm

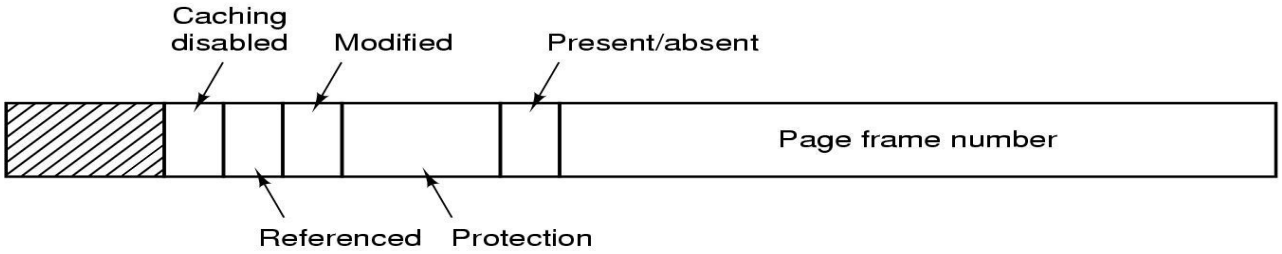


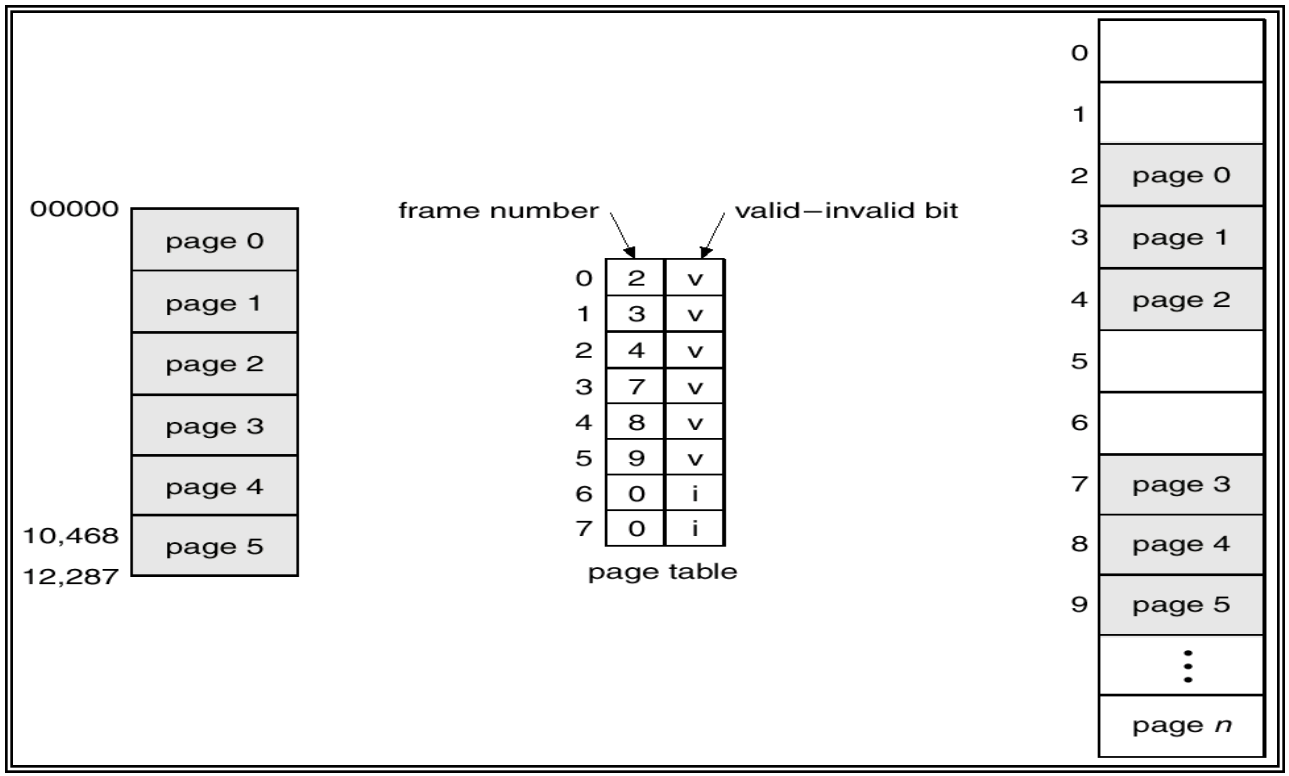
Bảng trang nghịch đảo

- Bảng trang duy nhất để quản lý bộ nhớ **của tất cả các tiến trình**.
- Mỗi phần tử của bảng trang nghịch đảo là cặp (pid, p)
  - pid là mã số của tiến trình
  - p là số hiệu trang.
- Mỗi địa chỉ ảo là một bộ ba (pid, p, d).



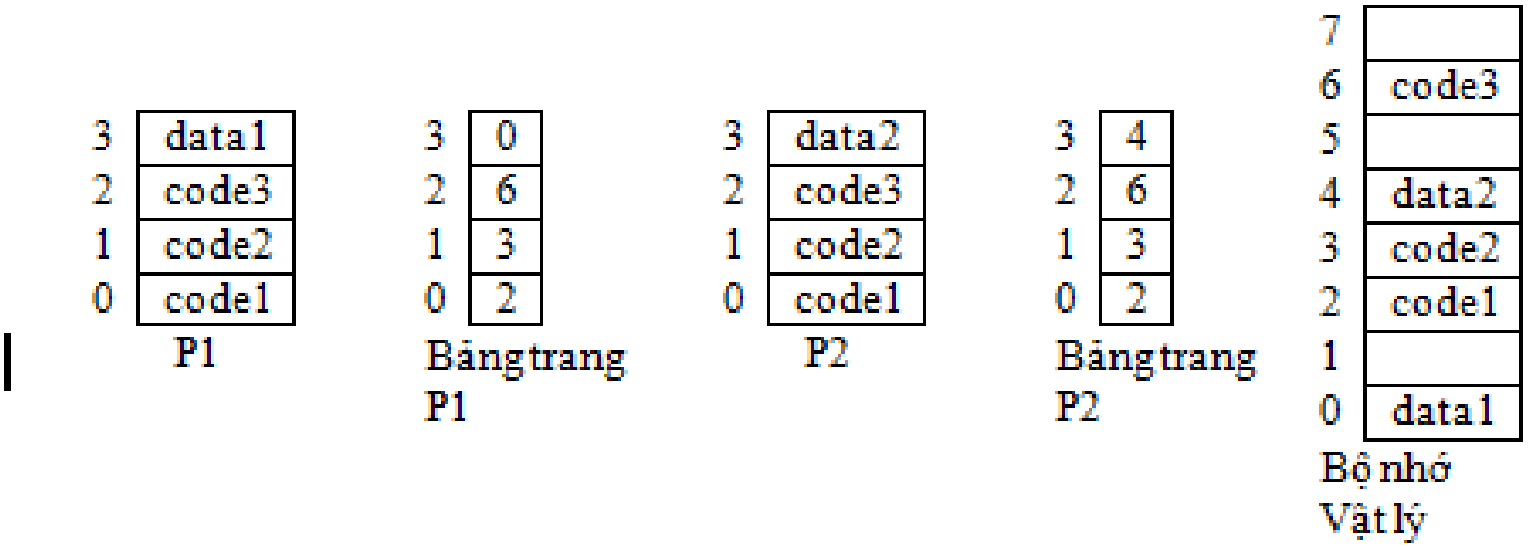
Bảo vệ trang


Caching disabled
Modified
Present/absent
Page frame number
Referenced
Protection


00000
page 0
page 1
page 2
page 3
page 4
10,468
page 5
12,287
frame number
valid–invalid bit
0 2 v
1 3 v
2 4 v
3 7 v
4 8 v
5 9 v
6 0 i
7 0 i
page table
0
1
2 page 0
3 page 1
4 page 2
5
6
7 page 3
8 page 4
9 page 5
page n

Chia xẻ bộ nhớ

- Các tiến trình dùng chung một số khung trang
  - ghi cùng số hiệu khung trang vào bảng trang của mỗi tiến trình



Hình 4.30: hai tiến trình P1,P2 dùng chung ba trang 0,1,2

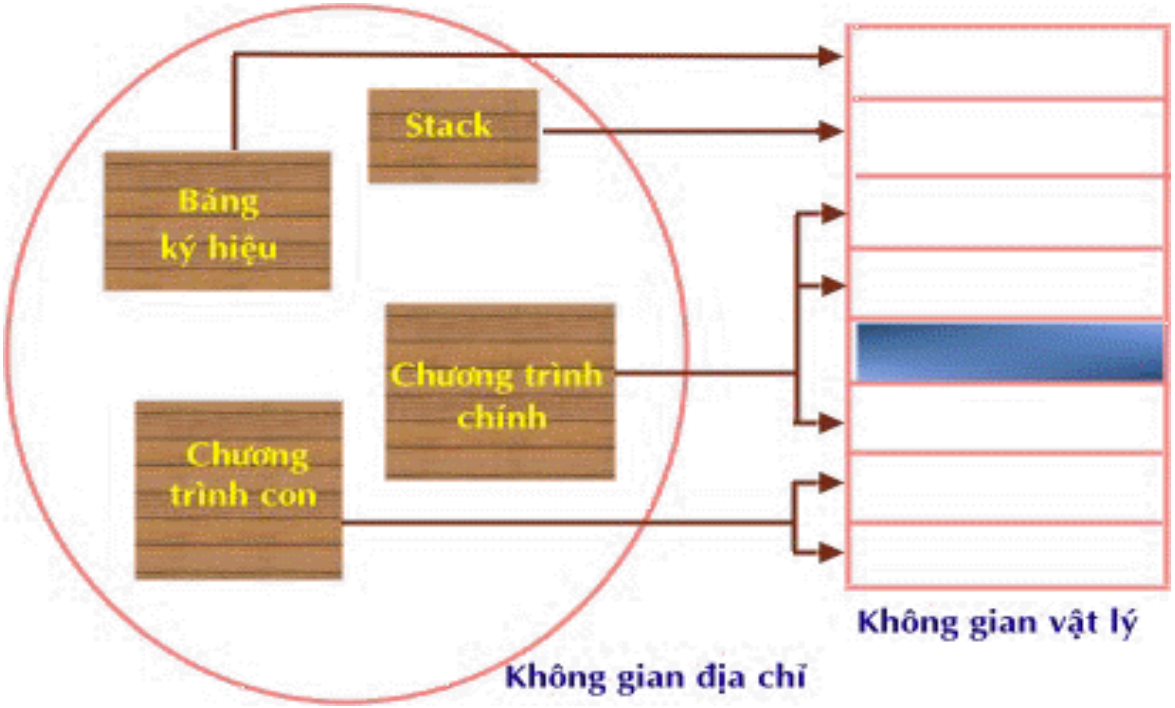
Mô hình phân trang – Nhận xét

- Loại bỏ được hiện tượng phân mảnh ngoại vi.
- Vẫn có hiện tượng phân mảnh nội vi.

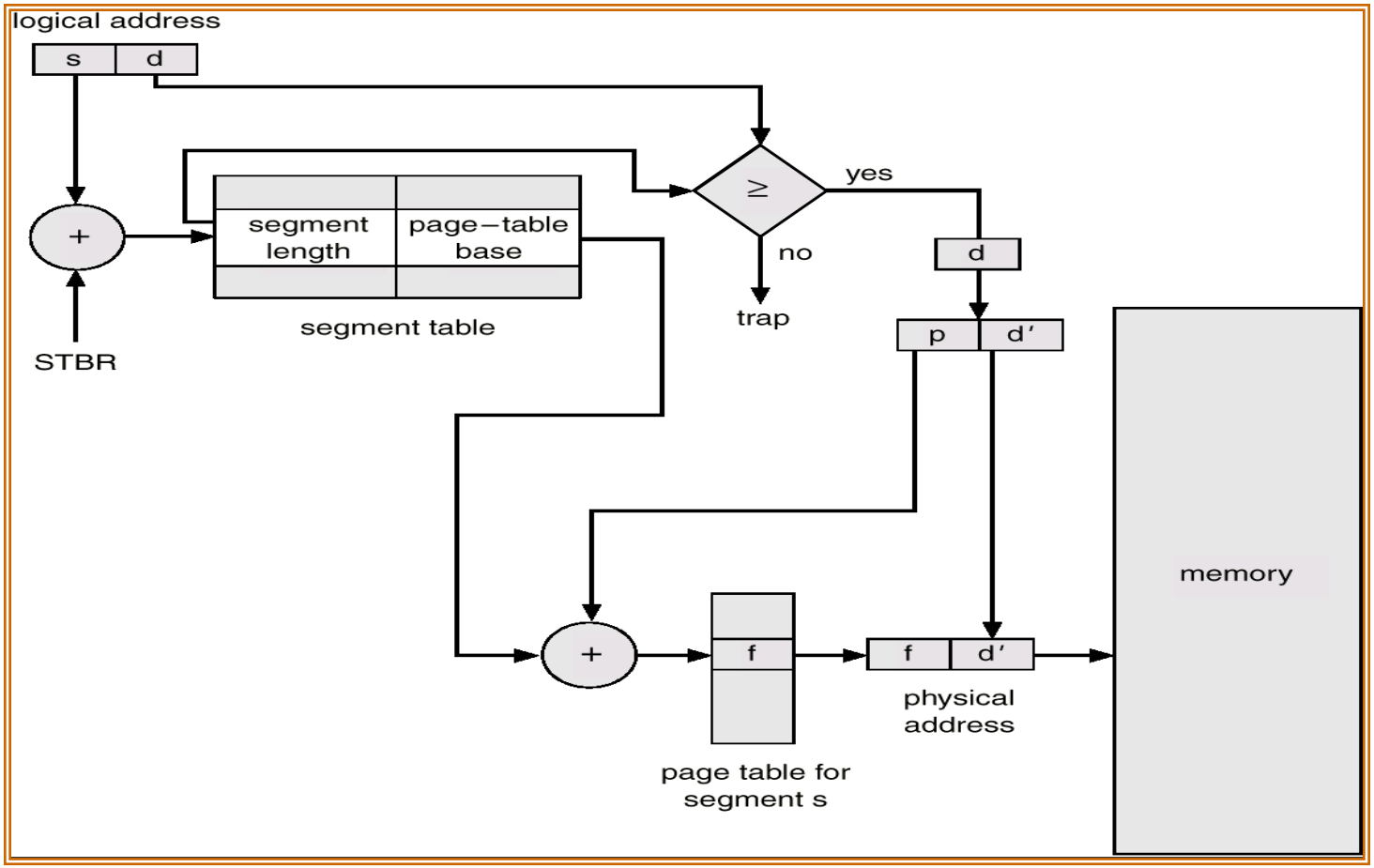
- Kết hợp cả hai kỹ thuật phân trang và phân đoạn:
  - phân trang các phân đoạn.

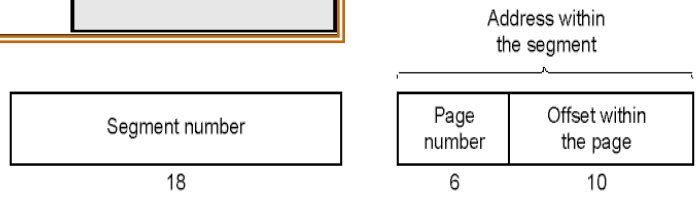
Mô hình phân đoạn kết hợp phân trang (Paged segmentation)

- Một tiến trình gồm nhiều phân đoạn.
- Mỗi phân đoạn được chia thành nhiều trang, lưu trữ vào các khung trang có thể không liên tục.



Cơ chế MMU trong mô hình phân đoạn kết hợp phân trang

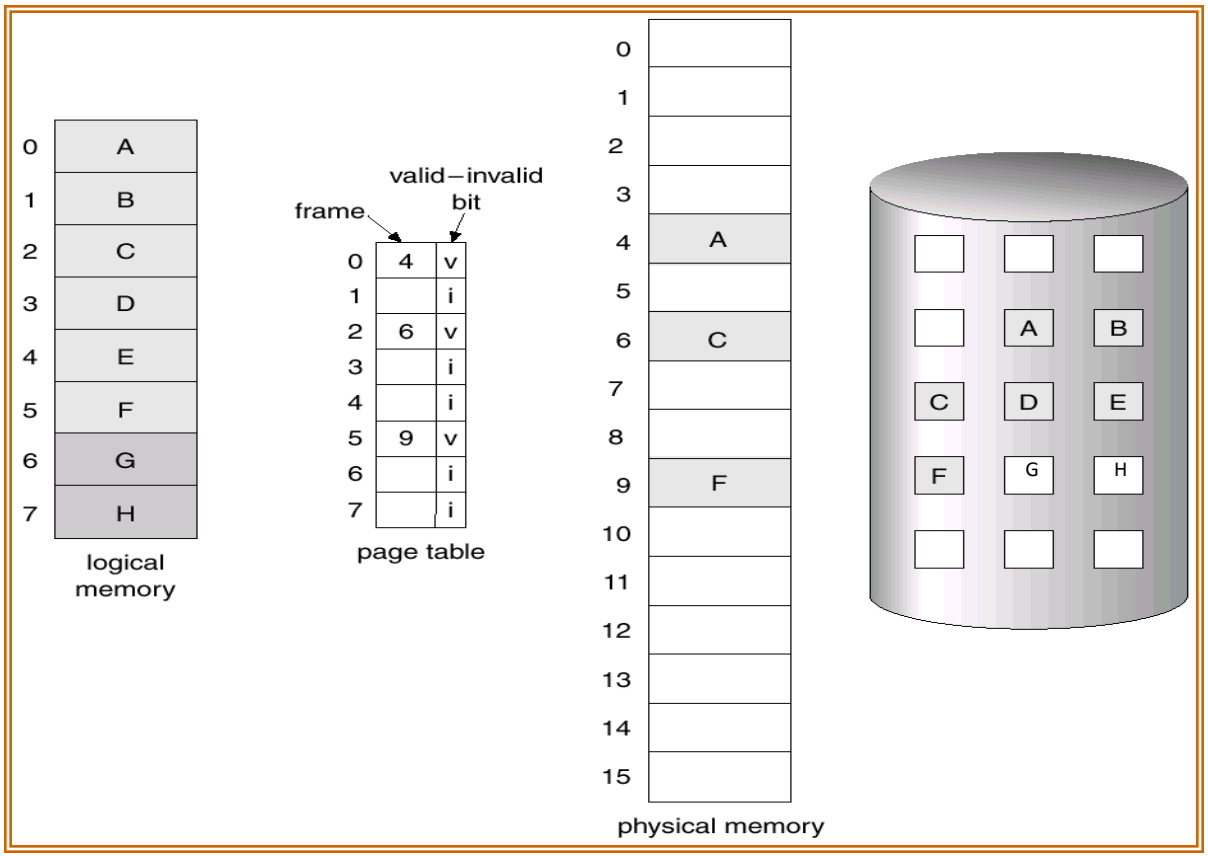




BỘ NHỚ ẢO

- Dùng bộ nhớ phụ lưu trữ tiến trình, các phần của tiến trình được chuyển vào-ra giữa bộ nhớ chính và bộ nhớ phụ.
- Phân trang theo yêu cầu (Demand paging).
- Phân đoạn theo yêu cầu (Demand segmentation)

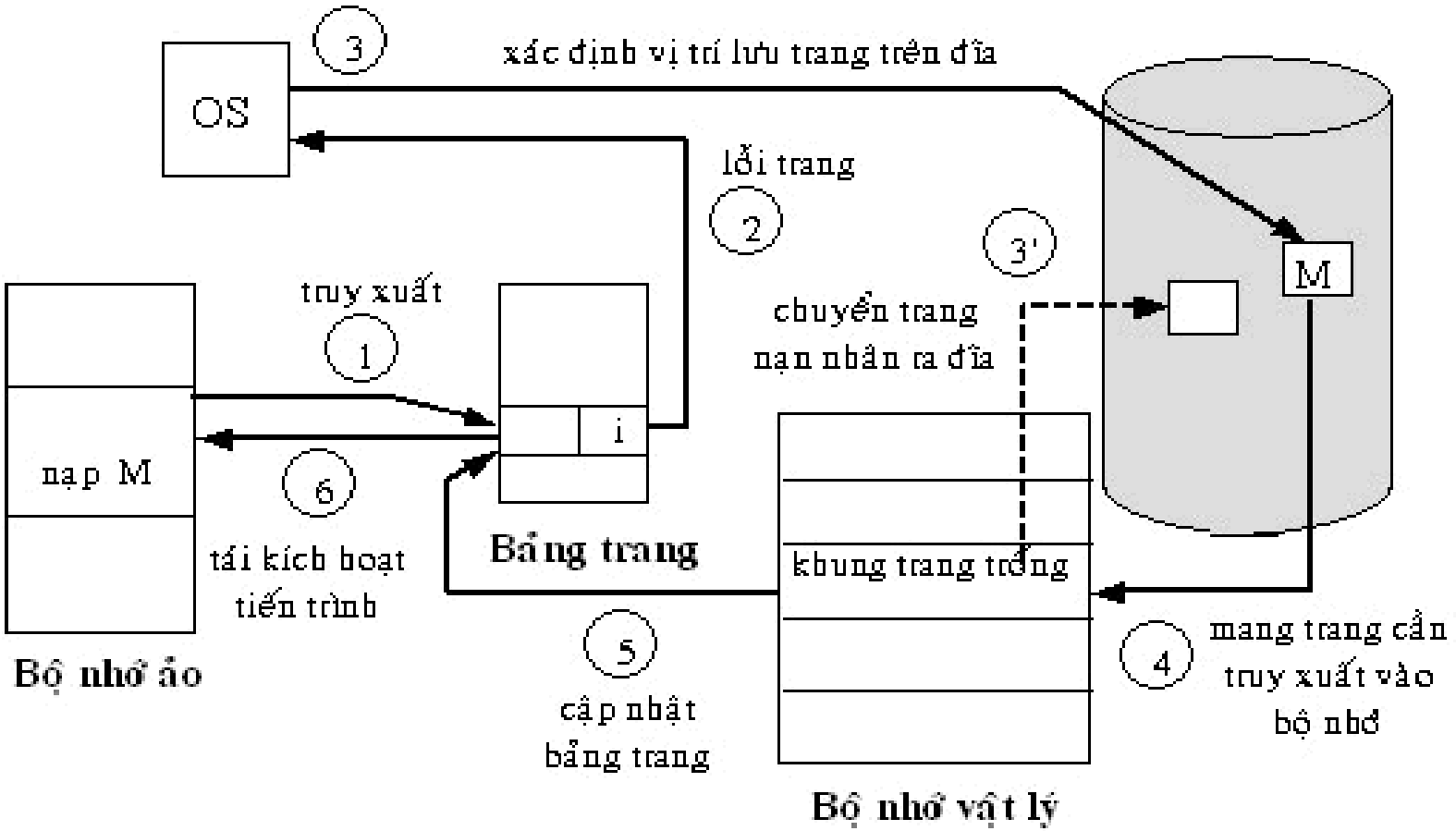
Phân trang theo yêu cầu (Demand paging)



Trường chứa bit "kiểm tra“:

- 1 (valid) là trang đang ở trong bộ nhớ chính
- 0 (invalid) là trang đang được lưu trên bộ nhớ phụ hoặc trang không thuộc tiến trình

Chuyển địa chỉ ảo (p,d) thành địa chỉ vật lý



Thay thế trang

- Bit "cập nhật" (dirty bit):
  - 1: nội dung trang có bị sửa đổi.
  - 0: nội dung trang không bị thay đổi.

số hiệu khung trang chứa trang hoặc địa chỉ trên đĩa của trang	bit nhận diện trang có trong bộ nhớ (bit valid-invalid)	bit nhận diện trang có thay đổi (bit dirty)

Thời gian thực hiện một yêu cầu truy xuất bộ nhớ

- P: xác suất xảy ra lỗi trang ($0 \leq p \leq 1$).
- Memory access (ma): thời gian một lần truy xuất bộ nhớ.
- Effective Access Time (EAT): thời gian thực hiện một yêu cầu truy xuất bộ nhớ.
- Page fault overhead (pfo) :thời gian xử lý một lỗi trang.
- Swap page in (spi): thời gian chuyển trang từ đĩa vào bộ nhớ.
- Swap page out (spo): thời gian chuyển trang ra đĩa (swap page out có thể bằng 0).
- Restart overhead (ro): thời gian tái khởi động lại việc truy xuất bộ nhớ.
- **EAT = (1 – p) x ma+ p (pfo+ [spo]+ spi+ ro)**

- Ví dụ: Thời gian một lần truy xuất bộ nhớ là 1 micro second và giả sử 40% trang được chọn đã thay đổi nội dung và thời gian hoán chuyển trang ra/vào là 10 mili second . Tính ETA.

- EAT = (1 – p) + p (pfo+10000*0.4+10000+ro) micro second

Các thuật toán chọn trang nạn nhân

- Trang «nạn nhân»: trang mà sau khi thay thế sẽ gây ra ít lỗi trang nhất.

- ***Thuật toán FIFO (First In First Out)***
- ***Thuật toán tối ưu (Optimal Page Replacement Algorithm)***
- ***Thuật toán LRU (Least-recently-used)***
- **Các thuật toán xấp xỉ LRU**
  - ***Thuật toán với các bít history***
  - ***Thuật toán cơ hội thứ hai***
  - ***Thuật toán cơ hội thứ hai nâng cao (Not Recently Used Page Replacement Algorithm: NRU)***
- **Các thuật toán thống kê**
  - Thuật toán LFU (least frequently used)
  - Thuật toán MFU (most frequently used)

Thuật toán FIFO (First In First Out)

- Trang ở trong bộ nhớ lâu nhất sẽ được chọn làm trang nạn nhân

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2	2	2	2	4	4	4	0	0	0	0	0	0	0	7	7	7
	0	0	0	0	3	3	3	2	2	2	2	2	1	1	1	1	1	0	0
		1	1	1	1	0	0	0	3	3	3	3	3	2	2	2	2	2	1
*	*	*	*		*	*	*	*	*	*			*	*			*	*	*

Kí hiệu * là có lỗi trang và có 15 lỗi trang

Nghịch lý Belady

- Xét tiến trình truy xuất chuỗi trang theo thứ tự sau: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

Nếu sử dụng 3 khung trang, sẽ có 9 lỗi trang

1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	4	4	4	5	5	5	5	5	5
	2	2	2	1	1	1	1	1	3	3	3
		3	3	3	2	2	2	2	2	4	4
*	*	*	*	*	*	*			*	*	

Nếu sử dụng 4 khung trang, sẽ có 10 lỗi trang

1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	1	1	1	5	5	5	5	4	4
	2	2	2	2	2	2	1	1	1	1	5
		3	3	3	3	3	3	2	2	2	2
			4	4	4	4	4	4	3	3	3
*	*	*	*			*	*	*	*	*	*

Thuật toán tối ưu (Optimal Page Replacement Algorithm)

- Chọn trang lâu được sử dụng nhất trong tương lai.

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2	2	2	2	2	2	2	2	2	2	2	2	2	2	7	7	7
	0	0	0	0	0	0	4	4	4	0	0	0	0	0	0	0	0	0	0
		1	1	1	3	3	3	3	3	3	3	3	1	1	1	1	1	1	1
*	*	*	*		*		*			*			*				*		

- Số lượng lỗi trang phát sinh là thấp nhất.
- Không bị nghịch lý Belady.
- Khó cài đặt
- Phù hợp với hệ điều hành cho thiết bị gia dụng

Thuật toán LRU (Least-recently-used)

- Dựa vào thời điểm cuối cùng trang được truy xuất.
- trang được chọn để thay thế sẽ là trang lâu nhất chưa được truy xuất.

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2	2	2	2	4	4	4	0	0	0	1	1	1	1	1	1	1
	0	0	0	0	0	0	0	0	3	3	3	3	3	3	0	0	0	0	0
		1	1	1	3	3	3	2	2	2	2	2	2	2	2	2	7	7	7
*	*	*	*		*		*	*	*	*			*		*		*		

Cài đặt thuật toán LRU

- ***Sử dụng bộ đếm***
  - Cấu trúc phần tử trong bảng trang: thêm trường ghi nhận "thời điểm truy xuất gần nhất".
  - Cấu trúc của CPU: thêm một thanh ghi đếm (counter).

số hiệu khung trang chứa trang hoặc địa chỉ trang trên đĩa	bit valid - invalid	bit dirty	thời điểm truy xuất gần nhất

- ***Sử dụng danh sách liên kết***
  - Trang ở cuối danh sách là trang được truy xuất gần nhất
  - Trang ở đầu danh sách là trang lâu nhất chưa được sử dụng

Các thuật toán xấp xỉ LRU

- Mỗi phần tử trong bảng trang có thêm bit reference:
  - được khởi gán là 0 bởi hđh.
  - được phần cứng gán là 1 mỗi lần trang tương ứng được truy cập

số hiệu khung trang chứa trang hoặc địa chỉ trang trên đĩa	bit valid-invalid	bit dirty	bit reference

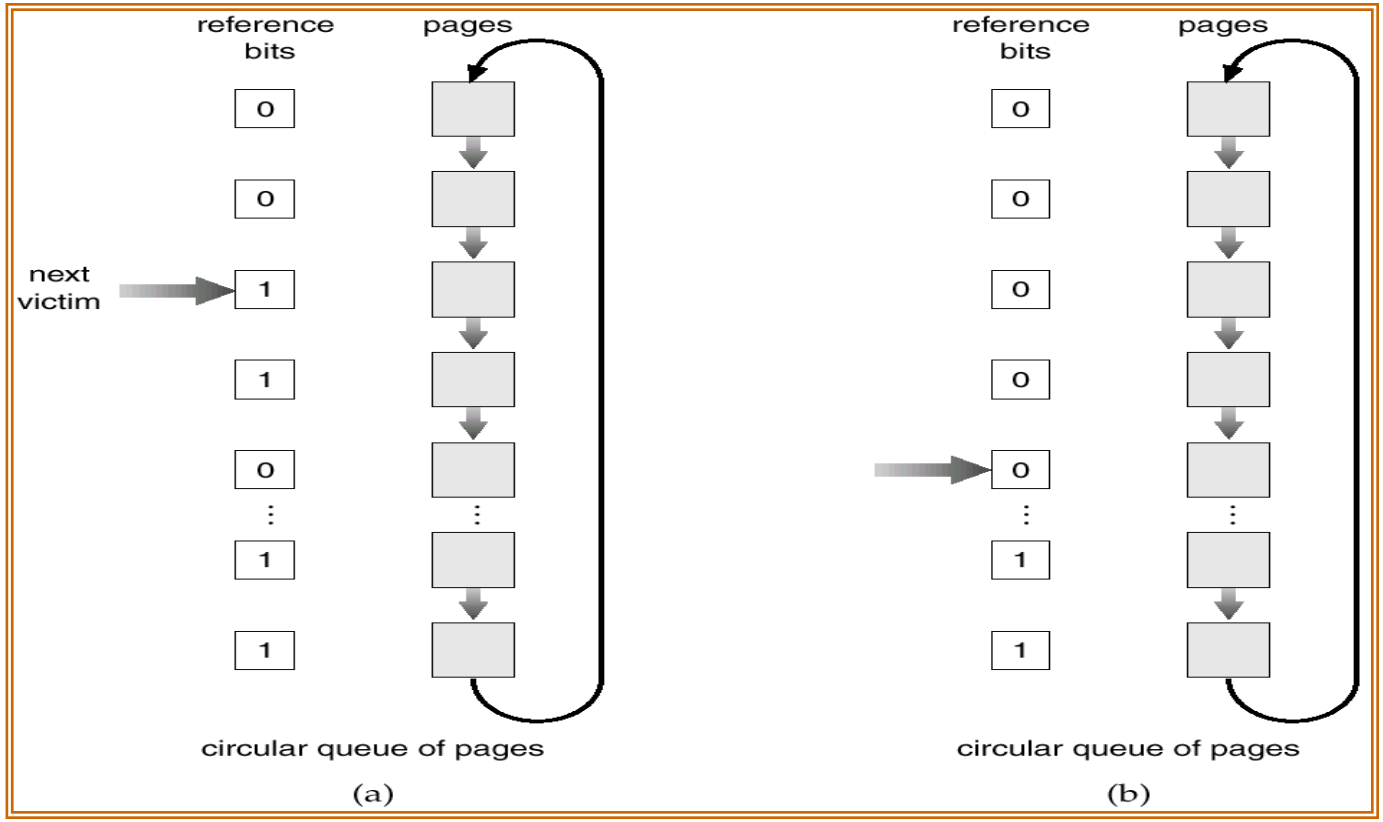
- Các thuật toán xấp xỉ LRU:
  - ***Thuật toán với các bít history.***
  - ***Thuật toán cơ hội thứ hai.***
  - ***Thuật toán cơ hội thứ hai nâng cao (Not Recently Used Page Replacement Algorithm: NRU)***

Thuật toán với các bít history

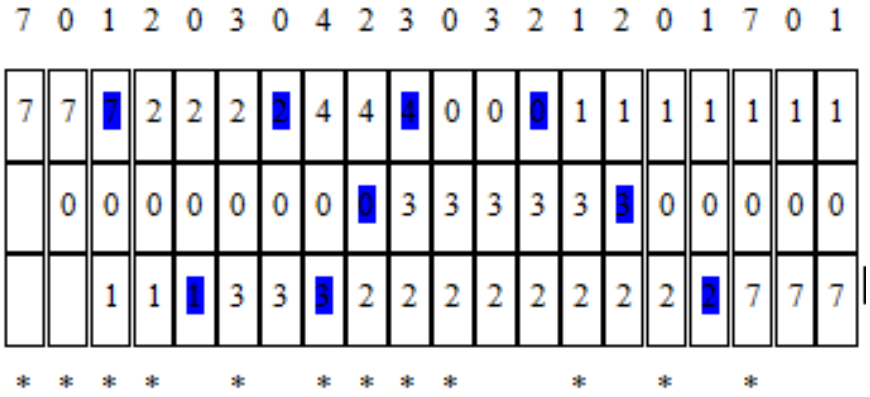
- Mỗi trang sử dụng thêm 8 bit lịch sử (history).
- Cập nhật các bít history:
  - dịch các bit history sang phải 1 vị trí để loại bỏ bit thấp nhất.
  - đặt bit reference của mỗi trang vào bit cao nhất trong 8 bit history của trang đó.
- 8 bit history sẽ lưu trữ tình hình truy xuất đến trang trong 8 chu kỳ cuối cùng.
- Trang "nạn nhân" là trang có giá trị history nhỏ nhất.

Thuật toán cơ hội thứ hai

- Tìm một trang theo nguyên tắc FIFO.
- Kiểm tra bit reference của trang đó.
  - Nếu bit reference là 0, chọn trang này.
  - Nếu bit reference là 1 thì gán lại là 0 rồi tìm trang FIFO tiếp theo



Thuật toán cơ hội thứ hai - Ví dụ



7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7(1)	7(1)	7(1)	0(0)	1(0)	2(1)	2(1)	3(0)	3(0)	4(1)	2(0)	2(0)	0(1)	3(0)	3(0)	1(0)	2(1)	0(0)	1(0)	7(1)
	0(1)	0(1)	1(0)	2(1)	0(1)	3(1)	0(0)	4(1)	2(1)	3(0)	0(1)	3(1)	2(0)	1(0)	2(1)	0(1)	1(0)	7(1)	0(1)
		1(1)	2(1)	0(1)	3(1)	0(1)	4(1)	2(1)	3(1)	0(1)	3(1)	2(1)	1(0)	2(1)	0(1)	1(1)	7(1)	0(1)	1(1)

Thuật toán cơ hội thứ hai nâng cao (Not Recently Used Page Replacement Algorithm: NRU)

- Lớp 1 (0,0): gồm những trang có (ref,dirty)=(0,0). (độ ưu tiên thấp nhất)
  - không được truy xuất gần đây và không bị sửa đổi
  - tốt nhất để thay thế.
- Lớp 2 (0,1):
  - không truy xuất gần đây nhưng đã bị sửa đổi.
  - Trường hợp này không thật tốt, vì trang cần được lưu trữ lại trước khi thay thế.
- Lớp 3 (1,0):
  - được truy xuất gần đây, nhưng không bị sửa đổi.
  - Trang có thể nhanh chóng được tiếp tục được sử dụng.
- Lớp 4 (1,1): (độ ưu tiên cao nhất)
  - trang được truy xuất gần đây, và bị sửa đổi.
  - Trang có thể nhanh chóng được tiếp tục được sử dụng và trước khi thay thế cần phải được lưu trữ lại.
- Trang "nạn nhân": trang đầu tiên tìm thấy trong lớp có độ ưu tiên thấp nhất.

Các thuật toán thống kê

- Biến đếm: lưu số lần truy xuất đến một trang.
- Thuật toán LFU (least frequently used):
  - Thay thế trang có giá trị biến đếm nhỏ nhất, nghĩa là trang ít được sử dụng nhất.
- Thuật toán MFU (most frequently used):
  - Thay thế trang có giá trị biến đếm lớn nhất, nghĩa là trang được sử dụng nhiều nhất.

Cấp phát số lượng khung trang

- ***Cấp phát ngang bằng***
  - m khung trang và n tiến trình.
  - Mỗi tiến trình được cấp m/n khung trang.
- ***Cấp phát theo tỷ lệ kích thước***
  - s_i :kích thước của tiến trình p_i
  - $S = \sum s_i$ là tổng kích thước của tất cả tiến trình
  - m : số lượng khung trang có thể sử dụng
  - a_i : số khung trang được cấp phát cho tiến trình p_i

    $$ai = si / S \times m$$

  - Ví dụ: Tiến trình 1= 10K, tiến trình 2=127K và có 62 khung trang trống. Khi đó có thể cấp cho
    - tiến trình 1: 10/137 x 62 ~ 4 khung
    - tiến trình 2: 127/137 x62 ~ 57 khung
- ***Cấp phát theo tỷ lệ độ ưu tiên:***

Thay thế trang

- ***Thay thế toàn cục***
  - Chọn trang "nạn nhân" từ tập tất cả các khung trang trong hệ thống.
  - Có nhiều khả năng lựa chọn hơn.
  - Số khung trang cấp cho một tiến trình có thể thay đổi.
  - Các tiến trình không thể kiểm soát được tỷ lệ phát sinh lỗi trang của mình.
- ***Thay thế cục bộ***
  - Chỉ chọn trang thay thế trong tập các khung trang được cấp cho tiến trình phát sinh lỗi trang.
  - Số khung trang cấp cho một tiến trình sẽ không thay đổi

Hệ thống trì trệ (thrashing)

- Không có đủ các khung trang -> thường xuyên phát sinh các lỗi trang-> nhiều thời gian sử dụng CPU để thực hiện thay thế trang.
- **Mô hình tập làm việc (working set).**

Mô hình tập làm việc (working set)

- WSSi (Δ , t): số phần tử của tập working set của tiến trình Pi tại thời điểm t.
  - tập các trang được tiến trình truy xuất đến trong Δ lần truy cập cuối cùng tính tại thời điểm t.
- m :số khung trang trống.
- D = $\sum$WSSi : tổng số khung trang yêu cầu cho toàn hệ thống.
- Tại thời điểm t: cấp cho Pi số khung trang bằng (WSSi)(Δ, t-1).
- D>m -> Trì trệ hệ thống