Phần 1: Cấu trúc logic của đĩa: Master Boot Record

I: Cấu trúc một phần tử của bảng phân chương

Stt	Offset	Size	Description
1	0	1B	Phân vùng tích cực? 80h nếu đúng; 0: Data
2	1	1B	Số hiệu mặt đĩa đầu của phân vùng
3	2	1W	Số hiệu sector và cylinder đầu của phân vùng
4	4	1B	Mã nhận diện hệ thống. 05/0F: Partition mở rộng; 06:Big Dos; 07:NTFS; 0B: FAT32,..
5	5	1B	Số hiệu đầu đọc cuối
6	6	1W	Số hiệu sector và cylinder cuối của phân vùng. (Số hiệu sector chỉ dùng 6 bit thấp)
7	8	1DW	Địa chỉ đầu, tính theo số hiệu sector
8	12	1DW	Số sector trong phân vùng

II: Bảng phân chương Master Boot Record

P1 80 01 01 00 01 07 14 64 14 00 00 00 0C 3F 00 00
P2 00 00 01 65 51 07 14 96 20 3F 00 00 40 1F 00 00
P3 00 00 01 97 51 07 54 2C 60 5E 00 00 C0 5D 00 00
P4 00 00 41 2D 51 07 54 F4 20 BC 00 00 00 7D 00 00

Partition	Boot	CHS Address (Vị trí đầu)			CHS Address (Vị trí cuối)			LBA #sector	Số sector	Partition type
		Hdr	Cyl	Sec	HdR	Cyl	Sec			
P1	Yes (80h)	1	0	1	7	100	20	20	16140	FAT 12
P2	No (00h)	0	101	1	7	150	20	16160	8000	Ontrack Disk Manager
P3	No (00h)	0	151	1	7	300	20	24160	24000	Ontrack Disk Manager
P4	No (00h)	0	301	1	7	500	20	48160	32000	Ontrack Disk Manager

III: Giải mã bảng phân chương MBR

Chú ý:

CHS Address (Cylinder-Head-Sector): Địa chỉ vật lý của partition.

H7 H6 H5 H4 H3 H2 H1 H0

C9 C8 C7 C6 C5 C4 C3 C2 C1 C0 S7 S6 S5 S4 S3 S2 S1 S0

LBA – Logical block addressing: Địa chỉ đầu tính theo số hiệu sector.

Khi giải mã bảng phân chương MBR (Master Boot Record) ta cần chú ý tới các bytes liên quan đến Sec (Sector) và Cyl (Cylinder). Khi tính toán, chuyển đổi các bytes này ta sẽ đọc theo kiểu little-endian (reversed) tức là các bytes sẽ được đọc từ phải qua trái trong quá trình chuyển đổi từ hệ cơ số hexa sang hệ thập phân.

Địa chỉ vật lý phân vùng được xác định bằng địa chỉ CHS bao gồm:

- 8 bits (1 byte) biểu diễn H
- 16 bits (2 bytes) biểu diễn C-S
- trong 16 bits biểu diễn C-S: 6 bits thấp biểu diễn S (**S5 S4 S3 S2 S1 S0**) và hai bits **S7 S6**
- sẽ được chuyển đến 2 bits cao của C thành (**C9 C8**) **C9 C8 C7 C6 C5 C4 C3 C2 C1 C0**

Giải thích:

Xét **P1** **80 01 01 00** **01 07 14 64** **14 00 00 00** **0C 3F 00 00**

Ta có:

80h - phân vùng tích cực, 00h phân vùng Data.

01h - số hiệu mặt đĩa - head: 0000 0001 (2 - hệ nhị phân) => H = **1** (10 - hệ thập phân).

01 00h - số hiệu C và S: do ghi chú ở trên khi tính toán C, S ta đọc theo kiểu little-endian do vậy:

01 00h sẽ thành 0000 0000 0000 0001(2)(**00 01h** - little-edian) => S = **1** (10 - hệ thập phân) (6 bits thấp 00 0001 (2 - hệ nhị phân)), C = **0** (10 - hệ thập phân) (00 0000 0000 (2 - hệ nhị phân)).

01h - mã nhận diện hệ thống (Partition type): 0000 0001 (2) = **1** (10) = **FAT 12**.

07h - số hiệu dầu đọc cuối: 0000 0111 (2) => H = **7** (10).

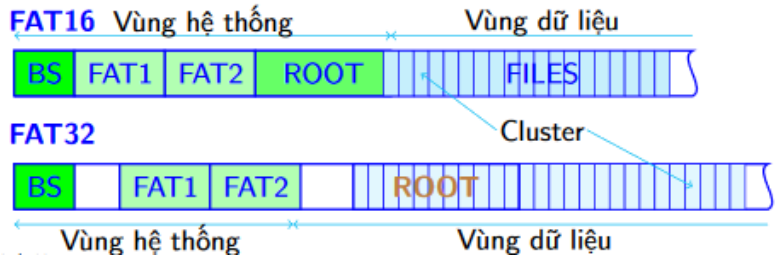
14 64h - số hiệu C và S: **64 14h** (reversed) = 0110 0100 0001 0100 (2) => S = **20** (10), C = **100** (10).

14 00 00 00h - LBA: => little-edian **00 00 00 14h (reversed)** = 0001 0100 (2) = **20** (10).

0C 3F 00 00h - số sector: => little-edian **00 00 3F 0Ch (reversed)** = 0011 1111 0000 1100 (2) = **16140** (10).

Tương tự cách tính như trên đối với các **P2**, **P3**, **P4** ta có được bảng giải mã MBR ở trên.

Phần 2: Hệ thống FAT



I: Boot sector

Cấu trúc gồm 3 phần:

- Bảng tham số đĩa (BPB: Bios Parameter Block)
- Chương trình mồi (*Boot strap loader*)
- Chữ ký hệ thống (*luôn là 55AA*)

II: Bảng tham số đĩa (BPB: Bios Parameter Block)

1: Phần chung cho FAT 12/16/32

Stt	Offset	Size	Description
1	0	3B	Nhảy đến đầu chương trình mồi
2	3	8B	Tên hệ thống file đã format đĩa
3	11	1W	K/thước 1 sector, thường là 512
4	13	1B	Số sector cho một cluster (32K-Cluster)
5	14	1W	Số scts đứng trước FAT/Số scts để dành
6	16	1B	Số bảng FAT
7	17	1W	Số phần tử của ROOT. FAT32: 00 00
8	19	1W	Tổng sector trên đĩa (< 32M) hoặc 0000
9	21	1B	Khuôn dạng đĩa (F8:HD, F0: Đĩa1.44M)
10	22	1W	Số sector cho một bảng FAT
11	24	1W	Số sector cho một rãnh
12	26	1W	Số đầu đọc ghi
13	28	1DW	Số sector ẩn- Sectors trước volume
14	32	1DW	Tổng số sector trên đĩa

2: Phần dành cho FAT 12/16

Stt	Offset	Size	Description
15	36	1B	Số hiệu ổ đĩa vật lý 0: ổ A; 80h: ổ C
16	37	1B	Để dành/Byte cao cho trường #ổ đĩa
17	38	1B	Boot sector mở rộng 29h
18	39	1DW	Volumn Serial number
19	43	11B	Volumn Label: nhãn đĩa
20	54	8B	Để dành, thường là đoạn text miêu tả dạng FAT
21	62	-	Bootstrap loader

3: Phần dành cho FAT 32

Stt	Offset	Size	Description
15	36	1DW	Tổng số sector cho bảng FAT
16	40	1W	Flags: #FAT chính
17	42	1W	Version: Phiên bản FAT32
18	44	1DW	Số hiệu cluster bắt đầu của ROOT
19	48	1W	#sector chứa File System information
20	50	1W	Số hiệu sector dùng backup Bootsector
21	52	12B	Để dành
22	64	1B	Số hiệu ổ đĩa vật lý 0: ổ A; 80h: ổ C
23	65	1B	Để dành/Byte cao cho trường #Driver
24	66	1B	Boot sector mở rộng. Luôn có giá trị 29h
25	67	1DW	Volumn Serial number
26	71	11B	Volumn Label: Nhãn đĩa
27	82	8B	Để dành, thường là đoạn text miêu tả dạng FAT

4: Ví dụ

EB 3C 90 57 49 4E 44 4F 57 4E 54 00 02 02 02 00

02 F4 01 00 00 F8 C8 00 24 00 28 00 00 02 00 00

80 32 02 00 80 00 29 D5 13 5B 24 44 55 4E 47 20

50 51 20 20 20 20 46 41 54 31 36 20 20 20 33 C9

Giải mã bảng tham số:

EB 3C 90h - 3 bytes đầu: Nhảy đến đầu chương trình mồi (bắt đầu từ byte **33**).

57 49 4E 44 4F 57 4E 54h - 8 bytes: Tên hệ thống file đã format đĩa: **WINDOWNT**.

00 02h - 2 bytes: Kích thước một sector: **512B** (10 - hệ thập phân) (little-edian: 02 00h = 0010 0000 0000 (2 – hệ nhị phân)).

02h - 1 byte: Số sector cho một cluster: **2** sector cho một cluster.

02 00h - 2 bytes: Số sector đứng trước bảng FAT thứ nhất: **2**.

02h - 1 byte: Số bảng FAT: **2**.

F4 01h - 2 bytes: Số phần tử của ROOT: **500** (kích thước một phần tử ROOT là 32B) => kích thước ROOT chiếm 31.25 sectors = 500*32/512.

00 00h - 2 bytes: Tổng số sector trên đĩa hoặc 0000.

F8h - 1 bytes: Mã nhận dạng khuôn dạng đĩa F8.

C8 00h - 2 bytes: Số sector cho một bảng FAT: **00 C8h (reversed)** = 1100 1000 (2) = **200** (10).

24 00h - 2 bytes: Số sector cho một rãnh: **00 24** = 0010 0100(2) = **36** (10).

28 00h - 2 bytes: Số đầu đọc ghi: **00 28h (reversed – little-edian)** = 0010 1000 (2) = **40** (10).

00 02 00 00h - 4 bytes: Số sector ẩn: **00 00 02 00** = 0010 0000 (2) = **32** (10).

80 32 02 00h - 4 bytes: Tổng số sector trên đĩa: **00 02 32 80h (reversed)** = 0010 0011 0010 1000 0000 (2) = **144000** (10) = 72000KiB = 70.3125MiB (1 sector = 512B).

80h -1 byte: Số hiệu ổ đĩa vật lý: **80** = ổ **C**.

00h - 1 byte: Để dành.

29h - 1 byte: Boot sector mở rộng **29h**.

D5 13 5B 24h - 4 bytes: Volume serial number: **245B-13D5**.

44 55 4E 47 20 50 51 20 20 20 20h - 11 bytes: Volume label - nhãn đĩa: **DUNG PQ**.

46 41 54 31 36 20 20 20h - 8 bytes: Kiểu FAT: **FAT16** (chuyển đổi **46 41 54 31 36 20 20 20h** sang mã ASCII).

33 C9h - bắt đầu chương trình mồi (boostrap loader).

III: Bảng FAT (File Allocation Table)

Một ổ đĩa có 17 cluster, kích thước của mỗi cluster là 1024 byte. Giả sử 17 phần tử đầu của bảng FAT có giá trị cho ở bảng sau:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	2	3	-1	0	0	13	8	9	-1	0	12	-1	14	16	0	-1

Và 3 entry đầu của Root Directory có giá trị sau:

Filename	Ext	Attrib	Start cluster	Size
Music		D	11	
Autoexec	bat		6	4032
Vidu	txt	R	7	3018

Các clusters dữ liệu của thư mục Music, file Autoexec.bat, Vidu.txt tương ứng là:

Root entry

Music		D		Time	Date	11	Size
Autoexec	bat			Time	Date	06	4032
Vidu	txt	R		Time	Date	07	3018

Music: M, Autoexec: A, Vidu: V, Free space: F, Other: O

FAT

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Special values		03	-1	00	00	13	08	09	-1	00	12	-1	14	16	00	-1

DATA

#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12	#13	#14	#15	#16
O	O	F	F	A	V	V	V	F	M	M	A	A	F	A

Từ bảng FAT ta thấy ổ đĩa đã sử dụng 11 clusters để phân cấp bộ nhớ cho các files và còn lại 4 clusters còn trống.

Các clusters 7, 8, 9 được sử dụng để phân cấp cho tập tin Vidu.txt (R Attribute: Read only), clusters 6, 13, 14, 16 sử dụng phân cấp cho tập tin Autoexec.bat và clusters 11, 12 phân cấp cho thư mục Music (D Attribute).

File Autoexec.bat được phân cấp bộ nhớ ở vị trí cluster đầu là cluster thứ 6 với kích thước 4032B chiếm gần hết 4 clusters (4096B) được phân cấp.

File Vidu.txt được phân cấp bộ nhớ ở vị trí cluster đầu là cluster thứ 7 với kích thước 3018B chiếm gần hết 3 clusters (3072B) được phân cấp.

Ổ đĩa còn dư 4 clusters (4096B) do vậy nếu cấp phát bộ nhớ cho file boot.ini có kích thước 4318B như vậy là không đủ. Do vậy yêu cầu thêm mới tệp tin boot.ini là không thực hiện được.

IV: Thư mục gốc (Root Directory)

48 45 54 48 4F 4E 47 20 53 59 53 00 00 00 00 00
00 00 00 00 00 00 29 6D 0E 71 C8 00 A7 5D 00 00
42 4F 4F 54 20 20 20 20 49 4E 49 00 00 00 00 00
00 00 00 00 00 00 19 8F 0A B5 AC 01 29 A2 00 00
56 49 52 55 53 20 20 20 45 58 45 00 00 00 00 00
00 00 00 00 00 00 32 10 12 62 56 02 A3 53 00 00

Giải mã thông tin 3 phần tử trong thư mục gốc:

48 45 54 48 4F 4E 47 20h - 8 bytes: Chuỗi ASII chứa tên file: **HETHONG**.

53 59 53h - 3 bytes: Chuỗi ASII chứa phần mở rộng: **SYS**.

00h - 1 byte: Thuộc tính của file: **00h** = 0000 0000 (2) = used.

U	U	A	D	V	S	H	R
0	0	0	0	0	0	0	0

U: Unused (), A: Archive, D: Subdirectory, V: Volume label, H: Hidden, R: Read only

00 00 00 00 00 00 00 00 00 00h - 10 bytes: Không dùng với FAT 12/16. Sử dụng với FAT 32.

29 6Dh - 2 bytes: Thời gian cập nhật cuối cùng: **6D 29h (reversed)** = 0110 1101 0010 1001 (2).

Cấu trúc trường thời gian: 5 bits (bits 15-11) cho trường giờ H, 6 bits (bits 10-5) cho trường phút M, 5 bits (bits 4-0) cho trường giây S = giây/2.

H4	H3	H2	H1	H0	M5	M4	M3	M2	M1	M0	S4	S3	S2	S1	S0
0	1	1	0	1	1	0	1	0	0	1	0	1	0	0	1
13					41						9				

Như vậy thời gian cập nhật file HETHONG.SYS là **13 giờ 41 phút 18 giây**.

0E 71h - 2 bytes: Ngày cập nhật cuối: **71 0Eh (reversed)** = 0111 0001 0000 1110 (2).

Cấu trúc trường ngày tháng: 7 bits (bits 15-9) cho trường năm Y + 1980, 4 bits (bits 8-5) cho trường tháng M, 5 bits (bits 4-0) cho trường ngày D.

Y6	Y5	Y4	Y3	Y2	Y1	Y0	M3	M2	M1	M0	D4	D3	D2	D1	D0
0	1	1	1	0	0	0	1	0	0	0	0	1	1	1	0
56							8				14				

Như vậy ngày tháng năm cập nhật file HETHONG.SYS là: **14 tháng 8 năm 2036**.

C8 00h - 2 bytes: Số hiệu cluster bắt đầu của file: **00 C8h (reversed)** = 1100 1000 (2) = **200** (10).

Chú ý: Nếu hệ thống file là FAT 32 thì số hiệu cluster bằng 2 bytes ở phần cao với vị trí Offset là 20 + 2 bytes ở phần thấp với vị trí Offset là 26.

Ví dụ: Giả sử ta lấy luôn bảng thư mục gốc ở trên làm ví dụ (không phải FAT 32) thì số hiệu cluster nếu là FAT 32 vị trí tương ứng sẽ là: **00 00 C8 00h = 00 C8 00 00h** (reversed).

A7 5D 00 00h - 4 bytes: Kích thước file tính bằng byte: **00 00 5D A7h (reversed)** = 0101 1101 1010 0111(2) = 23975B (10).

Tương tự cách tính toán trên ta có thông tin về hai phần tử còn lại theo thứ tự trên xuống trong bảng thư mục gốc là:

- File BOOT.INI với thuộc tính như HETHONG.SYS, Thời gian cập nhật cuối cùng: **17 giờ 56 phút 50 giây**, : Ngày cập nhật cuối: **10 tháng 8 năm 2070**, Số hiệu cluster bắt đầu của file: **428**, Kích thước file tính bằng byte: **41513B**.
- File VIRUS.EXE với thuộc tính như HETHONG.SYS, Thời gian cập nhật cuối cùng: **2 giờ 1 phút 36 giây**, : Ngày cập nhật cuối: **18 tháng 0 năm 2029**, Số hiệu cluster bắt đầu của file: **428**, Kích thước file tính bằng byte: **21411B**.

Phần 3: Quản lý bộ nhớ

I: Chiến lược phân trang

1: Nguyên tắc

- Bộ nhớ vật lý được chia thành từng khối có kích thước bằng nhau: *trang vật lý (frames).*
  - Trang vật lý được đánh số 0, 1, 2, ...: địa chỉ vật lý của trang.
  - Trang được dùng làm đơn vị phân phối nhớ.
- Bộ nhớ logic (*chương trình*) được chia thành từng trang có kích thước bằng trang vật lý: *trang logic (pages).*
- Bảng quản lý trang (*PCB: Page Control Block*) dùng để xác định mối quan hệ giữa trang vật lý và trang logic.
- Địa chỉ truy nhập được chia thành:
  - Số hiệu trang (**p**): Chỉ số trong PCB để tìm đ/chỉ cơ sở trang.
  - Độ lệch trong trang (**d**): Kết hợp địa chỉ cơ sở của trang để tìm ra đ/chỉ vật lý.

2: Ví dụ

Ví dụ 1: Một tiến trình được nạp vào bộ nhớ theo mô hình phân trang với kích thước trang là 512B. Bảng trang như sau:

2
6
5
3

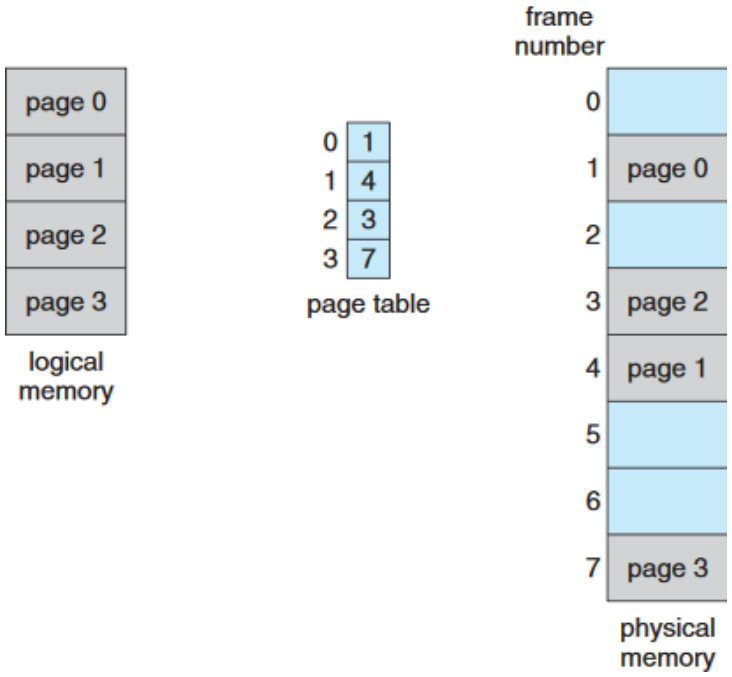
Hãy chuyển các địa chỉ logic sau thành địa chỉ vật lý:

a) **689**

b) **1613**

Do kích thước trang là 512B, bảng quản lý trang (PCB) có bốn phần tử do đó số hiệu trang logic (**p**) lần lượt là 0, 1, 2, 3 tương ứng với số hiệu trang vật lý (**f**) trong bảng PCB là: 2, 6, 5, 3.

Để hiểu rõ hơn về địa chỉ logic, đia chỉ vật lý và sự liên quan của chúng đến số hiệu trang logic, vật lý (**p**, **f**) kết hợp với độ lệch **d** ta xem hình minh họa dưới đây.



Ở hình vẽ trên ta có bộ nhớ logic (logical memory) được chia thành từng trang có kích thước bằng trang vật lý của bộ nhớ vật lý (physical memory).

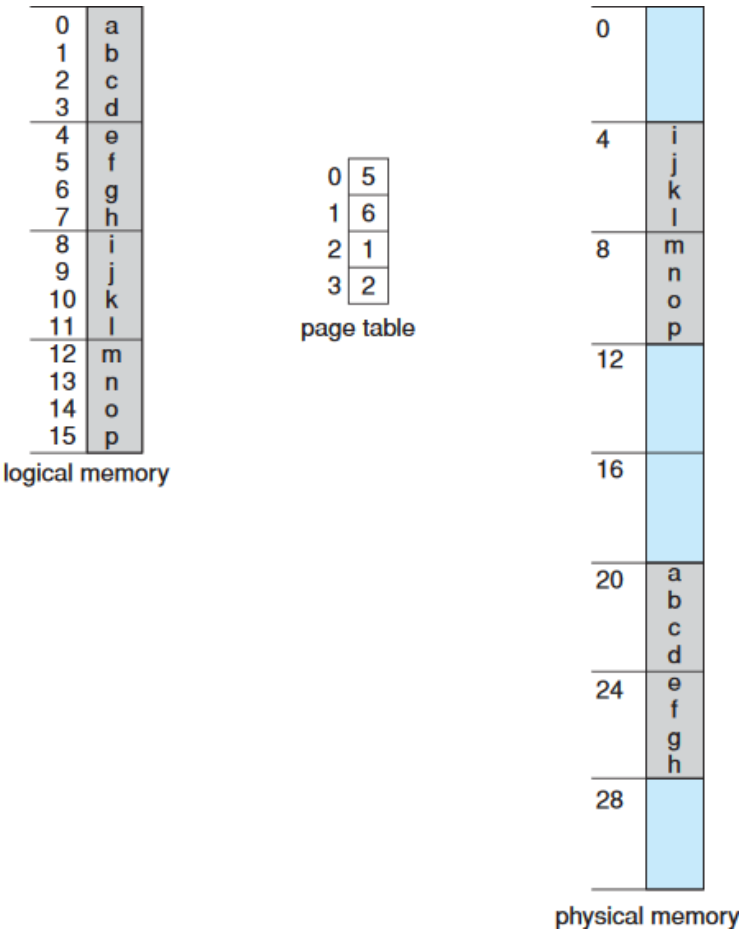
Bộ nhớ logic gồm có bốn trang với số hiệu trang được đánh dấu là 0 (page 0), 1 (page 1), 2 (page 2), 3 (page 3) - xem trong bảng quản lý trang (page table).

Với mỗi số hiệu trang logic sẽ tương ứng với số hiệu trang vật lý bên cạnh trong page table.

Như vậy trong hình ta thấy được page 0 có số hiệu trang logic là **0** tương ứng với số hiệu trang vật lý là **1**, page 1: **<p, f>** = **<1, 4>**, page 2: **<p, f>** = **<2, 3>**, page 3: **<p, f>** = **<3, 7>**.

Một điểm lưu ý nữa đó là các trang logic nằm ở vị trí liên tiếp nhau còn các trang vật lý nằm rải rác trên bộ nhớ vật lý.

Lưu ý: Đối với kiến trúc máy tính 32 bits, kích thước khung trang là 4KB = 2^12B => có tất cả 2^20 phần tử trong PCB. Như vậy cần tối thiểu 20 bits (thông thường sẽ sử dụng 32 bits = 4B) để lưu trữ một phần tử trong số 2^20 phần tử trong PCB => cần 20*2^20 = 2.5MB bộ nhớ để lưu trữ bảng PCB cho một tiến trình (mỗi tiến trình có một bảng PCB của riêng nó) => sẽ tốn khá nhiều tài nguyên bộ nhớ => giải pháp: trang nhiều mức.



Ta sẽ quan sát một hình ảnh mới chi tiết hơn các phần tử bên trong bộ nhớ logic và vật lý:

Do kích thước trang logic bằng kích thước trang vật lý, mà theo kiến trúc của bộ nhớ vật lý thì chúng được cấu thành từ các ngăn nhớ với kích thước thông thường là **1B**.

Như trên hình vẽ một trang nhớ gồm 4 phần tử là các kí tự trong bảng chữ cái với kích thước 4B.

Có tất cả 4 trang nhớ => kích thước bộ nhớ logic cần thiết là 16B với địa chỉ logic từ 0, 1, ..., 15.

Vậy độ lệch **d** để xác định địa chỉ truy nhập phần tử trong bộ nhớ ở đây là gì: đó chính là khảng cách từ vị trí đầu của trang mà phần tử thuộc về đến vị trí của phần tử đang đứng của trang đó.

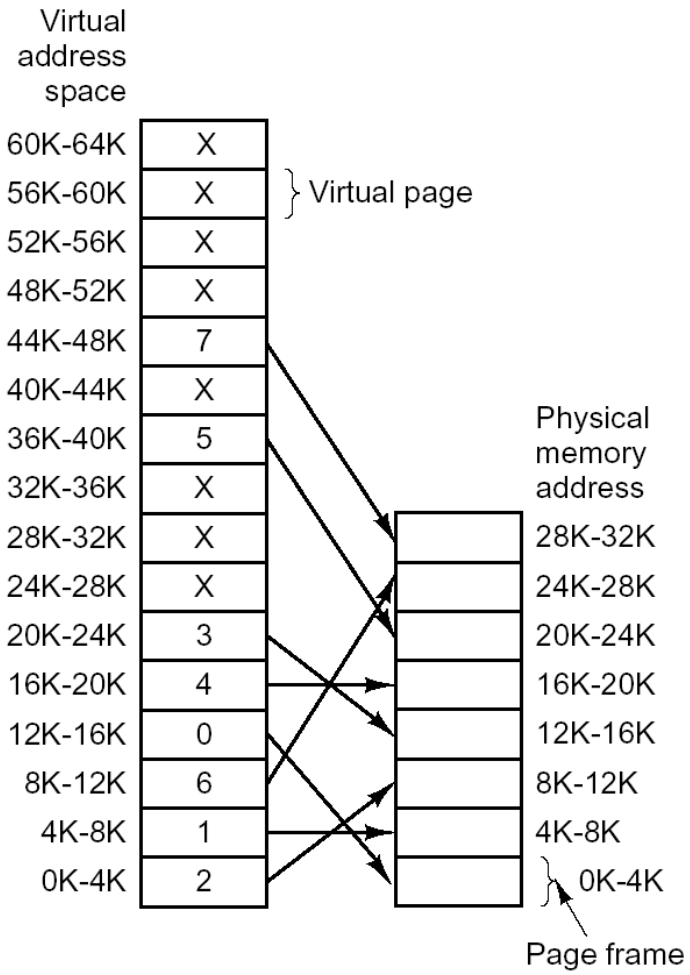
Ví dụ: phần tử chữ cái **h** đang đứng có địa chỉ logic là **7** thuộc trang có số hiệu trang là **1** với địa chỉ đầu của trang đó là 4 => độ lệch **d** ở đây là 7 - 4 = 3.

Mà số hiệ trang logic **1** tương ứng với số hiệu trang vật lý **6** mà kích thước trang vật lý bằng trang logic (**s** = 4B) => Vị trí của phần tử **h** trong bộ nhớ vật lý là: 6 * 4 + 3 = 27 (**f** * **s** + **d**).

Chú ý: Để biết vị trí bắt đầu phần tử trong trang vật lý hay trang logic ta chỉ cần lấy số hiệu trang tương ứng nhân với kích thước trang.

Quay trở lại ví dụ ban đầu: ta có địa chỉ logic 689 thuộc trang có số hiệu trang logic **p** = 1 ⇔ số hiệu trang vật lý **f** = 6, địa chỉ bắt đầu của trang này là 1*512 = 512 => **d** = 689 – 512 = 177 => địa chỉ vật lý tương ứng với **689** là: 6 * 512 + 177 = **3249**. Tương tự như trên ta có **1613** ⇔ **1613**.

Ví dụ 2: Xét một hệ thống quản lý bộ nhớ phân trang kích thước khung trang 4K. Tại thời điểm hiện tại trạng thái của bảng quản lý trang như hình vẽ. Xác định các giá trị địa chỉ vật lý tương ứng với các giá trị địa chỉ ảo sau: **20**, **4100**, **8300**.



Kích thước khung trang là 4KB = 4096B (có nhiều tài liệu sử dụng 1KiB = 1024B và 1KB = 1000B).

20 < 4095 (địa chỉ cuối của khung trang đầu bắt đầu từ 0 đến 4095 bao gồm cả 4095) => địa chỉ ảo (logic) thuộc khung trang thứ nhất với số hiệu trang logic **p = 0**, theo hình vẽ số hiệu khung trang sẽ bắt đầu từ dưới lên trên theo thứ tự tăng dần => số hiệu trang vật lý tương ứng là **f = 2** => địa chỉ đầu của khung trang vật lý là **f*4K = 2*4096 = 8192 (f*s)**. Độ lệch **d = 20 – 0 = 20** (địa chỉ logic trừ đi địa chỉ đầu của khung trang mà nó thuộc về) => địa chỉ vật lý cần tìm là **f*s + d = 8192 + 20 = 8212**.

4100 thuộc khung trang có số hiệu trang logic **p = 1** (4096 < 4100 < 8191), **f = 1**, **s = 4096**, **d = 4100 – 4096 = 4** => địa chỉ vật lý cần tìm là: 1*4096 + 4 = **4100**.

8300: **p = 2**, **f = 6**, **s = 4096**, **d = 108** => địa chỉ vật lý: **24684**.

II: Chiến lược đổi trang (bộ nhớ ảo)

FIFO (First In First Out): Vào trước ra trước.

OPT/MIN: Thuật toán thay thế trang tối ưu, đưa ra trang có lần sử dụng tiếp theo cách xa nhất.

LRU (Least Recently Used): Trang có lần sử dụng cuối cách lâu nhất.

Ví dụ 1: Một hệ thống quản lý bộ nhớ với 4 khung trang trong bộ nhớ vật lý, giả sử hệ thống chỉ có một tiến trình được chia thành 8 trang ảo được nạp và thực hiện. Ban đầu 4 khung trang đều trống. Giả thiết một chuỗi số hiệu các trang được tham chiếu theo trình tự thời gian như sau: 0 1 7 2 3 2 7 1 0 3. Hỏi có bao nhiêu lần lỗi trang xảy ra trong các chiến lược thay thế trang FIFO, OPT, LRU.

FIFO (6 lỗi)

0	1	7	2	3	2	7	1	0	3
0	0	0	0	3	-	-	-	3	-
	1	1	1	1	-	-	-	0	-
		7	7	7	-	-	-	7	-
			2	2	-	-	-	2	-

OPT (6 lỗi)

0	1	7	2	3	2	7	1	0	3
0	0	0	0	3	-	-	-	3	-
	1	1	1	1	-	-	-	0	-
		7	7	7	-	-	-	7	-
			2	2	-	-	-	2	-

LRU (7 lỗi)

0	1	7	2	3	2	7	1	0	3
0	0	0	0	3	-	-	-	0	0
	1	1	1	1	-	-	-	1	1
		7	7	7	-	-	-	7	7
			2	2	-	-	-	2	3

Ví dụ 2: Dãy truy nhập: 1 2 3 4 1 2 5 1 2 3 4 5 với 3 frames.

FIFO (9 lỗi) - các trang màu đỏ là các trang sẽ bị thay thế khi 3 frames đã đầy.

1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	4	4	4	5	-	-	5	5	-
	2	2	2	1	1	1	-	-	3	3	-
		3	3	3	2	2	-	-	2	4	-

OPT (7 lỗi)

1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	1	-	-	1	-	-	3	4	-
	2	2	2	-	-	2	-	-	2	2	-
		3	4	-	-	5	-	-	5	5	-

LRU (10 lỗi)

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

Lưu ý:

Đối với thuật toán FIFO: Ta có bảng các phần tử được nạp vào bộ nhớ đầu tiên đến tiếp theo, theo thứ tự từ trái qua phải sau mỗi lần nạp hoặc thay thế (các trang màu xanh là các trang nạp mới hoặc thay thế trang vào trước tiên trong bộ nhớ - các trang màu đỏ) trang như sau:

1			Nạp mới 1
1	2		Nạp mới 2
1	2	3	Nạp mới 3
2	3	4	Thay thế 4 cho 1
3	4	1	Thay thế 1 cho 2
4	1	2	Thay thế 2 cho 3
1	2	5	Thay thế 5 cho 4
2	5	3	Thay thế 3 cho 1
5	3	4	Thay thế 4 cho 2

Đối với thuật toán LRU: Trang có lần được sử dụng cuối cách lâu nhất sẽ nằm trái nhất của dãy truy nhập. Ví dụ ta thấy trên bảng LRU bên trên có 3 trang 5, 1, 2 tại cột thứ 7 từ trái sang màu xám sẽ có phần từ sử dụng cuối lâu nhất là **1** ở dãy truy nhập. Nếu tiếp theo của **5** là trang không tồn tại trong frames thì trang tiếp theo đó sẽ thay thế **1**.