

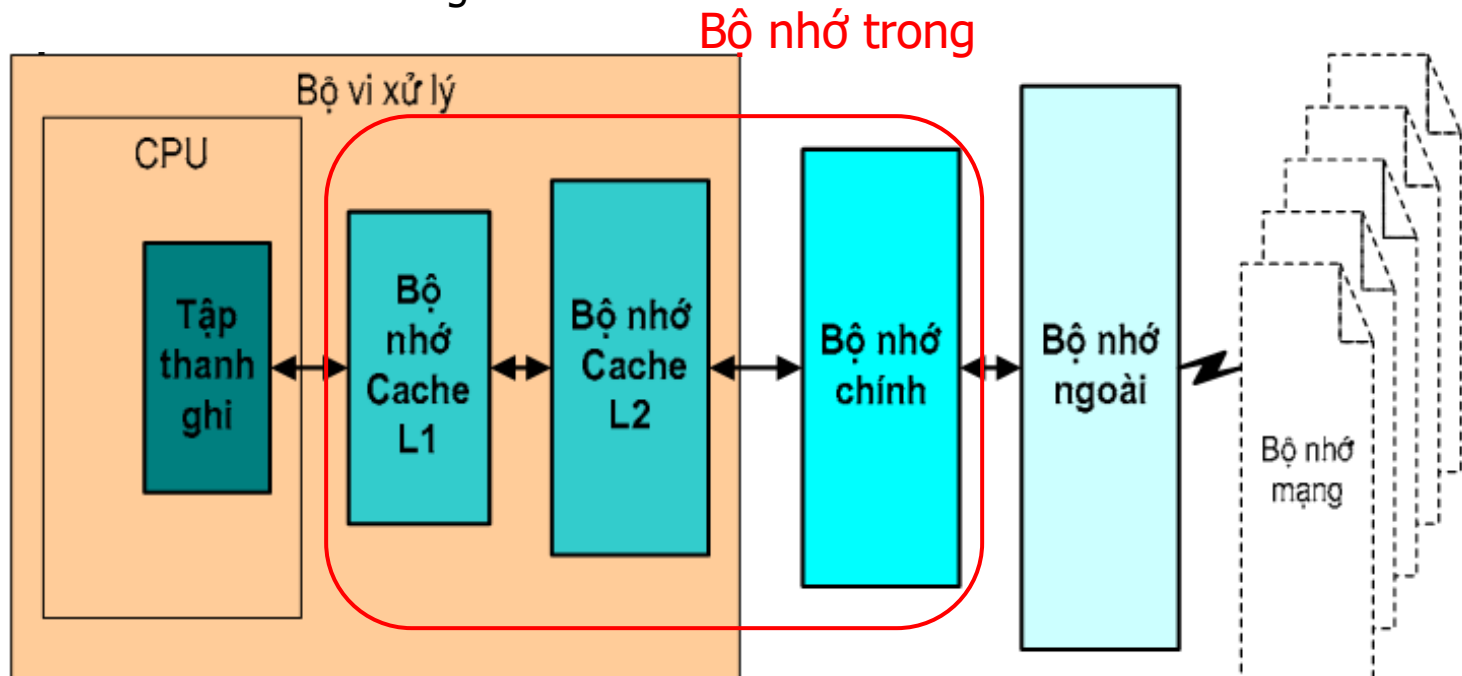
# HỆ THỐNG MÁY TÍNH

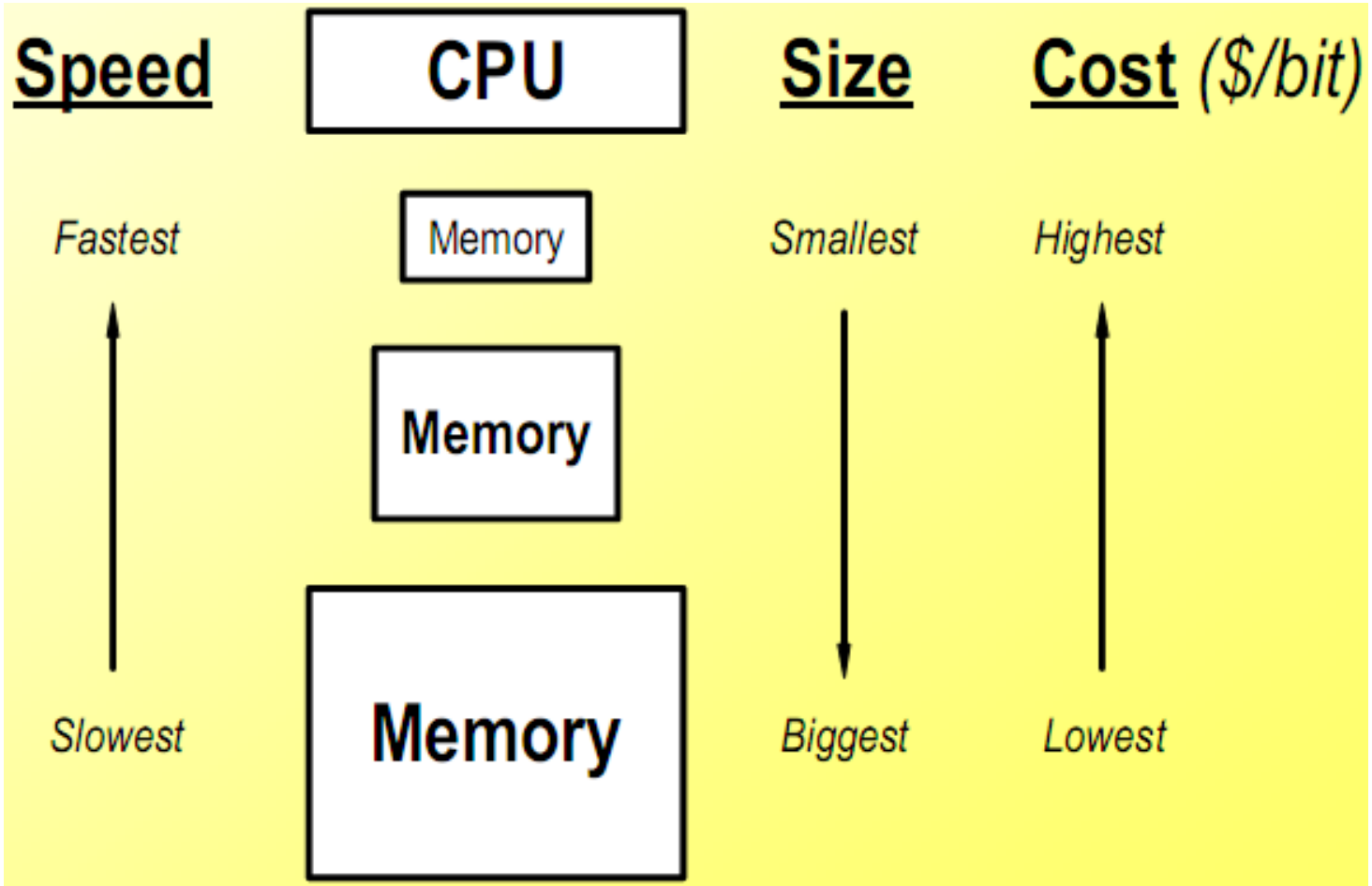
10 – Bộ nhớ

# Tổng quan về bộ nhớ

2

- Từ trái sang phải:
  - Dung lượng tăng dần
  - Tốc độ giảm dần
  - Giá thành trên 1 bit giảm dần



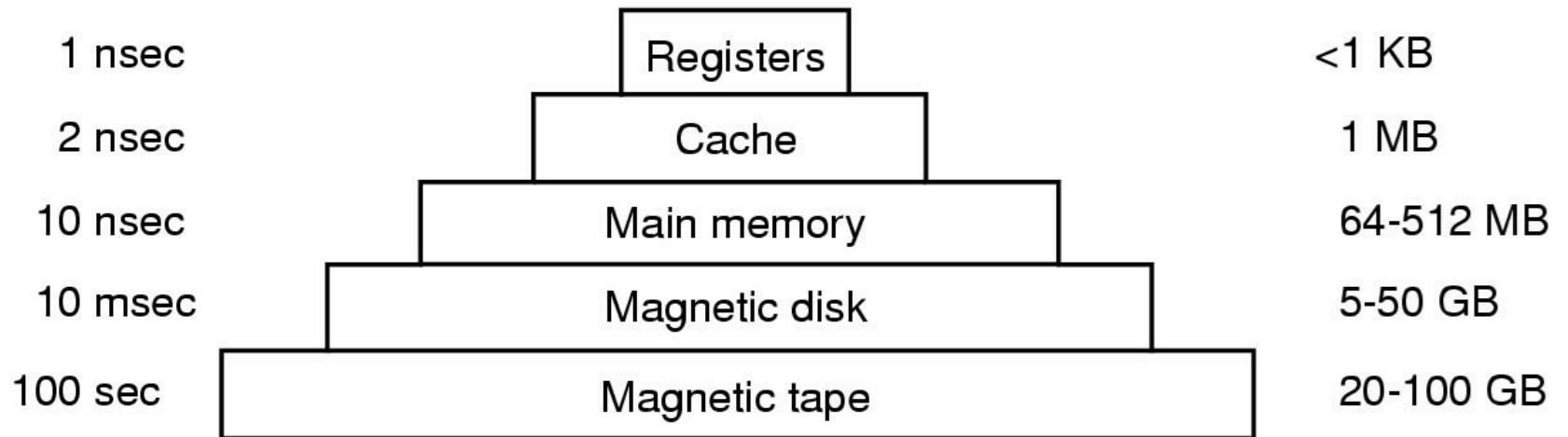


# Ví dụ

4

Typical access time

Typical capacity



# Phân loại

5

## □ Phương pháp truy cập

- Tuần tự (băng từ)
- Trực tiếp (các loại đĩa)
- Ngẫu nhiên (bộ nhớ bán dẫn như RAM, ROM)
- Liên kết (cache)

## □ Kiểu vật lý

- Bộ nhớ bán dẫn (cache, thanh ghi, RAM, ROM)
- Bộ nhớ từ (HDD, FDD)
- Bộ nhớ quang (CD-ROM, DVD)

# Bộ nhớ ngoài

6

- Một số bộ nhớ ngoài thông dụng:
  - ▣ Băng từ (Magnetic tape)
  - ▣ Đĩa từ (Magnetic disk)
  - ▣ Đĩa quang (Optical disk)
  - ▣ Flash disk

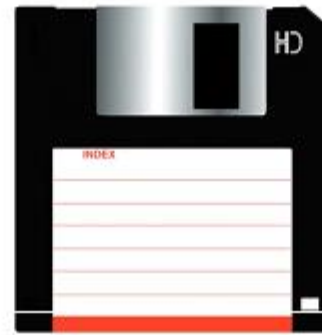
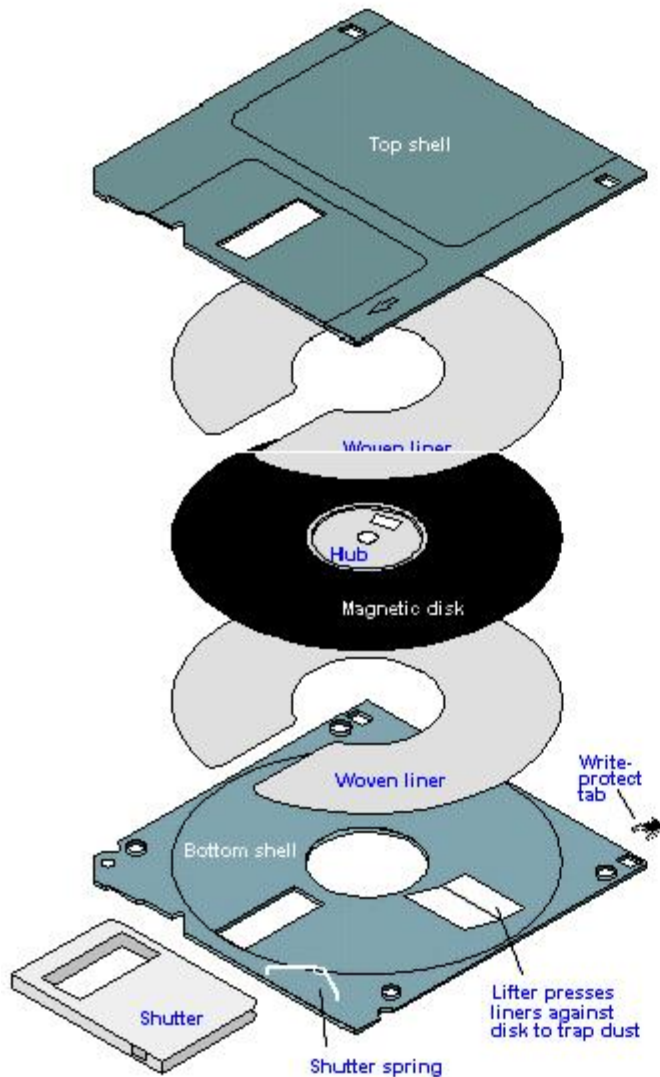
# Băng từ

7



# Đĩa từ: Đĩa mềm

8



3.5 inches, 1.44MB



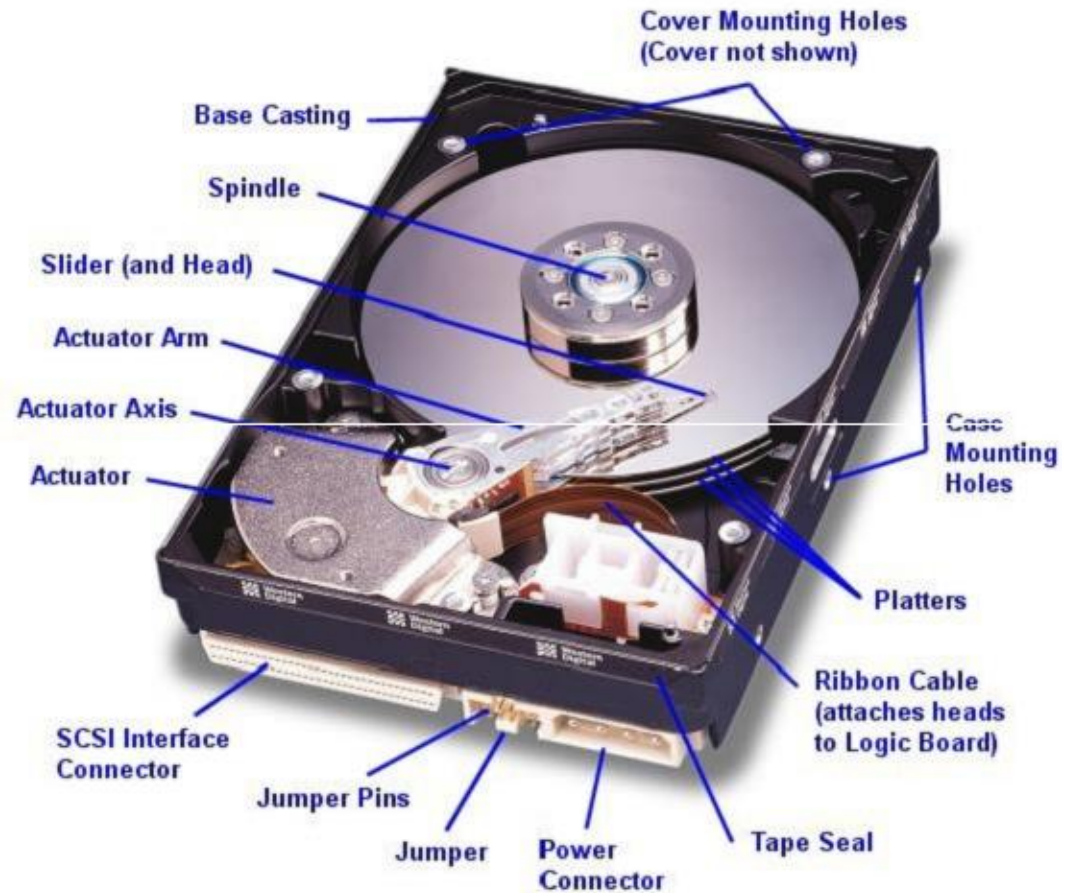
5.25 inches, 360KB





# Đĩa từ: Đĩa cứng

9



# Đĩa quang: CD

10



Type	Sectors	Data max size (MB)	Audio max size (MB)	Time (minute)
8 cm	94,500	193.536	222.264	21
650 MB	333,000	681.984	783.216	74
700 MB	360,000	737.28	846.72	80
800 MB	405,000	829.44	952.56	90
900 MB	445,500	912.384	1,047.82	99



52x / 32x / 52x : speed  
for CD-R / CD-RW / CD

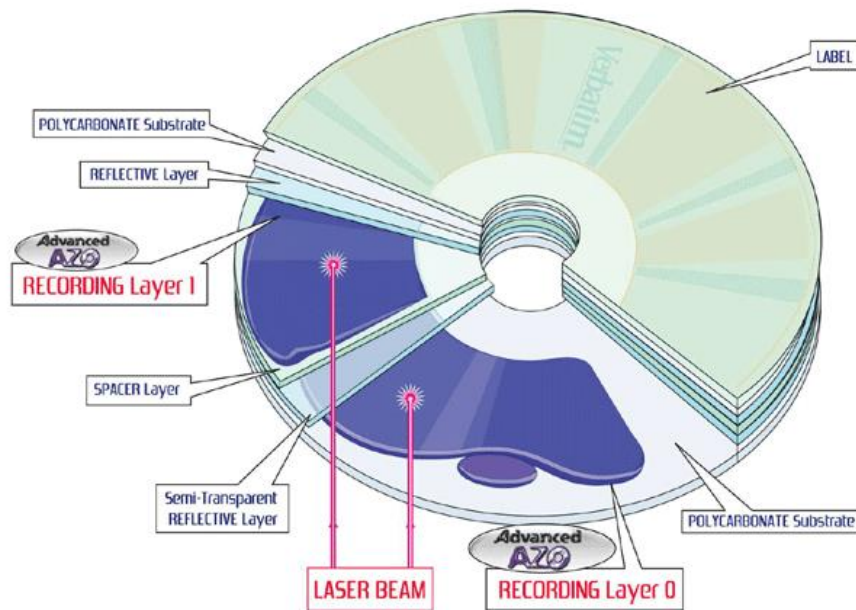


Transfer Speed	Megabytes/s	Megabits/s
1x	0.15	1.2
2x	0.3	2.4
4x	0.6	4.8
8x	1.2	9.6
10x	1.5	12
12x	1.8	14.4
20x	3	24
32x	4.8	38.4
36x	5.4	43.2
40x	6	48
48x	7.2	57.6
50x	7.5	60
52x	7.8	62.4

# Đĩa quang: DVD

11

- Digital Video Disk: chỉ dùng trên ổ đĩa xem video
- Ghi 1 hoặc 2 mặt, mỗi mặt có 1 (single layer) hoặc 2 lớp (double layer)
- Thông dụng: 4.7 GB/lớp



	Single layer (GB)	Double layer (GB)
12 cm, single sided	4.7	8.5
12 cm, double sided	9.4	17.1
8 cm, single sided	1.4	2.6
8 cm, double sided	2.8	5.2

# HD-DVD & Blue-ray Disc

12



**HD DVD**

	Single layer	Dual layer
12 cm, single sided	15 GB	30 GB
12 cm, double sided	30 GB	60 GB
8 cm, single sided	4.7 GB	9.4 GB
8 cm, double sided	9.4 GB	18.8 GB

**BD**

	Single layer	Dual layer
12 cm, single sided	25 GB	50 GB
8 cm, single sided	7.8 GB	15.6 GB



# Flash disk

13



# Memory card

14

Name	Acronym	Form factor
PC Card	PCMCIA	85.6 × 54 × 3.3 mm
CompactFlash I	CF-I	43 × 36 × 3.3 mm
CompactFlash II	CF-II	43 × 36 × 5.5 mm
SmartMedia	SM / SMC	45 × 37 × 0.76 mm
Memory Stick	MS	50.0 × 21.5 × 2.8 mm
Memory Stick Duo	MSD	31.0 × 20.0 × 1.6 mm
Memory Stick Micro M2	M2	15.0 × 12.5 × 1.2 mm
Multimedia Card	MMC	32 × 24 × 1.5 mm
Reduced Size Multimedia Card	RS-MMC	16 × 24 × 1.5 mm
MMCmicro Card	MMCmicro	12 × 14 × 1.1 mm
Secure Digital Card	SD	32 × 24 × 2.1 mm
miniSD Card	miniSD	21.5 × 20 × 1.4 mm
microSD Card	microSD	11 × 15 × 1 mm
xD-Picture Card	xD	20 × 25 × 1.7 mm
Intelligent Stick	iStick	24 × 18 × 2.8 mm
μ card	μcard	32 × 24 × 1 mm



**Card reader**

CompactFlash,  
Memory Stick,  
Secure Digital,  
and xD



# Hệ thống nhớ lưu trữ lớn: RAID

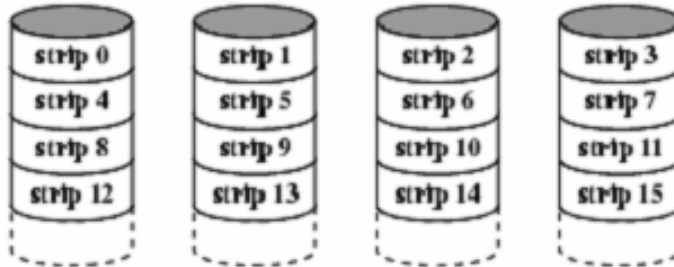
15

- ❑ Redundant Array of Inexpensive (Independent) Disks
- ❑ Tập các đĩa cứng vật lý được OS xem như 1 ổ logic duy nhất có **dung lượng lớn**
- ❑ Dữ liệu được lưu trữ phân tán trên các ổ đĩa vật lý → **truy cập song song (nhanh)**
- ❑ Có thể sử dụng dung lượng dư thừa để lưu trữ các thông tin kiểm tra chẵn lẻ, cho phép khôi phục lại thông tin khi đĩa bị hỏng → **an toàn thông tin**
- ❑ Có 7 loại phổ biến (RAID 0 – 6)

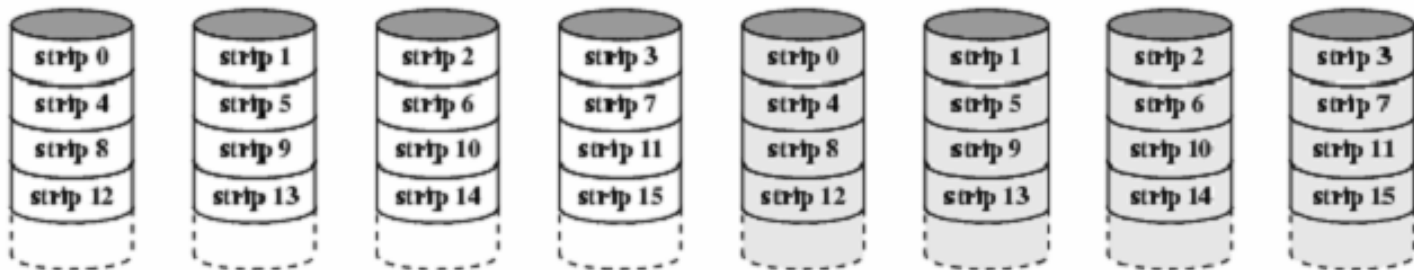


# RAID 0, 1, 2

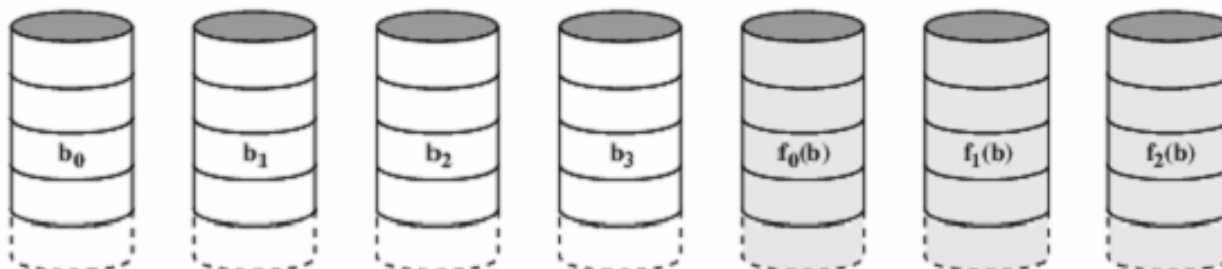
16



(a) RAID 0 (non-redundant)



(b) RAID 1 (mirrored)

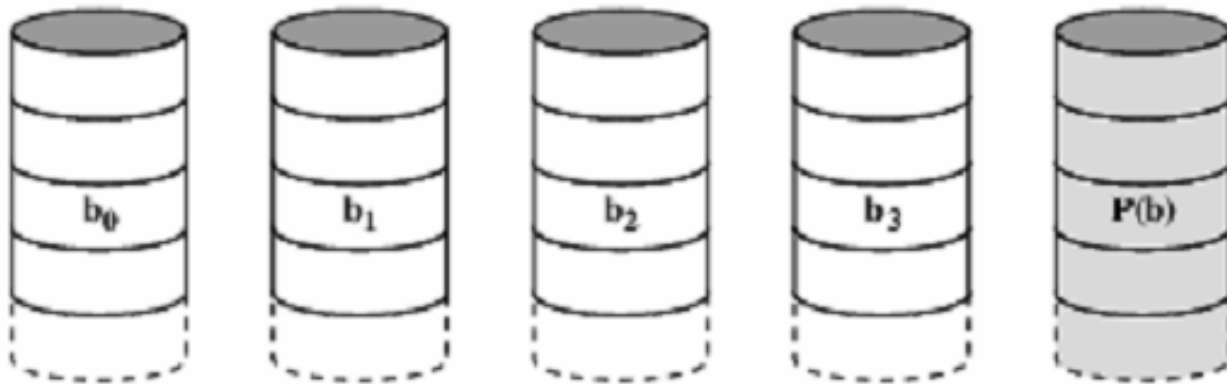


(c) RAID 2 (redundancy through Hamming code)

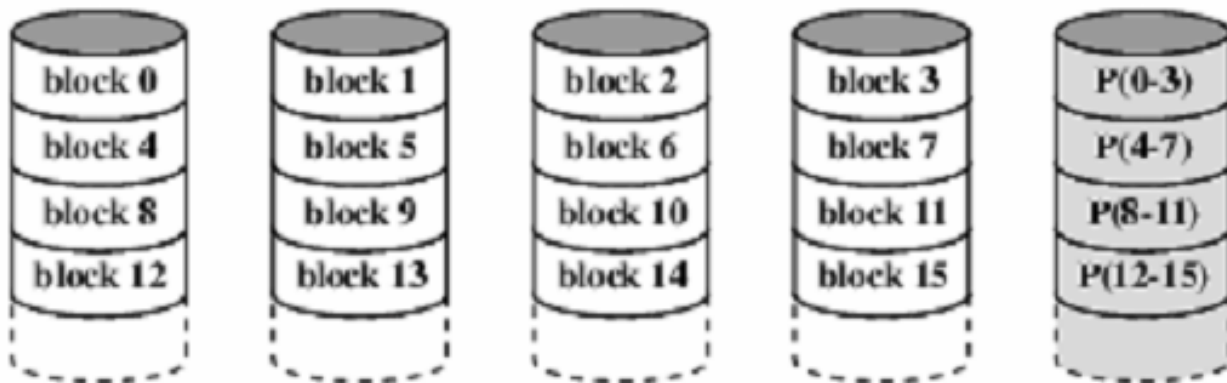


# RAID 3, 4

17



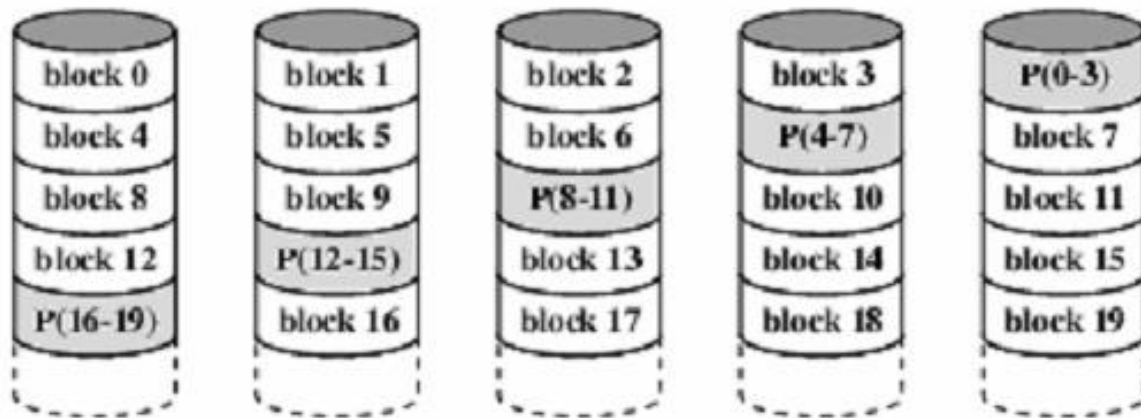
(d) RAID 3 (bit-interleaved parity)



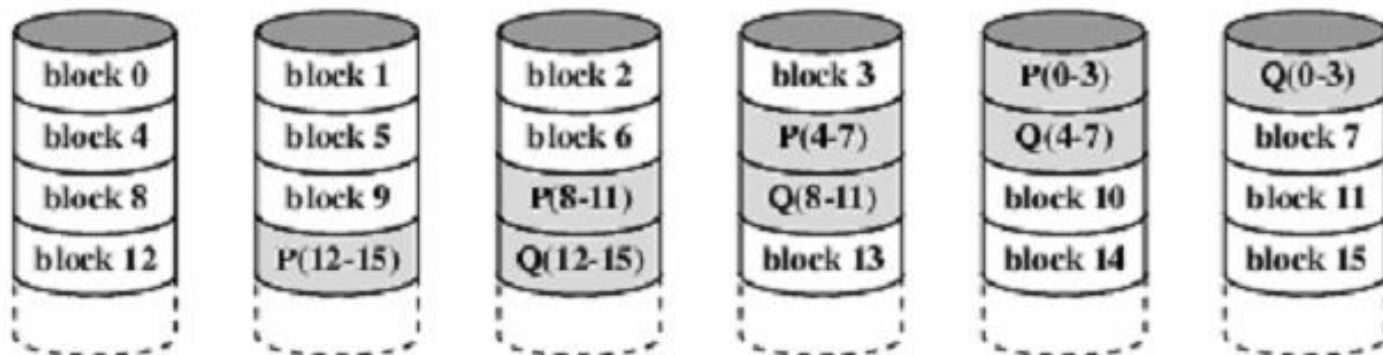
(e) RAID 4 (block-level parity)

# RAID 5, 6

18



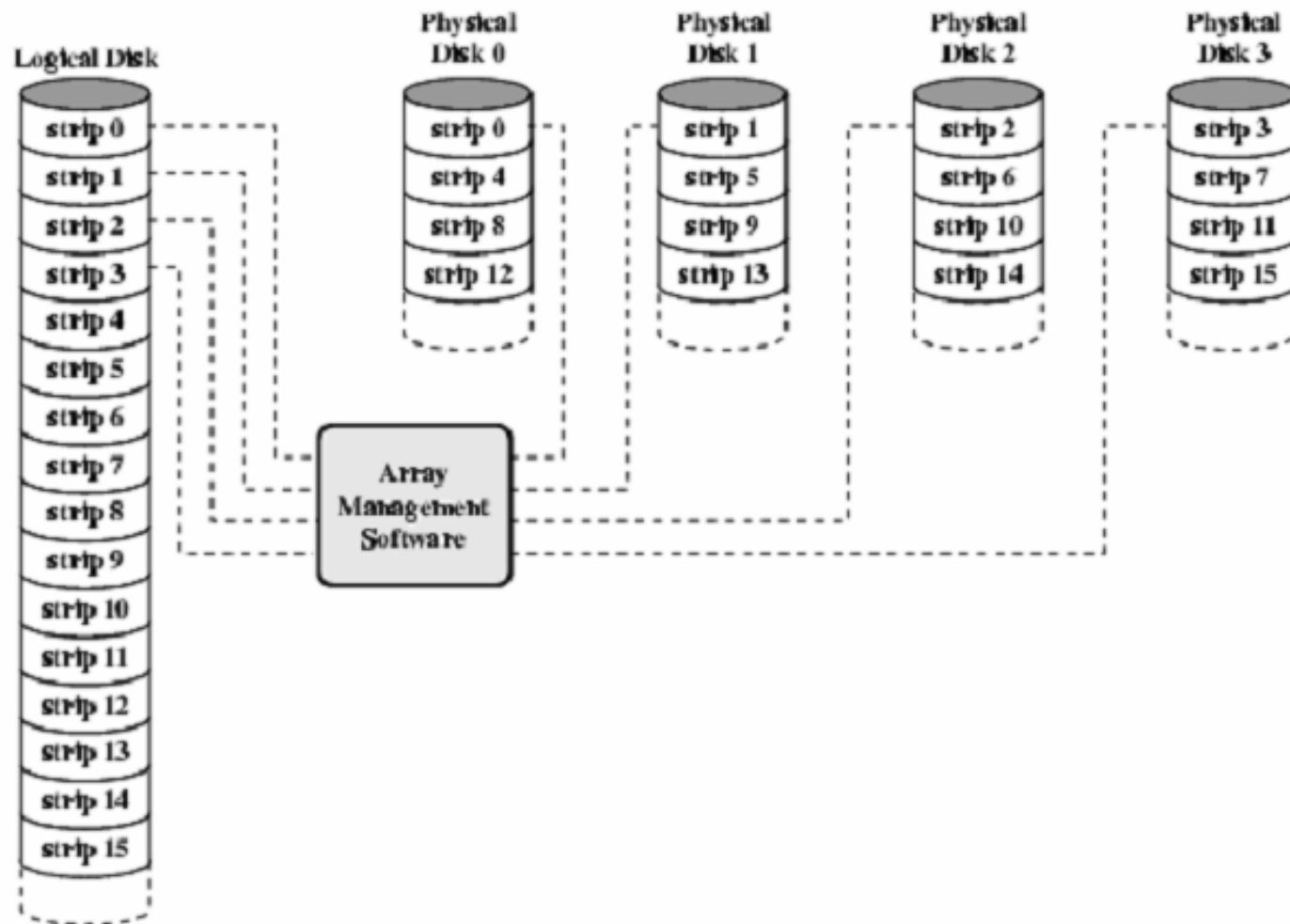
(f) RAID 5 (block-level distributed parity)



(g) RAID 6 (dual redundancy)

# Ảnh xạ dữ liệu của RAID 0

19



# Bộ nhớ trong

20

- Bộ nhớ chính

- Tồn tại dưới dạng các module nhớ DRAM (Dynamic Random Access Memory)

- Bộ nhớ đệm

- Tích hợp trên chip của CPU
- Sử dụng công nghệ lưu trữ SRAM (Static Random Access Memory)

# Phân loại RAM

21

SRAM (Static RAM)	DRAM (Dynamic RAM)
<ul style="list-style-type: none"><li>- Các bit được lưu trữ bằng các Flip-Flop → Thông tin ổn định</li><li>- Cấu trúc phức tạp</li><li>- Dung lượng chip nhỏ</li><li>- Tốc độ nhanh</li><li>- Đắt tiền</li><li>- Dùng làm bộ nhớ Cache</li></ul>	<ul style="list-style-type: none"><li>- Các bit được lưu trữ trên tụ điện → Cần phải có mạch refresh</li><li>- Cấu trúc đơn giản</li><li>- Dung lượng lớn</li><li>- Tốc độ chậm hơn</li><li>- Rẻ tiền hơn</li><li>- Dùng làm bộ nhớ chính</li></ul>

Technology	Typical access time	\$ per Mbyte in 1997
SRAM	5 - 25 ns	\$100 - \$250
DRAM	60 - 120 ns	\$5 - \$10
Magnetic disk	10 - 20 million ns	\$0.10 - \$0.20

# Bộ nhớ chính

22

- Chứa các chương trình đang thực hiện và các dữ liệu đang thao tác
- Tồn tại trên mọi hệ thống máy tính
- Bao gồm các ngăn nhớ được đánh địa chỉ trực tiếp bởi CPU
- Dung lượng của bộ nhớ chính < Không gian địa chỉ bộ nhớ mà CPU quản lý
- Sử dụng công nghệ lưu trữ DRAM

# Phân loại DRAM

23

- SIMM (Single Inline Memory Module): Cũ, chậm
- DIMM (Dual Inline Memory Module): Phổ biến
- RIMM (Rhombus Inline Memory Module): Mới, nhanh nhất

SIMM



DIMM



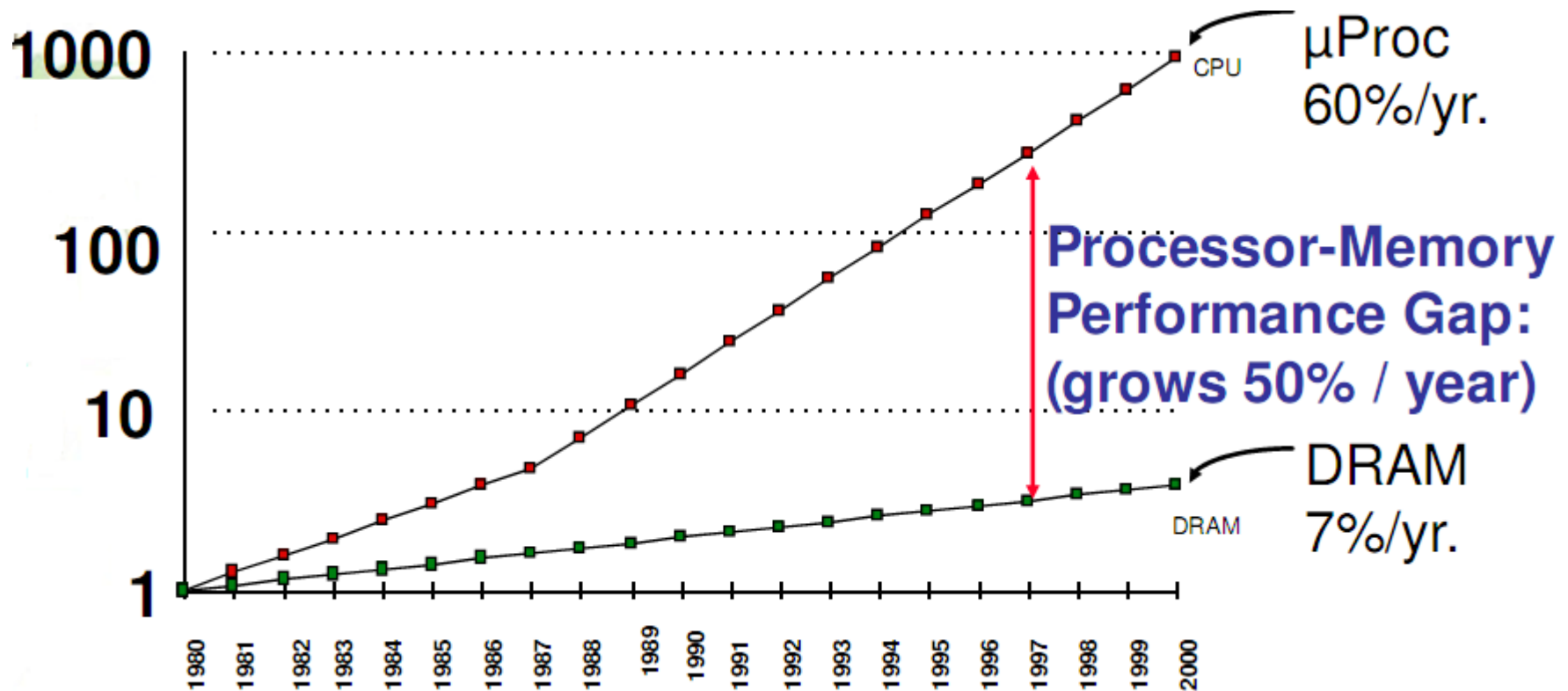
RIMM



# Bộ nhớ đệm

24

- Là loại bộ nhớ trung gian giữa CPU và bộ nhớ chính, có tác dụng làm giảm thời gian truy xuất bộ nhớ RAM





# Bộ nhớ đệm

25

- Khi cần đọc 1 ô nhớ từ bộ nhớ:
  - Kiểm tra xem có trong cache chưa?
    - Nếu chưa có (**cache miss**): chép ô nhớ đó và 1 số ô nhớ lân cận từ bộ nhớ chính vào cache
    - Nếu đã có (**cache hit**): đọc từ cache, không cần truy xuất bộ nhớ chính
- **Cache là bản copy một phần của bộ nhớ chính**
- Cache (dùng công nghệ SRAM) có tốc độ truy xuất cao hơn so với bộ nhớ chính (dùng công nghệ DRAM)

# Hai nguyên lý cơ sở khi truy xuất

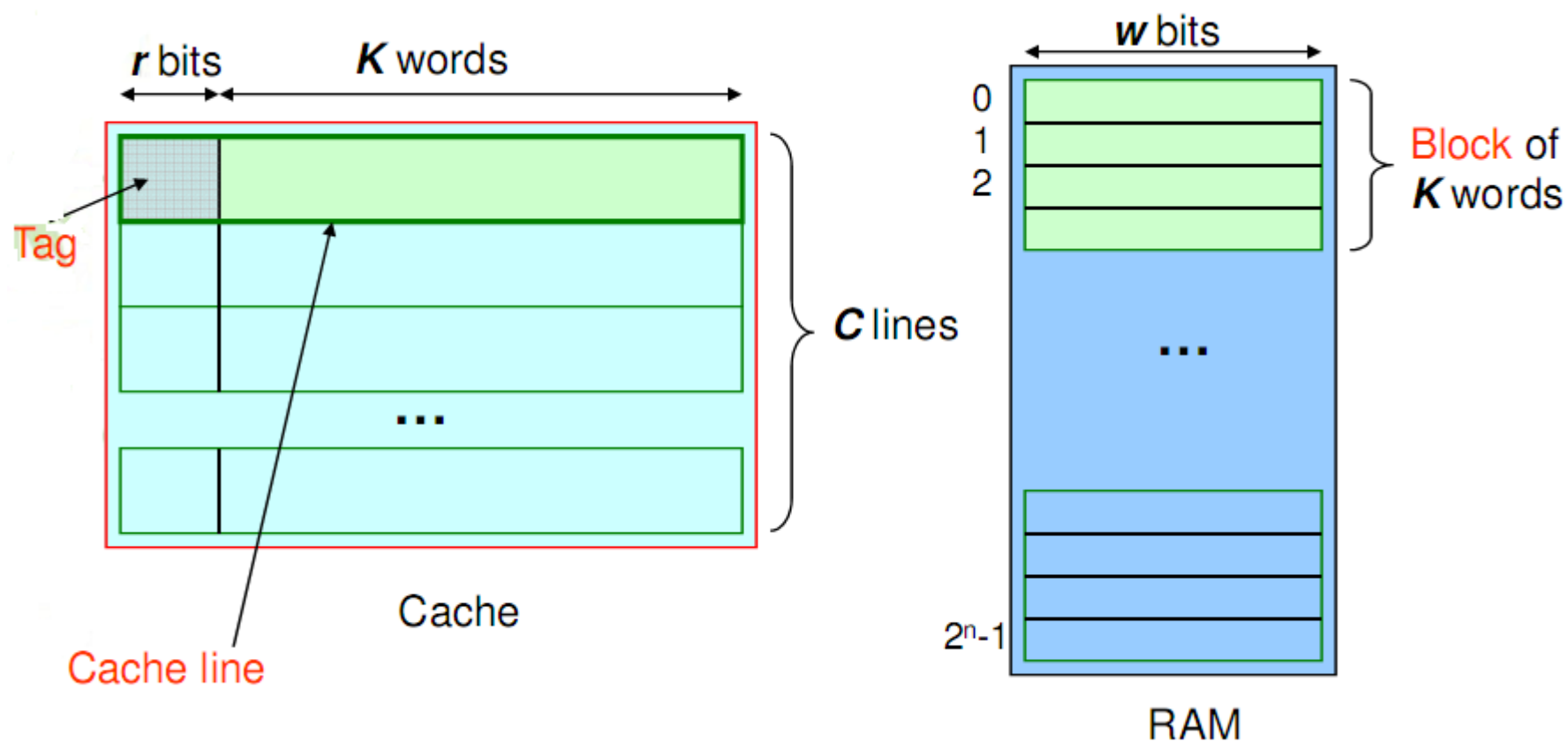
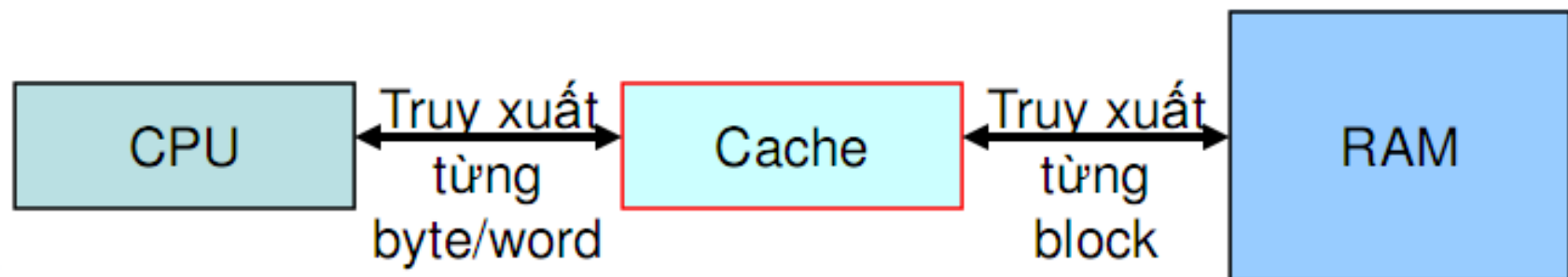
26

- **Temporal locality (Cục bộ về thời gian)**
  - Nếu một ô nhớ được dùng đến ở thời điểm hiện tại, nó dễ có khả năng được dùng đến lần nữa trong tương lai gần
- **Spatial locality (Cục bộ về không gian)**
  - Nếu một ô nhớ được dùng đến ở thời điểm hiện tại, những ô lân cận dễ có khả năng sắp được dùng đến

# Các vấn đề đặt ra

27

- Khi cần truy xuất 1 ô nhớ, làm sao biết ô nhớ đó đã có trong cache hay chưa? Nếu đã có thì ở chỗ nào trong cache?
- Những ô nhớ nào sẽ được lựa chọn để đưa vào cache? Việc lựa chọn xảy ra khi nào?



# Ý nghĩa

29

- Bộ nhớ chính có  $2^n$  byte nhớ, đánh số từ  $0 \rightarrow 2^n - 1$
- Bộ nhớ chính và Cache được chia thành thành các khối có kích thước bằng nhau
  - 1 Block của bộ nhớ chính = 1 Line của cache
- Một số Block của bộ nhớ chính được nạp vào các Line của cache
- Nội dung Tag (thẻ nhớ) cho biết Block nào của bộ nhớ chính hiện đang được chứa ở Line đó (chứ không phải số thứ tự của Line đó trong Cache)

# Các phương pháp ánh xạ

30

- **Direct mapping** (ánh xạ trực tiếp)
- **Associative mapping** (ánh xạ liên kết toàn phần)
- **Set associative mapping** (ánh xạ liên kết tập hợp)

# Direct mapping

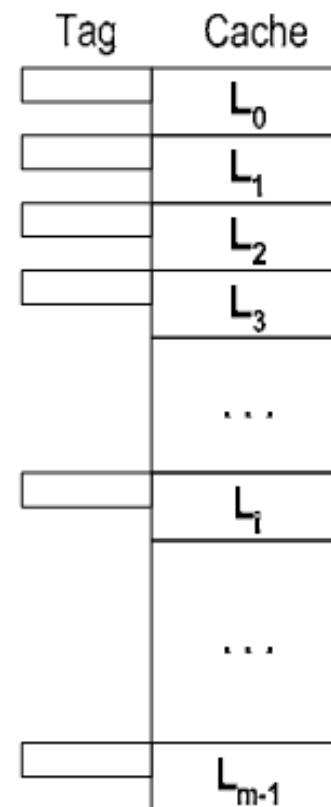
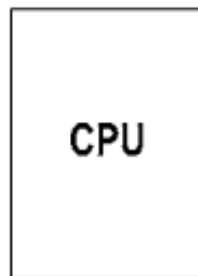
31

□ Mỗi Block của BNC chỉ có thể được nạp vào 1 Line của cache:

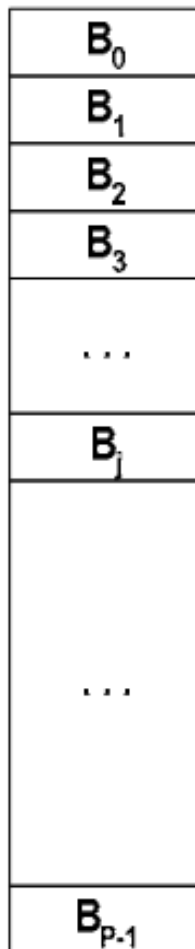
- $B_0 \rightarrow L_0$
- $B_1 \rightarrow L_1$
- ...
- $B_{m-1} \rightarrow L_{m-1}$
- $B_m \rightarrow L_0$
- $B_{m+1} \rightarrow L_1$
- ...

□ Tổng quát:

- $B_j$  chỉ có thể nạp vào  $L_{j \bmod m}$
- $m$  là số Line của cache



Bộ nhớ chính



# Direct mapping

32

- Mỗi một địa chỉ X trong bộ nhớ chính gồm **N bit** chia thành 3 trường:

- **Trường Word** gồm **W bit** xác định kích thước 1 từ nhớ (ô) trong 1 Block = 1 Line:

→ Kích thước của Block / Line =  $2^W$

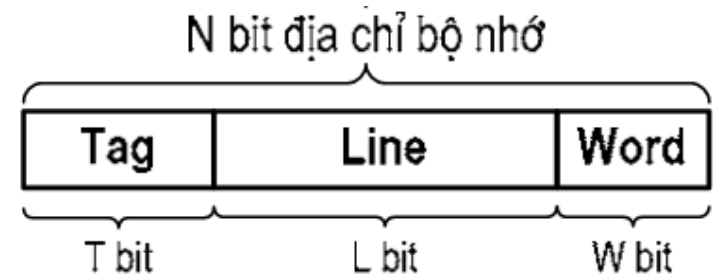
- **Trường Line** gồm **L bit** xác định địa chỉ 1 Line trong cache

→ Số Line trong cache =  $2^L$

- **Trường Tag** gồm **T bit**

→  $T = N - (W + L)$

- Xác định X có nằm trong Cache không (cache hit) hay vẫn đang nằm ở bộ nhớ





# Ví dụ

33

- Không gian địa chỉ bộ nhớ chính = 4 GB
  - Dung lượng cache = 256 KB
  - Kích thước 1 Line = 1 Block = 32 byte
- Xác định cụ thể số bit cho 3 trường địa chỉ của X (W, L, T) nếu tổ chức theo kiểu direct mapping

# Đáp án

34

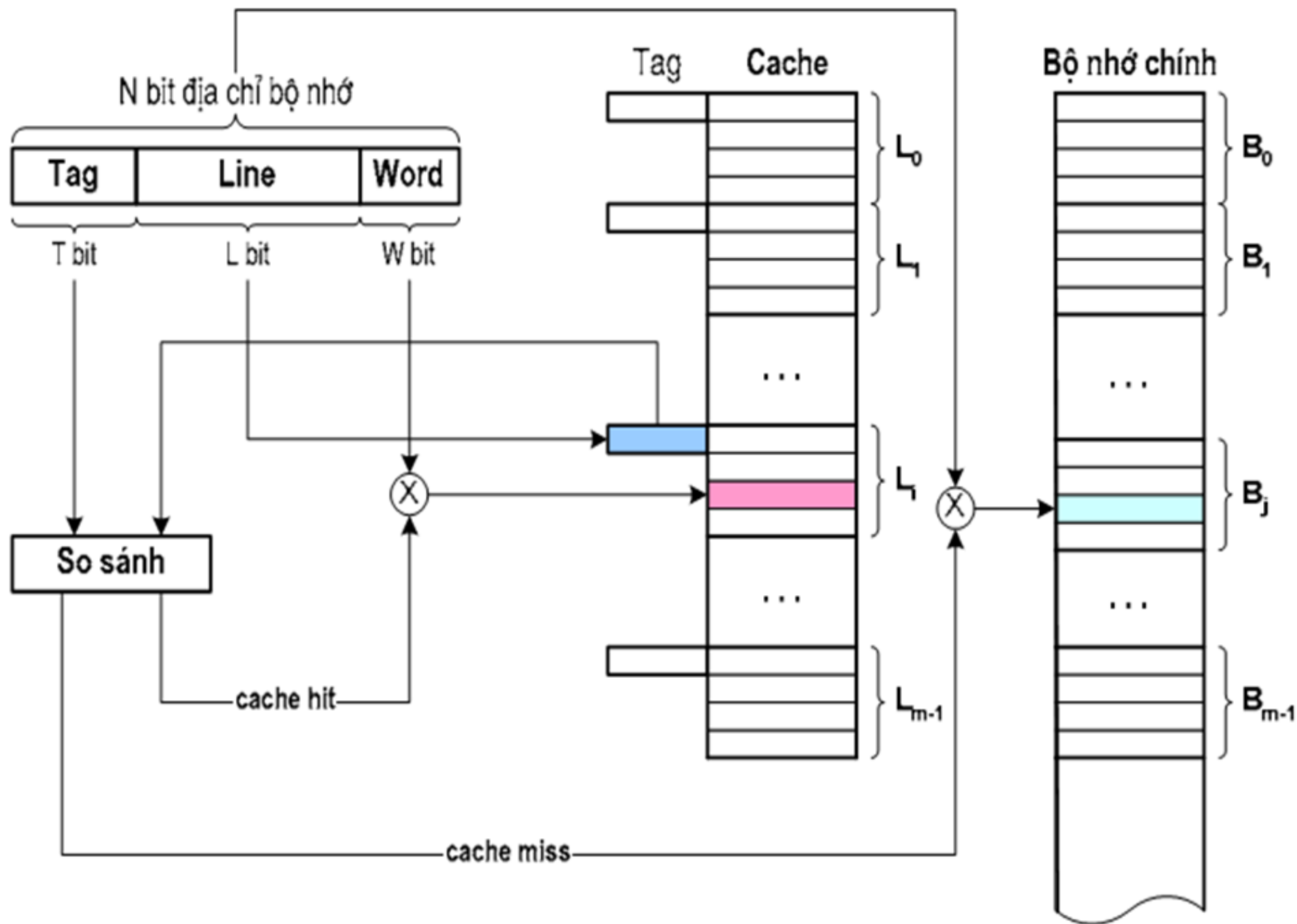
- Bộ nhớ chính = 4 GB =  $2^{32}$  byte  $\rightarrow$  **N = 32 bit**
  - Cache = 256 KB =  $2^{18}$  byte
- $\rightarrow$  Ta có thể dùng 18 bit để đánh địa chỉ từng từ nhớ (ô) trong Cache
- Line (bao gồm nhiều từ nhớ) = 32 byte =  $2^5$  byte  $\rightarrow$  **W = 5 bit**  
(Dùng 5 bit để đánh địa chỉ nội bộ các từ nhớ (ô) trong 1 Line)
- $\rightarrow$  Số Line trong cache =  $2^{18} / 2^5 = 2^{13}$  Line
- $\rightarrow$  **L = 13 bit** (Dùng 13 bit để đánh địa chỉ từng Line trong Cache)
- Tag = **T = N - (L + W) = 32 - (13 + 5) = 14 bit**

Tag 14 bit	Line 13 bit	Word 5 bit
---------------	----------------	---------------

# Nhận xét

35

- Ta có thể suy ra tổng số Block trong bộ nhớ chính
  - = Kích thước bộ nhớ chính / Kích thước 1 block
  - =  $2^{32} / 2^5 = 2^{27}$
- Dùng 27 bit để đánh địa chỉ 1 Block (= 14 + 13)
- Giả sử ta có **Block thứ M (27 bit**, giá trị từ 0 →  $2^{27} - 1$ ) muốn lưu vào cache thì sẽ lưu ở:
  - **Line thứ:  $L = M \% \text{Số Line trong cache} = M \% 2^{13}$  (13 bit)**
  - **Tag tại Line đó:  $T = M / \text{Số Line trong cache} = M / 2^{13}$  (14 bit)**



# Đánh giá Direct mapping

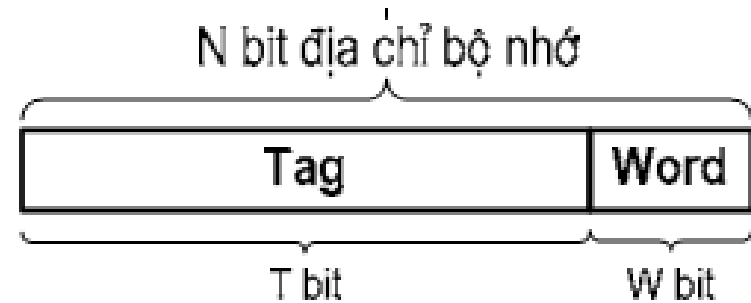
37

- Bộ so sánh đơn giản
  - Xác suất cache hit thấp
  - Giả sử muốn truy xuất đồng thời từ nhớ (ô) X tại Block thứ 0 và ô thứ Y tại Block thứ  $2^L$  thì sao?  
(L: Tổng số Line trong Cache)
- Bị xung đột thì cả 2 ô này đều sẽ được lưu ở Line thứ 0
- $(0 \% 2^L = 2^L \% 2^L = 0)$

# Associative mapping

38

- Mỗi Block có thể nạp vào bất kỳ Line nào của Cache
- Địa chỉ của bộ nhớ chính bao gồm 2 trường
  - ▣ Trường **Word** giống như trường hợp Direct Mapping
  - ▣ Trường **Tag** dùng để xác định số thứ tự Block của bộ nhớ chính được lưu ở Cache
- Tag xác định Block nào trong bộ nhớ chính đang nằm ở Line đó



# Ví dụ

39

- Không gian địa chỉ bộ nhớ chính = 4 GB
  - Kích thước 1 Line = 1 Block = 32 byte
- Xác định cụ thể số bit cho 2 trường địa chỉ của  $X(W, T)$  nếu tổ chức theo kiểu associative mapping

# Đáp án

40

- Bộ nhớ chính = 4 GB =  $2^{32}$  byte  $\rightarrow$  **N = 32 bit**
- Line (bao gồm nhiều từ nhớ) = 32 byte =  $2^5$  byte  $\rightarrow$  **W = 5 bit** (Dùng 5 bit để đánh địa chỉ nội bộ các từ nhớ (ô) trong 1 Line)
- Tag = **T = N - W = 32 - 5 = 27 bit**

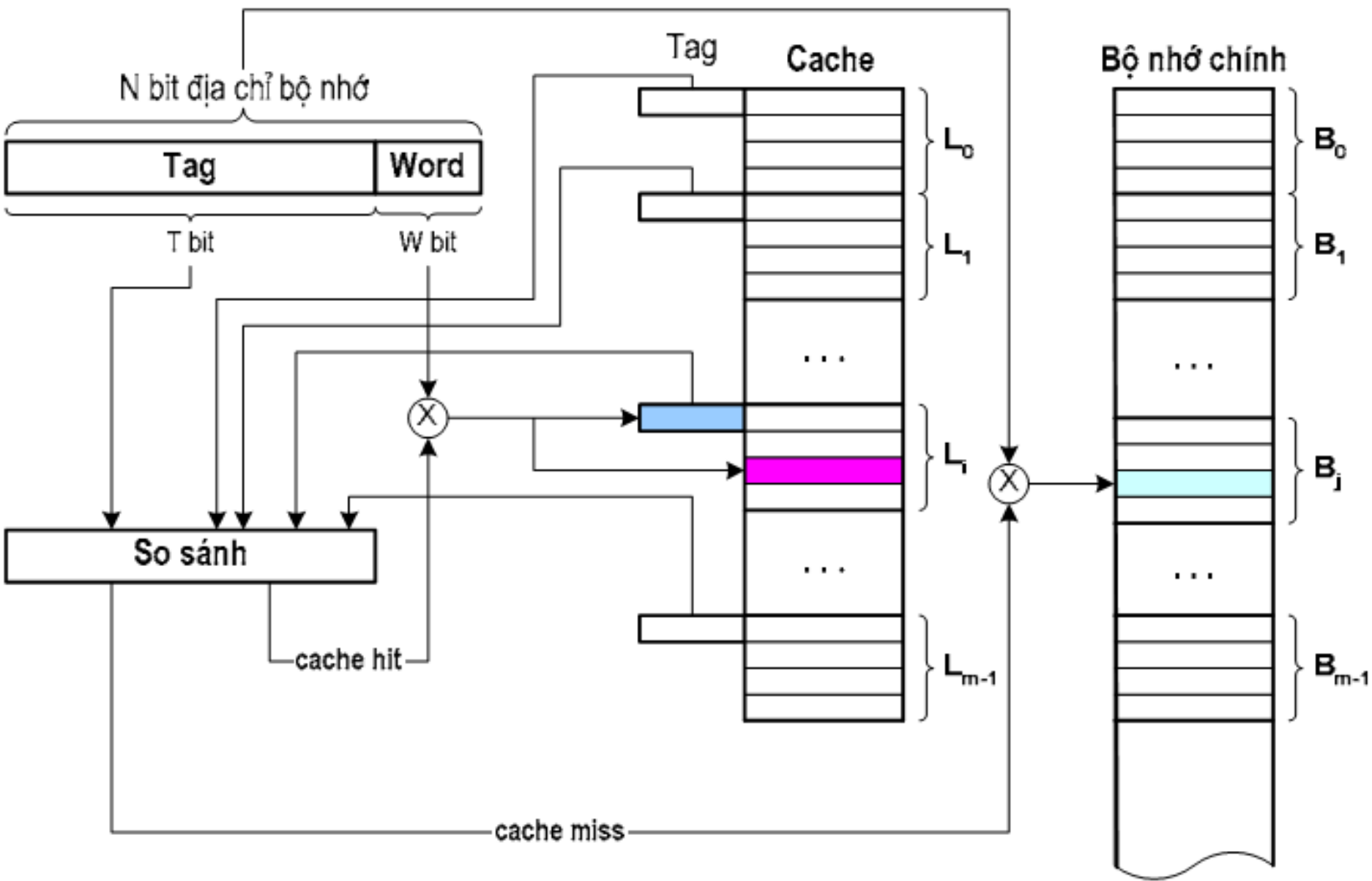
Tag 27 bit	Word 5 bit
---------------	---------------



# Nhận xét

41

- Ta có thể suy ra tổng số Block trong bộ nhớ chính
  - = Kích thước bộ nhớ chính / Kích thước 1 block
  - =  $2^{32} / 2^5 = 2^{27}$
- Dùng 27 bit để đánh địa chỉ 1 Block (= 14 + 13)
- Giả sử ta có **Block thứ M (27 bit)**, giá trị từ 0 →  $2^{27}$ 
  - 1) muốn lưu vào cache thì sẽ lưu ở bất kỳ Line nào miễn sao có Tag tại Line đó là:
    - ▣ **T = M (27 bit)**



# Đánh giá Associative mapping

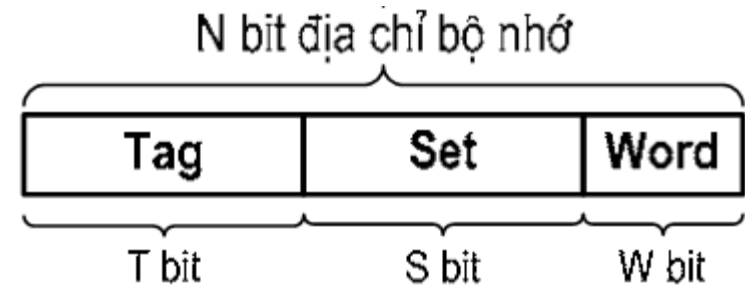
43

- ❑ Để tìm ra Line chứa nội dung của 1 Block, cần dò tìm và so sánh lần lượt với Tag của tất cả các Line của Cache
- Mất nhiều thời gian
- ❑ Xác suất cache hit cao
- ❑ Cần bộ so sánh phức tạp

# Set associative mapping

44

- ❑ Cache được chia thành các Tập (Set)
- ❑ Mỗi một Set chứa 1 số Line (2,4,8,16 Line)
  - Ví dụ: 4 Line / Set → 4-way associative mapping
- ❑ Ánh xạ theo nguyên tắc sau:
  - $B_0 \rightarrow S_0$
  - $B_1 \rightarrow S_1$
  - $B_2 \rightarrow S_2$
  - ...
- ❑ Địa chỉ của bộ nhớ chính bao gồm 3 trường
  - Trường **Word** xác định kích thước 1 Block (= 1 Line)
  - Trường **Set** xác định thứ tự Set trong Cache
  - Trường **Tag** dùng để xác định số thứ tự Block của bộ nhớ chính được lưu ở Cache



# Ví dụ

45

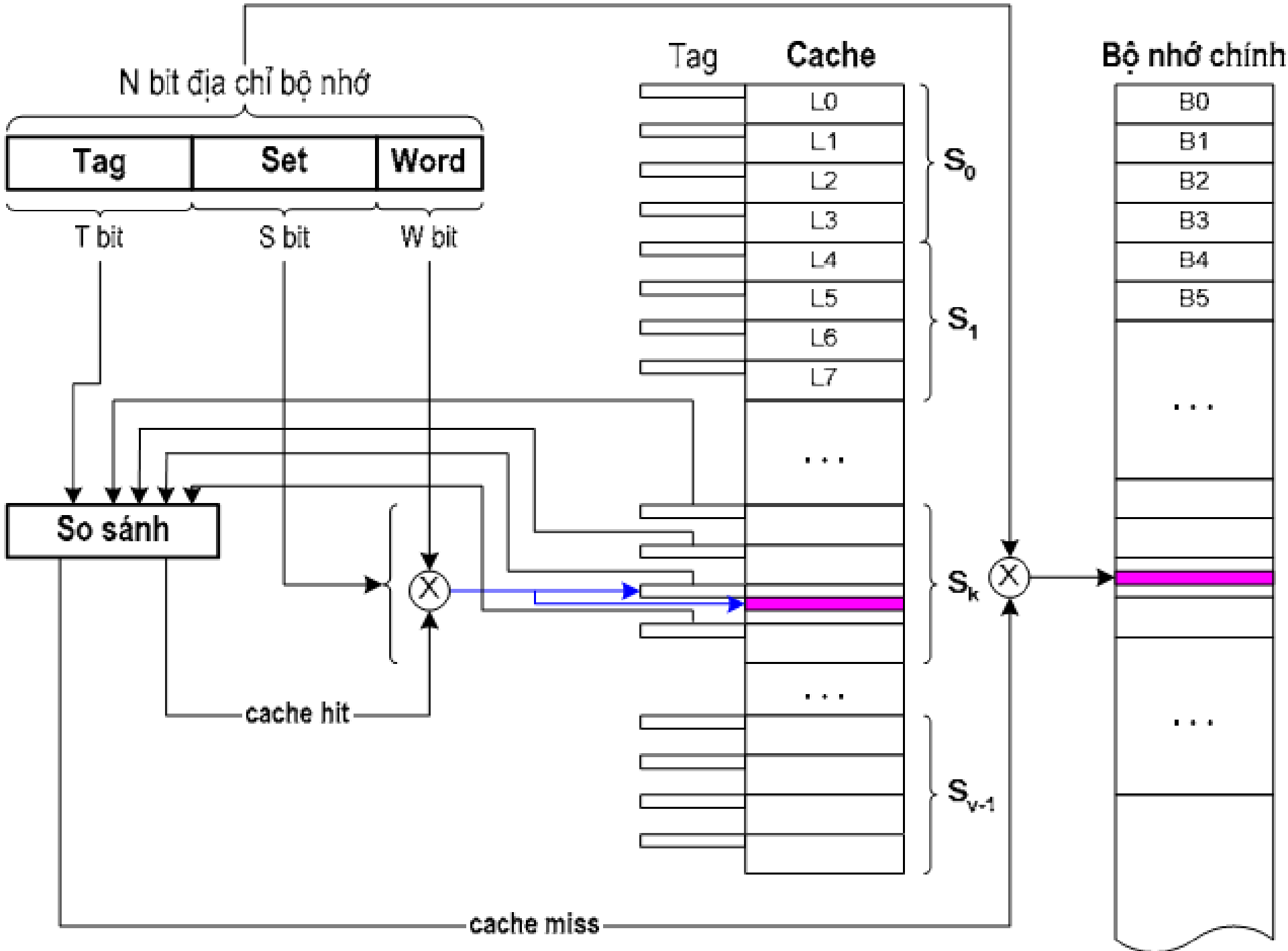
- Không gian địa chỉ bộ nhớ chính = 4 GB
  - Dung lượng cache = 256 KB
  - Kích thước 1 Line = 1 Block = 32 byte
- Xác định cụ thể số bit cho 3 trường địa chỉ của X (W, S, T) nếu tổ chức theo kiểu 4-way associative mapping

# Đáp án

Tag	Set	Word
16 bit	11 bit	5 bit

46

- Bộ nhớ chính = 4 GB =  $2^{32}$  byte → **N = 32 bit**
- Cache = 256 KB =  $2^{18}$  byte
  - Ta có thể dùng 18 bit để đánh địa chỉ từng từ nhớ (ô) trong Cache
- Line (bao gồm nhiều từ nhớ) = 32 byte =  $2^5$  byte → **W = 5 bit** (Dùng 5 bit để đánh địa chỉ nội bộ các từ nhớ (ô) trong 1 Line)
  - Số Line trong cache =  $2^{18} / 2^5 = 2^{13}$  Line
  - **L = 13 bit** (Dùng 13 bit để đánh địa chỉ từng Line trong Cache)
- Một Set trong Cache có 4 Line =  $2^2$  Line
  - Số Set trong Cache =  $2^{13} / 2^2 = 2^{11}$  Set → **S = 11 bit** (Dùng 11 bit để địa chỉ các Set trong Cache)
- Tag = **T = N - (S + W) = 32 - (11 + 5) = 16 bit**



# Các tham số ảnh hưởng hiệu suất Cache

48

## □ Block size

- **Nhỏ quá:** giảm tính lân cận (spatial locality)
- **Lớn quá:** số lượng block trong cache ít, thời gian chuyển block vào cache lâu (miss penalty)

## □ Cache size

- **Nhỏ quá:** số lượng Block có thể lưu trong cache quá ít, làm tăng tỷ lệ cache miss
- **Lớn quá:** tỷ lệ giữa vùng nhớ thực sự cần thiết so với vùng nhớ lưu vào cache sẽ thấp, nghĩa là overhead (tổng chi phí) sẽ cao, tốc độ truy cập cache giảm



# Thuật toán thay thế (Replacement Algorithm)

49

- Khi cần chuyển 1 Block mới vào trong Cache mà không tìm được Line trống, vậy phải bỏ Line nào ra?
- Một số cách chọn:
  - **Random:** Thay thế ngẫu nhiên
  - **FIFO (First In First Out):** Thay thế Line nào nằm lâu nhất trong Cache
  - **LFU (Least Frequently Used):** Thay thế Line nào trong Cache có số lần truy cập ít nhất trong cùng 1 khoảng thời gian
  - **LRU (Least Recently Used):** Thay thế Line nào trong Cache có thời gian lâu nhất không được tham chiếu đến
- Tối ưu nhất: LRU

# Write Policy

50

- Nếu 1 Line bị thay đổi trong Cache, khi nào sẽ thực hiện thao tác ghi lên lại RAM ?
  - ▣ **Write Through:** ngay lập tức
  - ▣ **Write Back:** khi Line này bị thay thế
- Nếu nhiều processor chia sẻ RAM, mỗi processor có cache riêng:
  - ▣ **Bus watching with WT:** loại bỏ Line khi bị thay đổi trong 1 cache khác
  - ▣ **Hardware transparency:** tự động cập nhật các cache khác khi Line bị 1 cache thay đổi
  - ▣ **Noncacheable shared memory:** phần bộ nhớ dùng chung sẽ không được đưa vào cache

# Số lượng và Loại cache

51

- Có thể sử dụng nhiều mức cache (gọi là **level**):  
L1, L2, L3...
- Các cache ở **mức thấp** gọi có thể là **on-chip**, trong khi cache **mức cao** thường là **off-chip** và được truy cập thông qua external bus hoặc bus dành riêng
- Cache có thể dùng chung cho cả data và instruction hoặc riêng cho từng loại

# Cache trên các bộ xử lý Intel

52

- **80486:** 8 KB cache L1 trên chip (on-chip)
- **Pentium:** có 2 cache L1 trên chip
  - Cache lệnh: 8 KB
  - Cache dữ liệu: 8 KB
- **Pentium 4 (2000):** có 2 level cache L1 và L2 trên chip
  - **Cache L1:**
    - 2 cache, mỗi cache 8 KB
    - Kích thước Line = 64 byte
    - 4-way associative mapping
  - **Cache L2:**
    - 256 KB
    - Kích thước Line = 128 byte

# Sơ đồ bộ nhớ Pentium 4

53

