Các dạng bài tập

Tính các thông số đĩa từ:

+ Head(Track(Sector))

+ Tập hợp các track đồng tâm là Cyclinder

+ Head & Track đánh số từ 0, sector đánh số từ 1

+ Cứ mỗi lần chạy về vòng sector sẽ xuống head kế tiếp(VD: sector 17 = sector 18 – track 0 – head 0. Sector 18 = sector 1 – track 0 – head 1)

+ Sector vật lý > Logic: **l = t*side*st + h*st + s -1**

+ Sector logic > vật lý: **s = (l mod st) +1**

t = l div (st * side)

h = (l div st) mod side

st: số sector/track

th số track/side(head)

side: số lượng side

l: sector logic

h: giá trị head

t: giá trị track

s: giá trị sector

Các thuật toán đọc đĩa: Giả sử đầu đọc đang vị trí 11

FCFS: Theo thứ tự vào trước ra trước

STF: Chọn nhu cầu gần với vị trí hiện hành nhất

VD: Queue: 12 14 2 7 21 8 24

Cyclinder: 12 14 8 7 2 21 24

SCAN: Di chuyển về hướng xa nhất(vượt qua block cuối) rồi quay về hướng kia đến block cuối cùng

VD: Queue: 12 14 2 7 21 8 24

Cyclinder: 12 14 21 24 8 7 2

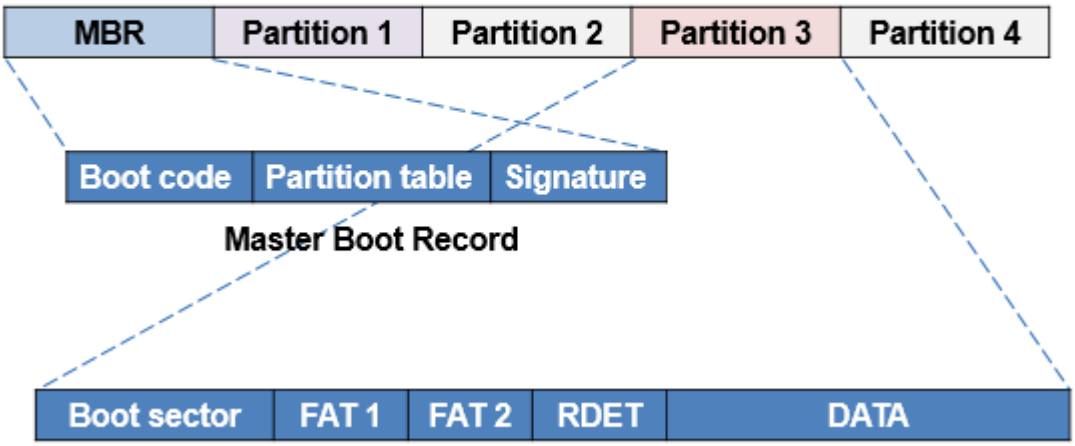
C-SCAN: Di chuyển về cuối > quay về đầu (ko xét block) > từ đầu chạy về đến hết block cuối cùng

VD: Queue: 12 14 2 7 21 8 24

Cyclinder: 12 14 21 24 2 7 8

Look — C-Look: Giống Scan & C-Scan nhưng di chuyển đến block xa nhất chứ không đến cuối cùng

ĐĨA TỪ - CẤU TRÚC



Các loại phân vùng:

0x07: Windows

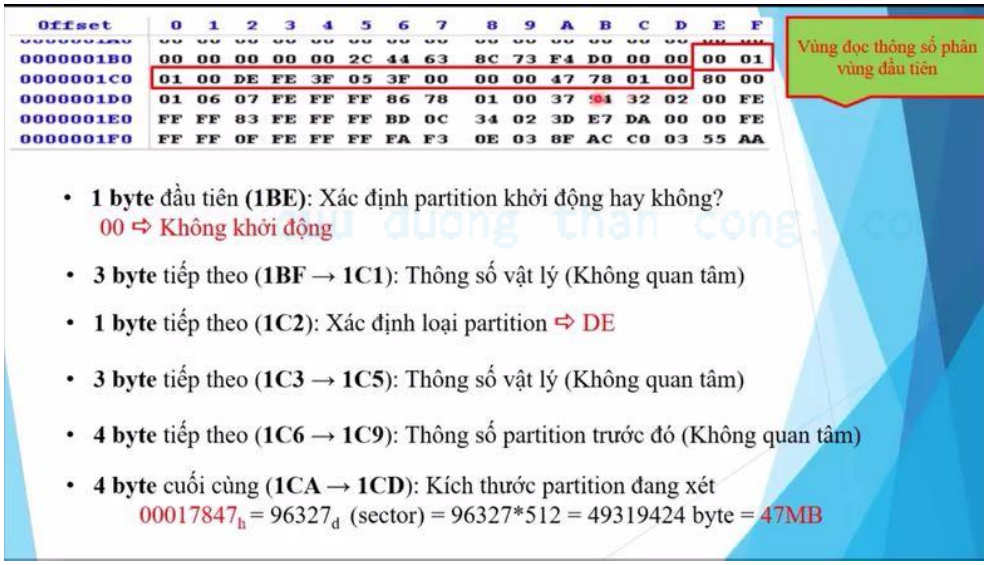
0x83: Linux

0x00: Không sử dụng

Loại partition:

06,0B: FAT— 07: NTFS — 0F: Extended—83: Linux

Master boot record



Bảng thông số Boot Sector của hệ thống tập tin FAT 16

Offset(hex)	Số byte	Ý nghĩa
0	3	Lệnh nhảy đến đầu đoạn mã boot
3	8	Tên công ty/version của HĐH
B	2	Số byte của sector, thường 512
D	**1**	**Số sector của cluster (Sc)**
E	**2**	**Số sector trước bảng FAT (Sb)**
10	**1**	**Số lượng bảng FAT (Nf), thường 2**
11	**2**	**Số entry của RDET(Sb) Thường 512 với FAT 16**
13	**2**	**Số sector của volume, bằng 0 nếu Sv>65535**
15	1	Kí hiệu loại volume
16	**2**	**Số sector của FAT (Sf)**
1	2	Số sector của track
1A	2	Số lượng đầu đọc(side)
1C	4	Khoảng cách từ nơi mô tả vol > đầu vol
20	**4**	**Kích thước volume (nếu số 2 byte tại offset 13h là 0)**
24	1	Ký hiệu vật lý của đĩa chứa vol (0 mềm, 80 cứng)
25	1	Dành riêng
26	1	Ký hiệu nhận diện của HĐH
27	4	SerialNumber của volume
2B	B	Volume Label
36	8	Loại FAT “FAT12”, “FAT16”,…chuỗi
3E	1CF	Đoạn chương trình Boot HĐH khi khởi động máy
1FE	2	Dấu hiệu kết thúc Boot Sector

Sector đầu tiên của vùng data: Ss = Sb + Nf*Sf +Sr

Sector đầu tiên của bảng RDET: Sb + Nf*Sf

- 2 byte tại offset 0B là: 00, 02

➔ Số byte trên mỗi sector của vol là: 0200h = 512 (byte)

- Giá trị của byte tại offset 0D là: 02

➔ Số sector trên mỗi cluster của vol là: S_C = 02h = 2 (sector)

- 2 byte tại offset 0E là: 08, 00

➔ Số sector trước vùng FAT là: S_B = 0008h = 8 (sector)

- Giá trị của byte tại offset 10 là: 02

➔ Số bảng FAT của vol là: N_F = 02h = 2d (bảng)

- 2 byte tại offset 11 là: 00, 02

➔ Số entry trên bảng RDET là: 0200h = 512 (entry)

➔ Kích thước bảng RDET là: S_R = (512*32) / 512 = 32 (sector).

- 2 byte tại offset 16 là: 20, 00

➔ Kích thước bảng FAT là: S_F = 0020h = 32 (sector)

- 2 byte tại offset 13 là: E0, 3F

➔ Tổng số sector trên vol là: S_V = 3FE0h = 16352 (vì 4 byte tại offset 20 đều là 00 nên kích thước vol được lấy ở 2 byte tại offset 13)

- Từ các thông số trên ta có thể tính ra được kích thước của vùng hệ thống:

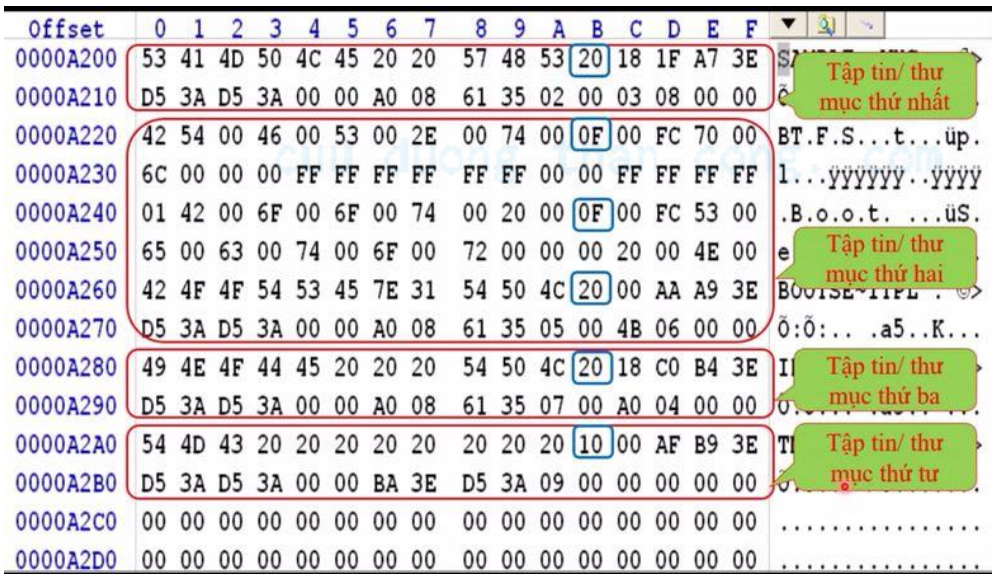
  $S_S = S_B + N_F * S_F + S_R$ = 8 + 2*32 + 32 = 104 (sector)

- Vậy vùng dữ liệu bắt đầu tại sector 104
  - cluster 2 sẽ chiếm 2 sector từ 104 đến 106
  - cluster 3 sẽ chiếm 2 sector từ 106 đến 108

  →Tổng quát, cluster K sẽ chiếm 2 sector bắt đầu tại sector có chỉ số 104 + 2*(K-2)

RDET

Xem offset 0B là entry chính hay phụ, nếu 0F là phụ, xét xuống tiếp



Nếu là entry chính xét offset 00:

00 > entry rỗng

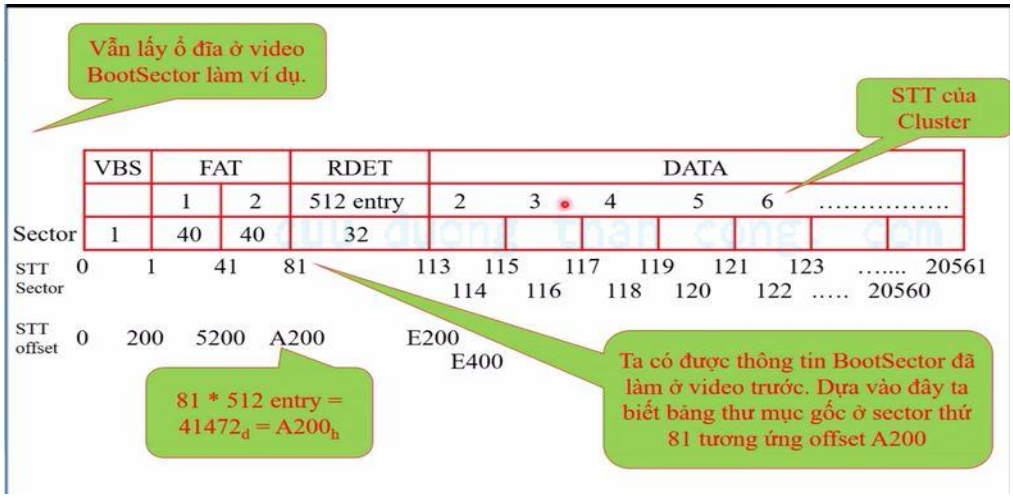
E5 > Tập tin/ thư mục bị xóa

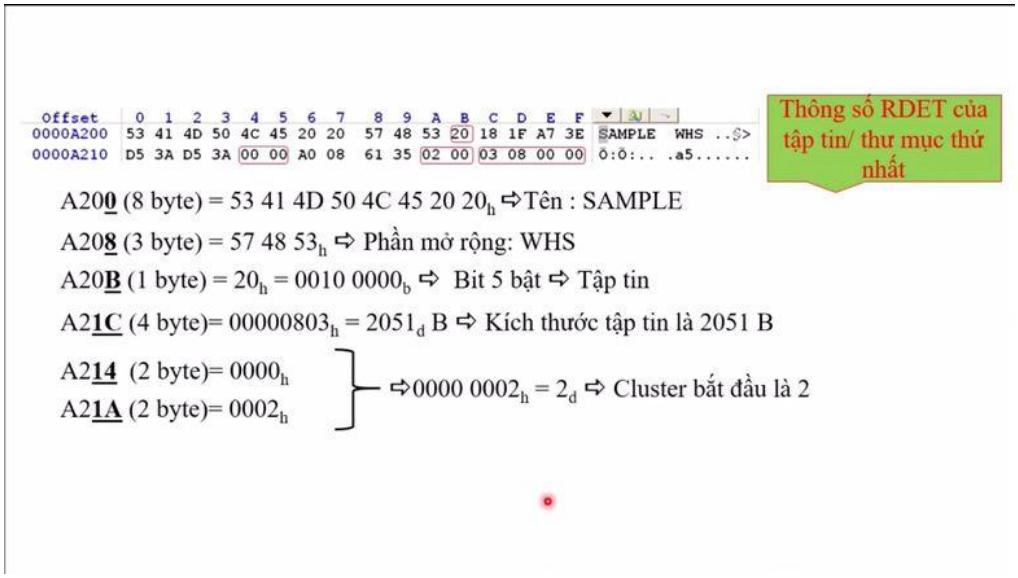
Khác > đang lưu trữ

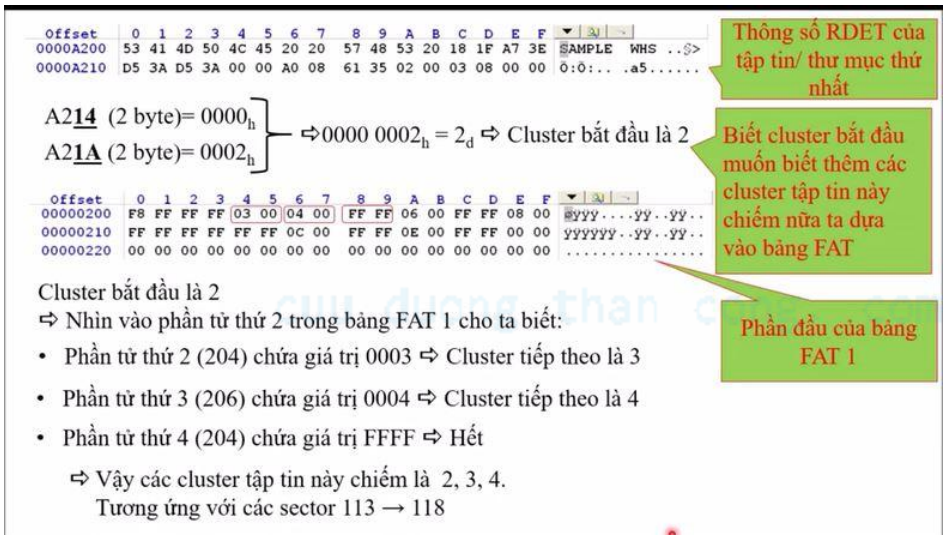
Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0000A200	53	41	4D	50	4C	45	20	20	57	48	53	20	18	1F	A7	3E	SAMPLE WHS ..§>
0000A210	D5	3A	D5	3A	00	00	A0	08	61	35	02	00	03	08	00	00	Õ:Õ:.. .a5.....
0000A220	42	54	00	46	00	53	00	2E	00	74	00	0F	00	FC	70	00	BT.F.S...t...üp.
0000A230	6C	00	00	00	FF	FF	FF	FF	FF	FF	00	00	FF	FF	FF	FF	l...
0000A240	01	42	00	6F	00	6F	00	74	00	20	00	0F	00	FC	53	00	.B.o.o.t. ...üS.
0000A250	65	00	63	00	74	00	6F	00	72	00	00	00	20	00	4E	00	e.c.t.o.r... .N.
0000A260	42	4F	4F	54	53	45	7E	31	54	50	4C	20	00	AA	A9	3E	BOOTSE~1TPL .ª©>
0000A270	D5	3A	D5	3A	00	00	A0	08	61	35	05	00	4B	06	00	00	Õ:Õ:.. .a5..K...
0000A280	49	4E	4F	44	45	20	20	20	54	50	4C	20	18	C0	B4	3E	INODE TPL .À´>
0000A290	D5	3A	D5	3A	00	00	A0	08	61	35	07	00	A0	04	00	00	Õ:Õ:.. .a5.. ...
0000A2A0	54	4D	43	20	20	20	20	20	20	20	20	10	00	AF	B9	3E	TMC ..¯¹>
0000A2B0	D5	3A	D5	3A	00	00	BA	3E	D5	3A	09	00	00	00	00	00	Õ:Õ:..º>Õ:.....
0000A2C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0000A2D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....

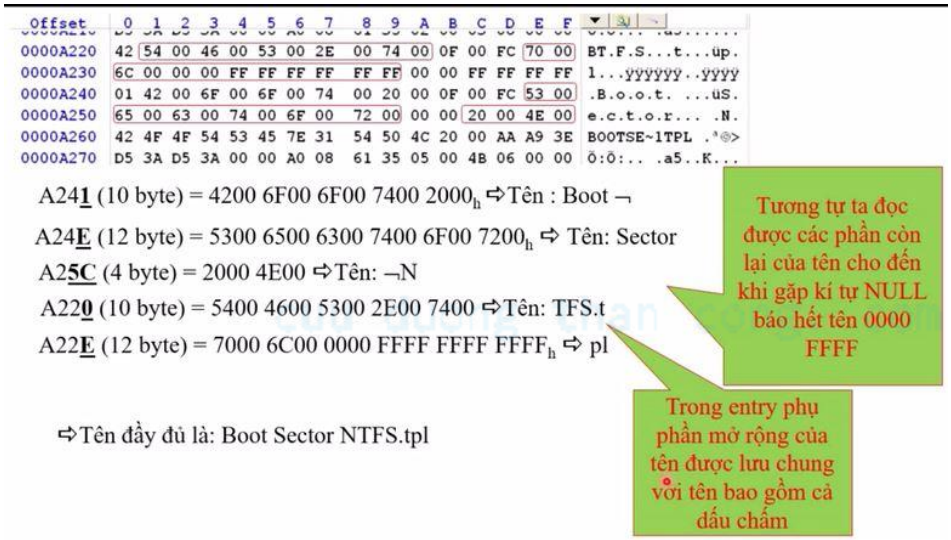
Đang lưu trữ tập tin/thư mục

DEC	HEX	SYM	DEC	HEX	SYM	DEC	HEX	SYM	DEC	HEX	SYM	DEC	HEX	SYM
0	0x00	NUL	25	0x19	EM	50	0x32	2	76	0x4C	L	102	0x66	f
1	0x01	SOH	26	0x1A	SUB	51	0x33	3	77	0x4D	M	103	0x67	g
2	0x02	STX	27	0x1B	ESC	52	0x34	4	78	0x4E	N	104	0x68	h
3	0x03	ETX	28	0x1C	FS	53	0x35	5	79	0x4F	O	105	0x69	i
4	0x04	EOT	29	0x1D	GS	54	0x36	6	80	0x50	P	106	0x6A	j
5	0x05	ENQ	30	0x1E	RS	55	0x37	7	81	0x51	Q	107	0x6B	k
6	0x06	ACK	31	0x1F	US	56	0x38	8	82	0x52	R	108	0x6C	l
7	0x07	BEL	32	0x20	SPACE	57	0x39	9	83	0x53	S	109	0x6D	m
8	0x08	BS	33	0x21	!	58	0x3A	:	84	0x54	T	110	0x6E	n
9	0x09	TAB	34	0x22	"	59	0x3B	;	85	0x55	U	111	0x6F	o
10	0x0A	LF	35	0x23	#	60	0x3C	<	86	0x56	V	112	0x70	p
11	0x0B	VT	36	0x24	$	61	0x3D	=	87	0x57	W	113	0x71	q
12	0x0C	FF	37	0x25	%	62	0x3E	>	88	0x58	X	114	0x72	r
13	0x0D	CR	38	0x26	&	63	0x3F	?	89	0x59	Y	115	0x73	s
14	0x0E	SO	39	0x27	'	64	0x40	@	90	0x5A	Z	116	0x74	t
15	0x0F	SI	40	0x28	(	65	0x41	A	91	0x5B	[	117	0x75	u
16	0x10	DLE	41	0x29	)	66	0x42	B	92	0x5C	\	118	0x76	v
17	0x11	DC1	42	0x2A	*	67	0x43	C	93	0x5D	]	119	0x77	w
18	0x12	DC2	43	0x2B	+	68	0x44	D	94	0x5E	^	120	0x78	x
19	0x13	DC3	44	0x2C	,	69	0x45	E	95	0x5F	_	121	0x79	y
20	0x14	DC4	45	0x2D	-	70	0x46	F	96	0x60	`	122	0x7A	z
21	0x15	NAK	46	0x2E	.	71	0x47	G	97	0x61	a	123	0x7B	{
22	0x16	SYN	47	0x2F	/	72	0x48	H	98	0x62	b	124	0x7C	|
23	0x17	ETB	48	0x30	0	73	0x49	I	99	0x63	c	125	0x7D	}
24	0x18	CAN	49	0x31	1	74	0x4A	J	100	0x64	d	126	0x7E	~
						75	0x4B	K	101	0x65	e	127	0x7F	DEL









Offset B: 10 – Thư mục, 20 – Tập tin

SDET

Các chiến lượt điều phối tiến trình

- **FIFO(FCFS)**
  Tiến trình nào vào trước thì chạy trước, sau khi hoàn thành tiến trình thì tiến trình tiếp theo chạy

MINH HỌA FCFS

P	$T_{arriveRL}$	CPU burst
P1	0	24
P2	1	3
P3	2	3

P	TT	WT
P1	24	0
P2	27-1	24-1
P3	30-2	27-2

$Avg_{WT} = (23+25)/3 = 16$

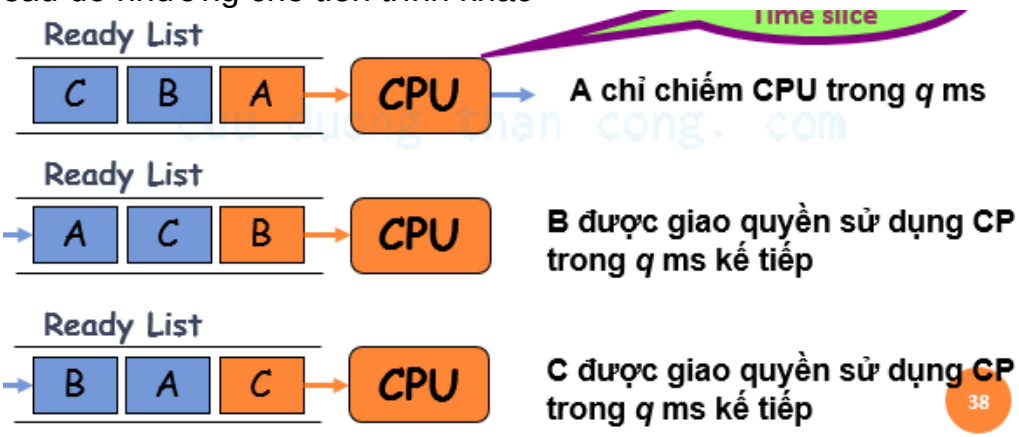
0: P1 vào RL
P1 dùng CPU
1: P2 vào RL
2: P3 vào RL

24: P1 kết thúc
P2 dùng CPU
27: P2 kết thúc
P3 dùng CPU

36

Nhận xét FCFS: Đơn giản, nhưng có thể xảy ra hiện tượng độc chiếm CPU(thời gian xử lý quá dài)

- **Điều phối Round Robin(RR)**
  Mỗi tiến trình chỉ sử dụng một lượng q(thời gian) cho mỗi lần sử dụng, sau đó nhường cho tiến trình khác



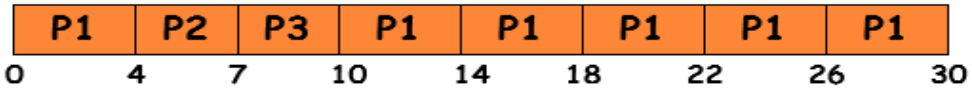
38

MINH HỌA RR, Q=4

P	$T_{arriveRL}$	CPU burst
P1	0	24
P2	1	3
P3	2	3

P	TT	WT
P1	30	0+(10-4)
P2	7-1	4-1
P3	10-2	7-2

$Avg_{WT} = (6+3+5)/3 = 4.66$



0:00 P1 vào, P1 dùng CPU
0:01 P2 vào (đợi)
0:02 P3 vào (đợi)
0:04 P1 hết lượt, P2 dùng CPU
0:07 P2 dừng, P3 dùng CPU
0:10 P3 dừng, P1 dùng CPU
0:14 P1 vẫn chiếm CPU
...

39

Nhận xét: Loại bỏ hiện tượng độc chiếm, nhưng phụ thuộc vào chọn lựa q, nếu chọn q không hợp lệ sẽ dẫn đến nhiều vấn đề khác

- **Điều phối với độ ưu tiên**
  Độc quyền: Lượt sử dụng kết thúc khi tiến trình kết thúc or bị khóa
  Không độc quyền: Lượt sử dụng khi tiếng trình kết thúc, bị khóa hoặc có tiến trình có độ ưu tiên cao hơn vào RL

ĐỘ ƯU TIÊN – KHÔNG ĐỘC QUYỀN

P	T_{RL}	Priority	CPU burst
P1	0	0	24
P2	1	2	3
P3	2	1	3

P	TT	WT
P1	30	0+(7-1)
P2	4-1	0
P3	7-2	4-2

$Avg_{WT} = (6+0+2)/3 = 2.66$

0: P1 vào, P1 dùng CPU
1: P2 vào (độ ưu tiên cao hơn P1)
P2 dành quyền dùng CPU
2: P3 vào (độ ưu tiên thấp hơn P2)
P3 không dành được quyền dùng CPU
4: P2 kết thúc, P3 dùng CPU
7: P3 dừng, P1 dùng CPU
30: P1 dừng

48

Nhận xét: Những tiến trình có độ ưu tiên thấp sẽ chờ cực lâu nếu những tiến trình còn lại quá nhiều và độ ưu tiên cao hơn

- **Shortest job first (SJF)**
  Độ ưu tiên được xếp theo số thời gian hiện hành còn lại của tiến trình

MINH HOA SJF (KHÔNG ĐỘC QUYỀN) (1)

P	$T_{arriveRL}$	CPU burst
P1	0	24
P2	1	3
P3	2	3

P	TT	WT
P1	30	0+(7-1)
P2	4-1	0
P3	7-2	4-2

$Avg_{WT} = (6+0+2)/3 = 2.66$

0:00 P1 vào, P1 dùng CPU
0:01 P2 vào (độ ưu tiên cao hơn P1)
P2 dành quyền dùng CPU
0:4 P2 kết thúc, P3 dùng CPU
0:7 P3 dừng, P1 dùng CPU
0:30 P1 dừng

53

MINH HOA SJF (KHÔNG ĐỘC QUYỀN) (2)

P	$T_{arriveRL}$	CPU burst
P1	0	24
P2	1	5
P3	3	4

P	TT	WT
P1	33	0+(10-1)
P2	5	0
P3	7	6-3

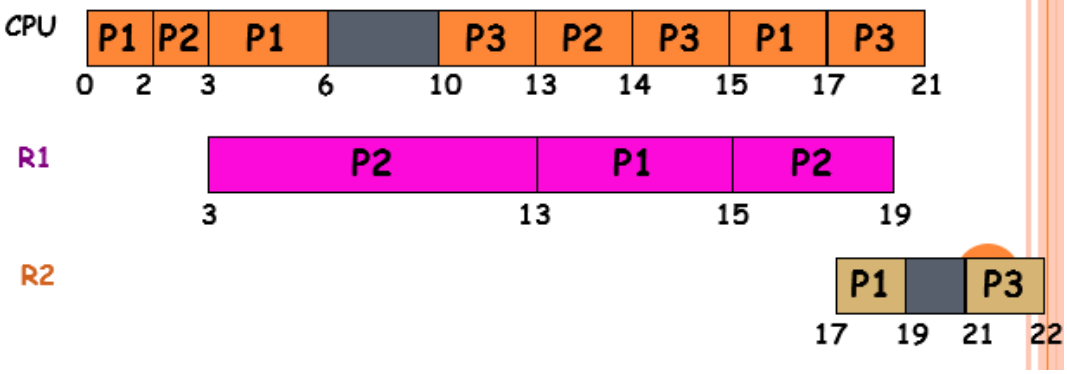
$Avg_{WT} = (9+0+3)/3 = 4$

0:00 P1 vào, P1 dùng CPU
0:01 P2 vào (độ ưu tiên cao hơn P1)
P2 dành quyền dùng CPU
0:03 P3 vào (độ ưu tiên < P2)
P2 dành quyền dùng CPU
0:6 P2 kết thúc, P3 dùng CPU
0:10 P3 dừng, P1 dùng CPU
0:33 P1 dừng

54

MINH HOA SJF (NHIỀU CHU KỲ CPU)

P	$T_{arriveRL}$	CPU1 burst	IO1 R	IO1 T	CPU2 burst	IO2 R	IO2 T
P1	0	5	R1	2	2	R2	2
P2	2	1	R1	10	1	R1	4
P3	10	8	R2	1	0	Null	0



cuu duong than cong. com

Cấp phát bộ nhớ động

VD: Có 3 vùng trống R1[512] – R2[250] – R3[700]

Tiến trình: P1[100] – P2[400] – P3[200]

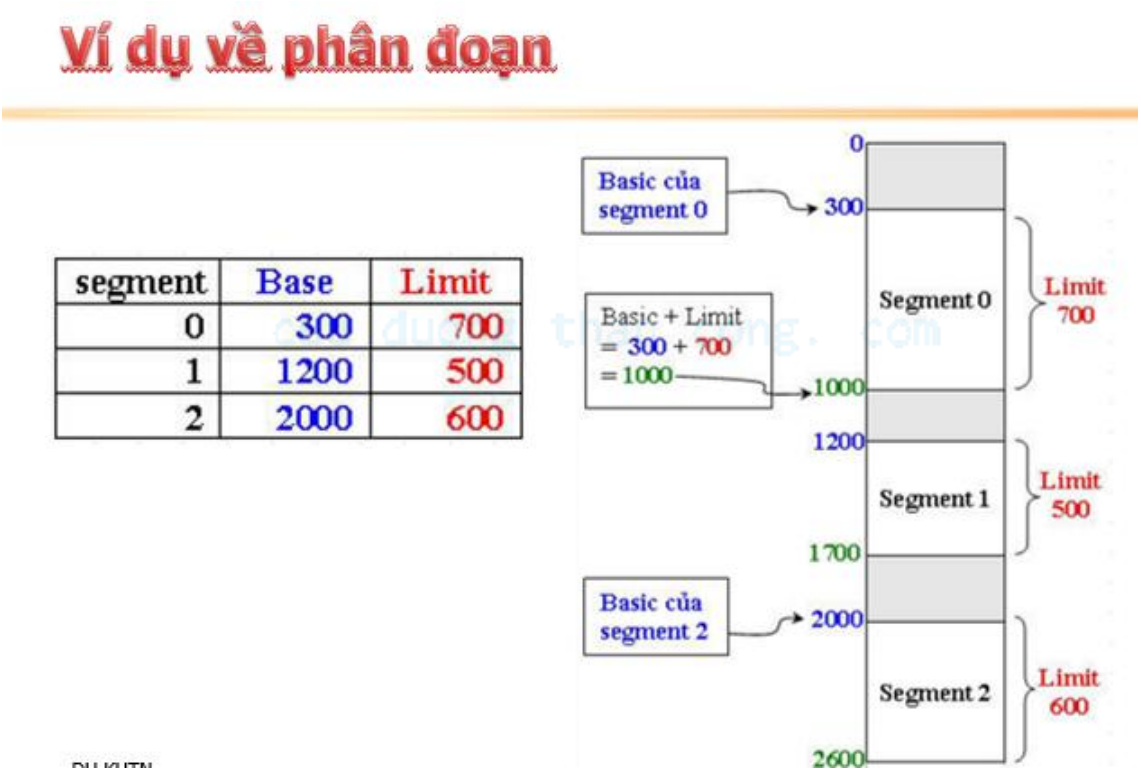
- **First fit:** Cấp phát vùng trống đầu tiên đủ yêu cầu

  Thực hiện chia: R1[P1 | P2 | 12] – R2[P3 | 50] – R3[700]
- **Best fit:** Cấp phát vùng trống nhỏ nhất đủ yêu cầu

  Thực hiện chia: R2[P1 | 150] – R1[P2 | 121] – R3[P3 | 500]
- **Worst fit:** Cấp phát vùng trống lớn nhất trong đó

  Thực hiện chia: R3[P1 | P2 | 200] – R1[P3 | 312] – R2[250]

Tính địa chỉ phân đoạn

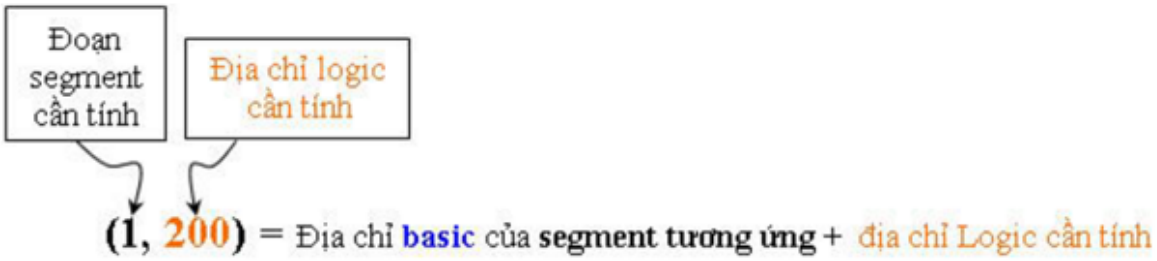


Giới hạn của một đoạn (segment từ base đến base+limit). Trong đây segment 0 giới hạn từ 300 đến 300+700=1000

Giả sử trong quá trình quản lý bộ nhớ ảo dạng phân đoạn, hệ điều hành duy trì Segment Table:

Segment	Base	Limit
0	300	700
1	1200	500
2	2000	600

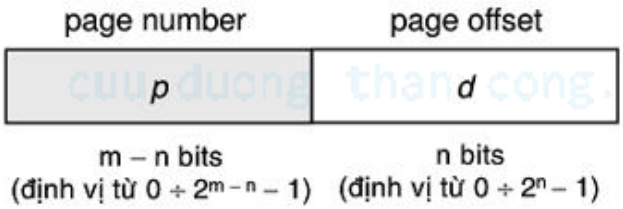
Hãy tính địa chỉ vật lý cho mỗi địa chỉ lô-gic sau: (1, 200), (1, 0), (0, 700), (2, 0), (2, 600)



+(1,200) = 1200 + 200 = 1400 (hợp lệ vì thuộc [1200:1700])
+ (1,0) = 1200 + 0 = 1200 (hợp lệ)
+ (0,700) = 300 + 700 = 1000 (hợp lệ)
+ (2,0) = 2000 + 0 = 2000 (hợp lệ)
+ (2, 600) = 2000 + 600 = 2600 (hợp lệ)

Cơ chế phân trang

- Địa chỉ luận lý gồm có:
  - *Số hiệu trang (Page number)* p
  - *Địa chỉ tương đối trong trang (Page offset)* d
- Nếu kích thước của không gian địa chỉ luận lý là 2^m, và kích thước của trang là 2^n (đơn vị là byte hay word tùy theo kiến trúc máy) thì



Bảng phân trang sẽ có tổng cộng $2^m/2^n = 2^{m-n}$ *mục* (entry)

VD: Kích thước bộ nhớ 2^5 = 32

Kích thước trang 2^2 = 4

⇨ 32/4 = 8 trang (2^(m-n))
Index từ 0 > 7
Page offset 0 > 3

Ví dụ kích thước trang là 1024, có địa chỉ logic:

a) 1251 => 1251/1024 = 1 dư 227 => nằm ở page số 1 bit thứ 227

b) 3249 => 3249/1024 = 3 dư 177 => nằm ở page số 3 bit thứ 177

Tính thời gian truy xuất hiệu dụng

Thời gian tìm kiếm ở thanh ghi kết hợp = ε (đơn vị thời gian)
Thời gian truy cập bộ nhớ là n đơn vị thời gian
Hit ratio: Số phần trăm (%) địa chỉ trang được tìm thấy ở các thanh ghi kết hợp/TLB
Hit ratio = α
Thời gian truy cập hiệu dụng (EAT):

$$EAT = (n + \varepsilon)\,\alpha + (2n + \varepsilon)(1 - \alpha) = 2n + \varepsilon - \alpha n$$

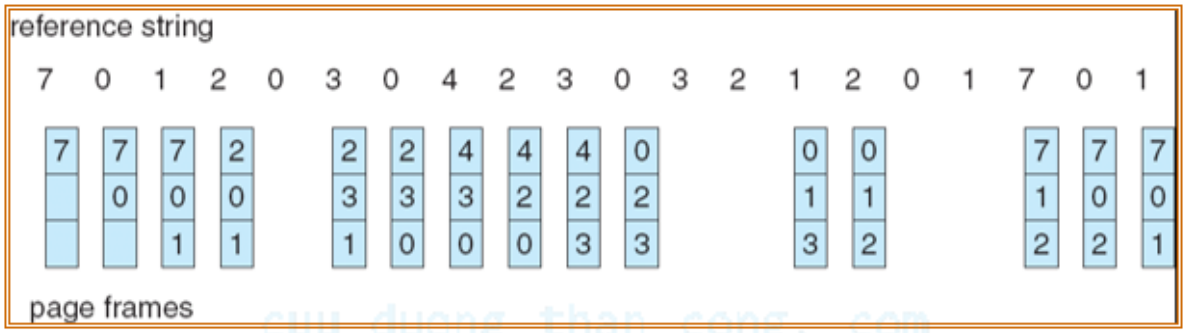
Xét một hệ thống sử dụng kỹ thuật phân trang, với bảng trang được lưu trữ trong bộ nhớ chính.
a)Nếu thời gian cho một lần truy xuất bộ nhớ bình thường là 200 nanoseconds, thì mất bao nhiêu thời gian cho một thao tác truy xuất bộ nhớ trong hệ thống này ?
b)Nếu sử dụng TLBs với hit-ratio (tỉ lệ tìm thấy) là 75%, thời gian để tìm trong TLBs xem như bằng 0, tính thời gian truy xuất bộ nhớ trong hệ thống (effective memory reference time)

a) Vì cần truy xuất 2 lần là index và offset nên mất 200*2 = 400 ns cho một thao tác truy xuất bộ nhớ
b) TLB hit 0.75 nên TLB miss 1-0.75 = 0.25. Vậy thời gian truy xuất bộ nhớ là : 0.75*200 + 0.25*400

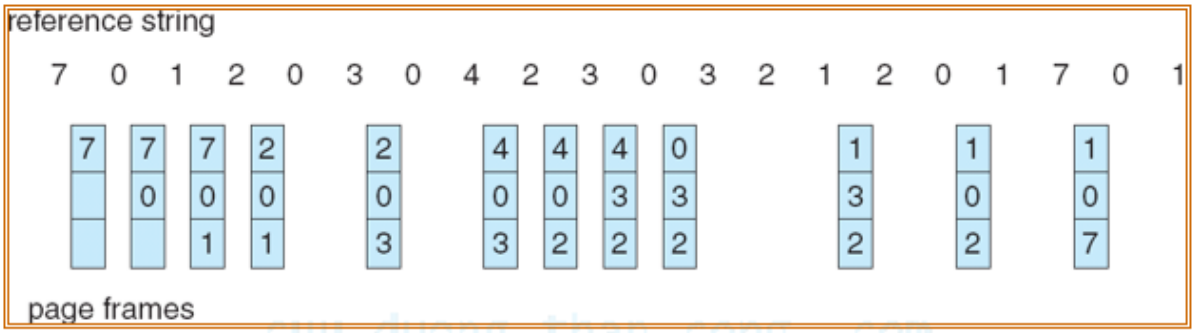
EAT (EMAT) = (1 – *p*) x (thời gian truy cập bộ nhớ)+ *p* ((thời gian phát hiện lỗi)
+ [swap page out]
+ swap page in
+ (thời gian restart quá trình xử lý))

Bài tập thay trang

- **FIFO:** Vào trước ra trước theo tuần tự

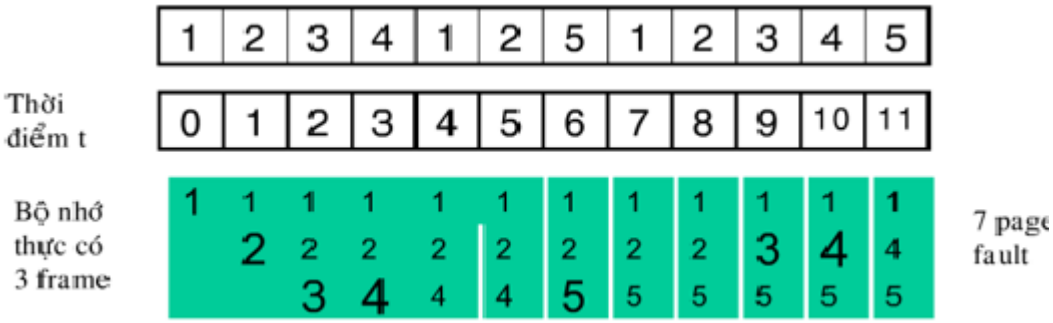


- **LRU:** Thay đi trang ít sử dụng gần đây nhất.
  Xem trước đó n(số trang -1) bước, tiến trình nào không xuất hiện thì thay đi



Như ví dụ trên thì ta có bảng 3 trang, nên xem 3-1 = 2 bước

- **Thuật toán tối ưu OPT:** Thay thế trang sẽ lâu đc sử dụng nhất trong tương lai
  Xem từ đó về sau n(số trang -1) bước, tiến trình nào không xuất hiện thì thay đi



- **Clock**

Clock (Cơ hội thứ hai)

Sắp xếp các trang thành vòng tròn, và dùng 1 đồng hồ

Dùng 1 use bit cho mỗi frame. Bật use bit lên khi mà frame đó được dùng.

Nếu use bit = 0, trang không sử dụng

Khi lỗi trang:

Di chuyển kim đồng hồ

Kiểm tra use bit

If 1, mới sử dụng, xóa và tiếp tục

If 0, chọn trang này để thay thế

Cơ hội thứ Nth

Tương tự ý tưởng trên nhưng,

Dùng 1 counter và 1 ***use bit***

Khi lỗi trang:

Dịch kim đồng hồ

Kiểm tra ***use bit***

If 1, xóa use bit và đặt counter = 0

If 0, tăng counter, if counter < N, tiếp tục, ngược lại chọn trang này để thay thế

Nhận xét

N lớn ⇒ gần giống kết quả với LRU

Nếu N quá lớn thì gặp vấn đề gì?