9.4)

Ta có: Logical Address (LA) = Page Number + Page Offset

Physical Address (PA) = Frame Number + Frame Offset

Vì tương ứng cho mỗi trang trên địa chỉ ảo (page) sẽ có một trang thật trên bộ nhớ vật lý (frame) cùng kích thước và một trang có 1024 từ = 2^{10} từ nên ⇒ Page Offset (cần 10 bits để lưu trữ) = Frame Offset = 10 bits

Có 64 pages = 2^6 pages ⇒ Cần tối thiểu 6 bits để biểu diễn ⇒ Page Number = 6

Có 32 frames = 2^5 frames ⇒ Cần tối thiểu 5 bits để biểu diễn ⇒ Frame Number = 5

Vậy a) Logical Address = 6 + 10 = 16 bits

b) Physical Address = 5 + 10 = 15 bits

9.6) Theo giả thiết, bộ nhớ được chia thành 6 phần có kích thước khác nhau và theo thứ tự như sau:

Phần	1	2	3	4	5	6
Kích thước	300KB	600 KB	350 KB	200 KB	750 KB	125 KB

Có các tiến trình với kích thước (theo thứ tự) như sau:

Tiến trình	a	b	c	d	e
Kích thước	115 KB	500 KB	358 KB	200 KB	375 KB

Cách hoạt động của thuật toán:

- First-fit: (g+) sách (169)

+ Phần 1 (300KB) sẽ cấp phát bộ nhớ cho tiến trình a (115 KB) nên phần 1 còn lại 300 KB − 115 KB = 185 KB

+ Vì phần 1 chỉ có 185 KB bộ nhớ còn trống nên không đủ cấp phát cho tiến trình b (500 KB), thuật toán sẽ duyệt tiếp và phần 2 (600 KB) đủ bộ nhớ để cấp phát nên phần 2 còn lại 600 KB − 500 KB = 100 KB

+ Tiếp tục như vậy ta có kết quả theo bảng sau:

Tiến trình	Phần bộ nhớ cấp phát	Phần bộ nhớ còn lại
a (115 KB)	1 (300 KB)	300KB − 115KB = 185KB
b (500 KB)	2 (600 KB)	600KB − 500KB = 100KB
c (358 KB)	5 (750KB)	750KB − 358 KB = 392 KB
d (200 KB)	3 (350 KB)	350 KB − 200KB = 150 KB
e (375 KB)	5 (392KB)	392KB − 375KB = 17 KB

+ Best-fit (gt)

- Trong bài này thuật toán sẽ phải duyệt qua hết tất cả các phần của bộ nhớ và thấy rằng ở phần 6 (125 KB) là phần có kích thước nhỏ nhất vẫn chứa được tiến trình a (115 KB) nên phần 6 sẽ cấp phát bộ nhớ cho tiến trình a và còn lại

125 KB - 115 KB = 10 KB

- Tiếp tục . . . . .

Tiến trình	Phần bộ nhớ cấp phát	Phần bộ nhớ còn lại
a (115 KB)	6 (125 KB)	- - - = 10 KB
b (500 KB)	2 (600 KB)	. . . = 100 KB
c (358 KB)	5 (750 KB)	- - = 392 KB
d (200 KB)	4 (200 KB)	- - - = 0 KB
e (375 KB)	5 (392 KB)	. - - = 17 KB

+ Worst-fit (gt)

- Thuật toán cũng sẽ phải duyệt qua hết tất cả các phần của bộ nhớ và chọn ra phần có kích thước lớn nhất để cấp phát bộ nhớ. ~~3 đoạn~~ Trong trường hợp hiện tại phần 5 (750 KB) sẽ cấp phát bộ nhớ cho tiến trình a và còn lại 750 KB - 115 KB = 635 KB

- Tiếp tục . . . . .

Tiến trình	Phần bộ nhớ cấp phát	Phần bộ nhớ còn lại
a (115 KB)	5 (750 KB)	- - - = 635 KB
b (500 KB)	5 (635 KB)	- - - = 135 KB
c (358 KB)	2 (600 KB)	. . . = 242 KB
d (200 KB)	3 (350 KB)	- - - = 150 KB
e (375 KB)		

- Vì k° còn phần bộ nhớ nào đủ lớn ~~đủ~~ để cấp phát cho tiến trình e (3..) nên tiến trình sẽ phải chờ

9.7)

Vì kích thước trang = 1KB = 1000B nên số bit tối thiểu cần để biểu diễn là:

$n \geq \log_2(1000) \approx 9,97 \Rightarrow n = 10$ bits = Page Offset

Các bước thực hiện:

- Chuyển đổi địa chỉ luận lý (Logical address) từ hệ thập phân sang hệ nhị phân
- Chia địa chỉ ở dạng thập phân thành 2 phần:
  + Page Offset (10 bits cuối)
  + Page Number các bits còn lại
- Chuyển đổi Page Number và Page Offset về lại hệ thập phân.

Dựa vào các bước trên ta có đáp án sau:

Câu	LA (thập phân)	LA (nhị phân) Page Number	LA (nhị phân) Page Offset	Page Number (TP)	Page Offset (TP)
a	3085		1100 0000 1101	3	13
b	42095		1010 0100 0110 1111	41	111
c	215201	0011	0100 1000 1010 0001	210	161
d	650000		1001 1110 1011 0001 0000	634	784
e	2000001	0001 1110	1000 0100 1000 0001	1953	129

9.25)

a) Vì thời gian một chu kì truy xuất bộ nhớ là 50 nanoseconds (ns) nên thời gian truy xuất bộ vật lý tham chiếu (bộ nhớ phân trang) sẽ cần tổng cộng 50 ns + 50 ns = 100 ns, 50 ns để truy cập đến bảng trang và 50 ns để truy cập đến bộ nhớ thực

b) Ta có:

+ Thời gian tìm kiếm trong TLB: $\varepsilon = 2$ ns
+ Thời gian một chu kì truy xuất bộ nhớ: $x = 50$ ns
+ Tỉ số giữa số lần chỉ số trang được tìm thấy trong TLB và số lần truy xuất khối ngoặc từ CPU: $\alpha = 75\%$

Vậy thời gian truy xuất hiệu dụng:

$$EAT = \alpha.(\varepsilon + x) + (1-\alpha)(\varepsilon + x + x)$$
$$= 75\%.(2+50) + (1-75\%)(2+50+50)$$
$$= 64,5 \text{ ns}$$

10.5)

Theo giả thiết ta có địa chỉ ảo có độ dài 12 bits, do đó sẽ có 2^{12} địa chỉ trong không gian địa chỉ ảo

Ngoài ra kích thước trang là 256 bytes (có $256 = 2^8$ địa chỉ trong một trang)

Do đó số lượng trang sẽ là $2^{12} / 2^8 = 2^4$

Nên cần 4 bits để biểu diễn tất cả các trang => PageNumber = 4

Mà Logical Address (LA) = PageNumber + Page Offset

12 = 4 + Page Offset

=> Page Offset = 8

Vậy 4 bits quan trọng nhất trong địa chỉ ảo sẽ biểu thị số trang và 8 bits còn lại sẽ là độ lệch trong

Tuy nhiên kích thước page (trong không gian địa chỉ ảo) luôn giống với kích thước frame trong bộ nhớ chính. Do đó Frame Offset = Page Offset = 8, 8 bits cuối cùng sẽ vẫn giữ nguyên trong địa chỉ vật lý như địa chỉ ảo.

Từ nhận định trên, nếu muốn lấy địa chỉ vật lý, ta chỉ cần xét 4 bits đầu tiên.

+ 9EF, xét Page = 9, tra bảng ta sẽ được Page Frame = 0

Vậy địa chỉ vật lý tương ứng là 0EF

+ 111, xét Page = 1, tra bảng ta sẽ được Page Frame = 2

Vậy địa chỉ vật lý tương ứng là 211

+ 700, xét Page = 7, Page Frame của Page = 7 chưa có trong bộ nhớ cho nên gây ra một lỗi trang. Hệ điều hành sẽ chọn một trang nạn nhân trên bộ nhớ chính, và hoán đổi với trang được yêu cầu đang ở bộ nhớ phụ. Khi đó bảng trang sẽ được cập nhật lại. Trong trường hợp này hệ điều hành sẽ xác định page frame có sẵn trong bộ nhớ vật lý hiện không được sử dụng (hiện tại là D) trở thành mục tiêu để tải trang được yêu cầu. Sau đó hệ điều hành khởi tạo thao tác I/O để đọc trang được yêu cầu từ đĩa cứng vào page frame trống trong bộ nhớ vật lý. Sau khi trang được yêu cầu tải vào page frame trống, hệ điều hành sẽ cập nhật lại bảng trang để phản ánh rằng trang hiện nằm trong bộ nhớ vật lý và đánh dấu page frame tương ứng là hợp lệ. Lúc này page frame trống còn lại E và F với E đứng đầu danh sách

Vậy địa chỉ ... D00

+ 0FF, xét Page = 0, Page Frame của Page = 0 cũng chưa có trong bộ nhớ và gây ra một lỗi trang. Cách giải thích tương tự như trên

Vậy . - - - . EFF

10.7)

Vì kích thước một trang là 200 nên

1 trang sẽ bao gồm 2 hàng (vì 1 hàng có kích thước 100)

a) TH: mỗi lần lặp hàng sẽ thay đổi => truy cập lần lượt theo từng cột

- Vì ban đầu 2 page frames còn lại đều trống (page frame 1 chứa tiến trình) nên ở lần lặp đầu tiên đã xuất hiện 1 lỗi trang
- Mỗi lần truy cập đến một ô thì hàng của ô đó sẽ được tải lên bộ nhớ, vì 1 trang chứa 2 hàng nên cứ sau 2 lần lặp sẽ gây ra 1 lỗi trang, suy ra 100 hàng trên 1 cột sẽ có 100/2 = 50 (lỗi trang) mỗi lần..
- Có 100 cột nên tổng số lỗi trang sẽ là 50.100 = 5000 (lỗi trang)

b) TH: mỗi lần lặp cột sẽ thay đổi => truy cập lần lượt theo từng hàng

- Vì...
- Khi truy cập đến ô đầu tiên thì hàng của ô đó sẽ được tải lên bộ nhớ, vì mỗi lần lặp thì cột sẽ thay đổi nên sau khi đi qua hết 100 cột thì vẫn chưa xuất hiện lỗi trang nào Tuy nhiên vẫn gây ra 1 lỗi trang
- Tiến trình chỉ gặp lỗi trang sau 2 lần lặp mà có thay đổi hàng
- Suy ra có 100 hàng thì có tất cả 100/2 = 50 (lỗi trang)

10.8) Dấu * đại diện cho 1 lỗi trang xuất hiện.

- Số lượng frame = 1. Vì không có trang nào được truy xuất 2 lần liên tiếp nên mọi thuật toán đều xuất hiện 20 lỗi trang.
- Số lượng frame = 2:

· LRU

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
1	1	3	3	2	2	5	5	2	2	2	2	7	7	3	3	1	1	3	3
	2	2	4	4	1	1	6	6	1	1	3	3	6	6	2	2	2	2	6
*	*	*	*	*	*	*	*	*	*		*	*	*	*	*	*		*	*

= 18

· FIFO

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
1	1	3	3	2	2	5	5	2	2	2	3	3	6	6	2	2	2	3	3
	2	2	4	4	1	1	6	6	1	1	1	7	7	3	3	1	1	1	6
*	*	*	*	*	*	*	*	*	*		*	*	*	*	*	*		*	*

= 18

- Optimal

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6	
1	1	3	4	4	1	5	6	6	1	1	3	3	3	3	3	1	1	3	6	
	2	2	2	2	2	2	2	2	2	2	2	7	6	6	2	2	2	2	2	
*	*	*	*		*	*	*		*		*	*	*		*	*		*	*	= 15

Soit 4 frames = 3

- LRU

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6	
1	1	1	4	4	4	5	5	5	1	1	1	7	7	7	2	2	2	2	2	
	2	2	2	2	2	2	6	6	6	6	3	3	3	3	3	3	3	3	3	
		3	3	3	1	1	1	2	2	2	2	2	6	6	6	1	1	1	6	
*	*	*	*		*	*	*	*	*		*	*	*		*	*		*		= 15

- FIFO

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6	
1	1	1	4	4	4	4	6	6	6	6	3	3	3	3	2	2	2	2	6	
	2	2	2	2	1	1	1	2	2	2	2	7	7	7	7	1	1	1	1	
		3	3	3	3	5	5	5	1	1	1	1	6	6	6	6	6	3	3	
*	*	*	*		*	*	*	*	*		*	*	*		*	*		*	*	= 16

- Optimal

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6	
1	1	1	1	1	1	1	1	1	1	1	3	3	3	3	3	3	3	3	3	
	2	2	2	2	2	2	2	2	2	2	2	7	7	7	2	2	2	2	2	
		3	4	4	4	5	6	6	6	6	6	6	6	6	6	1	1	1	6	
*	*	*	*			*	*				*	*			*	*			*	= 11

Soit 4 frames = 4

- LRU

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6	
1	1	1	1	1	1	1	1	1	1	1	1	1	6	6	6	6	6	6		
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2		
		3	3	3	3	5	5	5	5	5	3	3	3	3	3	3	3	3		
			4	4	4	4	6	6	6	6	6	7	7	7	7	1	1	1	1	
*	*	*	*			*	*				*	*	*			*				= 10

- FIFO

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
1	1	1	1	1	1	5	5	5	5	5	3	3	3	3	3	1	1	1	1
	2	2	2	2	2	2	6	6	6	6	6	7	7	7	7	7	7	3	3
		3	3	3	3	3	3	2	2	2	2	2	6	6	6	6	6	6	6
			4	4	4	4	4	4	1	1	1	1	1	1	2	2	2	2	2
*	*	*	*			*	*	*	*		*	*	*		*	*		*	

= 14

- Optimal

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
1	1	1	1	1	1	1	1	1	1	1	1	7	7	7	7	1	1	1	1
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
		3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
			4	4	4	5	6	6	6	6	6	6	6	6	6	6	6	6	6
*	*	*	*			*	*					*				*			

= 8

So 1g grams = 5

- LRU

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
		3	3	3	3	3	6	6	6	6	6	6	6	6	6	6	6	6	6
			4	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3
						5	5	5	5	5	5	7	7	7	7	7	7	7	7
*	*	*	*			*	*				*	*							

= 8

- FIFO

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
1	1	1	1	1	1	1	6	6	6	6	6	6	6	6	6	6	6	6	6
	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1
		3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2
			4	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3
						5	5	5	5	5	5	7	7	7	7	7	7	7	7
*	*	*	*			*	*		*	*	*	*							

= 10

· Optimal

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6	
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	
		3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	
			4	4	4	4	6	6	6	6	6	6	6	6	6	6	6	6	6	
						5	5	5	5	5	5	7	7	7	7	7	7	7	7	= 7
*	*	*	*			*	*					*								

Sri Lanka games = 6

· LRU

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6	
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	
		3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	
			4	4	4	4	4	4	4	4	4	7	7	7	7	7	7	7	7	
						5	5	5	5	5	5	5	5	5	5	5	5	5	5	
							6	6	6	6	6	6	6	6	6	6	6	6	6	
*	*	*	*			*	*					*								= 7

· FIFO

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6	
1	1	1	1	1	1	1	1	1	1	1	1	7	7	7	7	7	7	7	7	
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1	
		3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	2	2	2	
			4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	3	3	
						5	5	5	5	5	5	5	5	5	5	5	5	5	5	
							6	6	6	6	6	6	6	6	6	6	6	6	6	
*	*	*	*			*	*					*				*	*	*		= 10

- Optimal

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
		3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
			4	4	4	4	4	4	4	4	4	7	7	7	7	7	7	7	7
						5	5	5	5	5	5	5	5	5	5	5	5	5	5
							6	6	6	6	6	6	6	6	6	6	6	6	6
*	*	*	*			*	*					*							= 7

Số lg frames = 3. Vì chỉ có 3 trang được truy cập mà có 3 frames nên mọi thuật toán đều có 7 lỗi trang.

10.22) Cách giải thích tương tự bài 10.5

Theo giả thiết ta có địa chỉ ảo có độ dài 16 bits, do đó sẽ có 2^{16} địa chỉ trong không gian địa chỉ ảo

Ngoài ra kích thước trang là 4096 bytes (có $4096 = 2^{12}$ địa chỉ trong một trang)

Do đó số lượng trang sẽ là $2^{16} / 2^{12} = 2^4$

Nên cần 4 bits để biểu diễn tất cả các trang => PageNumber = 4

Mà Logical Address (LA) = PageNumber + Page Offset

16 = 4 + Page Offset

=> Page Offset = 12

Vậy 4 bits quan trọng nhất trong địa chỉ ảo sẽ biểu thị số trang và 12 bits còn lại sẽ là độ lệch trong

Tuy nhiên kích thước page (trong không gian địa chỉ ảo) luôn giống với kích thước frame trong bộ nhớ chính. Do đó FrameOffset = PageOffset = 12, 12 bits cuối cùng sẽ vẫn giữ nguyên trong địa chỉ vật lý như địa chỉ ảo.

Từ nhận định trên, nếu muốn lấy địa chỉ vật lý, ta chỉ cần xét 4 bits đầu tiên.

+ 0X 621C, xét Page = 6, tra bảng ta sẽ được Page Frame = 8

Vậy ... 0x821C ①

+ 0x F0A3, xét Page = 15 (F), tra bảng ta ..... = 2
Vậy ... 0x 20A3

+ 0x BC1A, _____ = 11 (B) _____ = 4
_____ 0x 4C1A

+ 0x 5BAA, _____ = 5, _____ = 13 (C)
_____ 0x CBAA

+ 0x 0BA1, _____ = 0, _____ = 9
_____ 0x 9BA1

b) Xác địa chỉ luận lý sau dẫn đến lỗi trang: 1BAA, 91CB, ...

c) Vì các Page = {0, 5, 6, 11, 15} đã được truy xuất nên các Reference Bit ở các Page này sẽ bật lên 1.

Khi lỗi trang xuất hiện (Page = {1, 9, 14 (E)}), thuật toán LRU sẽ chọn Page Frame ít được sử dụng gần đây nhất (có Reference bit = 0) như các Page = {2, 3, 4, 7, 8, 10 (A), 12 (C), 13 (D)}