



**KHOA CÔNG NGHỆ THÔNG TIN  
BỘ MÔN MẠNG VÀ CÁC HỆ THỐNG THÔNG TIN**

## **CHƯƠNG 4**

# **Bộ nhớ**

Kiến trúc và Tổ chức máy tính

# Mục tiêu

Sau khi hoàn thành chương này, sinh viên có khả năng:

- Trình bày được nhiệm vụ và cấu tạo cơ bản của bộ nhớ máy tính.
- Mô tả được nguyên lý hoạt động của CPU.
- Phân tích được kiến trúc của các bộ xử lý tiên tiến.
- Giải thích được tập lệnh cơ bản của bộ xử lý 8086.

# Nội dung

4.1. Tổng quan về hệ thống nhớ

4.2. Bộ nhớ bán dẫn

4.3. Bộ nhớ chính

4.4. Thiết kế bộ nhớ

4.5. Bộ nhớ đệm nhanh

# 4.1. Tổng quan về hệ thống nhớ

## 1. Các đặc trưng của bộ nhớ

### ■ Vị trí

#### ■ Bên trong CPU:

- tập thanh ghi

#### ■ Bộ nhớ trong:

- bộ nhớ chính
- bộ nhớ đệm (cache)

#### ■ Bộ nhớ ngoài:

- các thiết bị lưu trữ

### ■ Dung lượng

#### ■ Độ dài từ nhớ (tính bằng bit)

#### ■ Số lượng từ nhớ

# Các đặc trưng của bộ nhớ (tiếp)

- Đơn vị truyền
  - Từ nhớ
  - Khối nhớ
- Phương pháp truy nhập
  - Truy nhập tuần tự (băng từ)
  - Truy nhập trực tiếp (các loại đĩa)
  - Truy nhập ngẫu nhiên (bộ nhớ bán dẫn)
  - Truy nhập liên kết (cache)

# Các đặc trưng của bộ nhớ (tiếp)

- Hiệu năng (performance)
  - Thời gian truy nhập
  - Chu kỳ nhớ
  - Tốc độ truyền
- Kiểu vật lý
  - Bộ nhớ bán dẫn
  - Bộ nhớ từ
  - Bộ nhớ quang

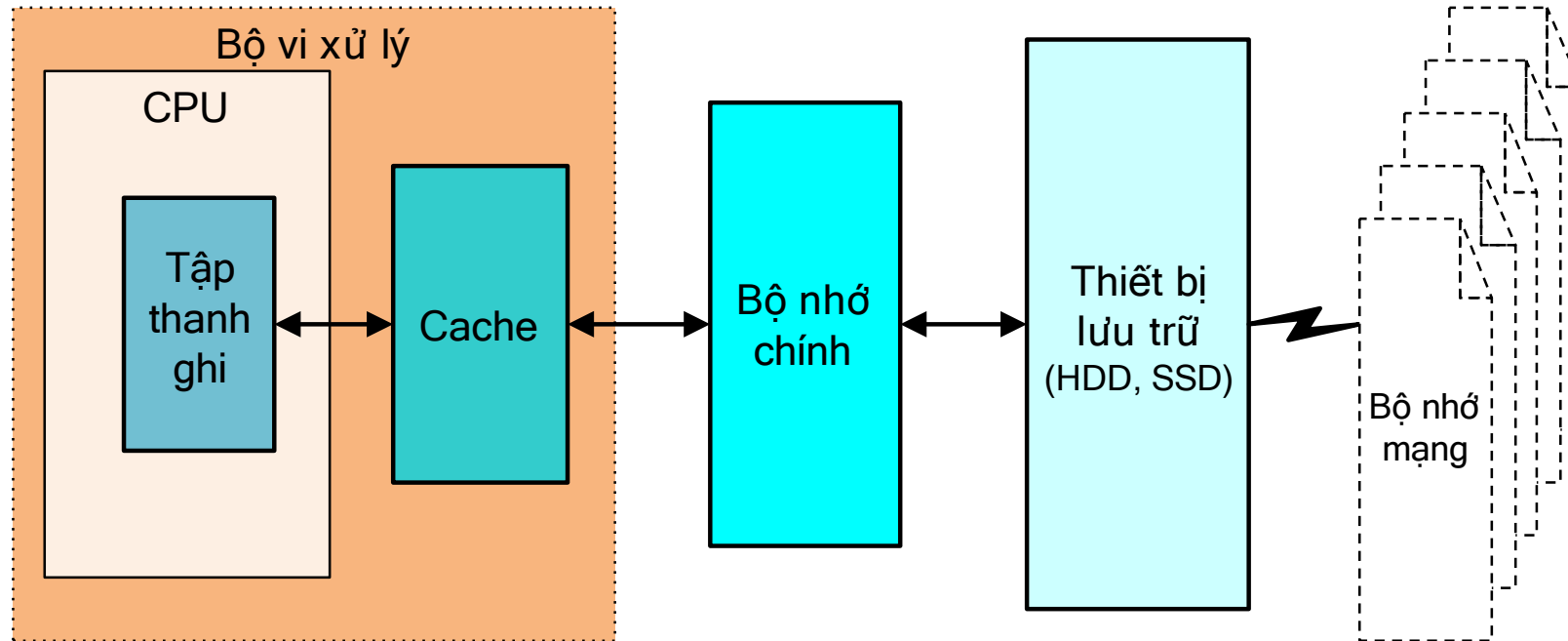
# Các đặc trưng của bộ nhớ (tiếp)

- Các đặc tính vật lý
  - Khả biến / Không khả biến (volatile / nonvolatile)
  - Xoá được / không xoá được
- Tổ chức

# Các đặc trưng của bộ nhớ (tiếp)

- Các đặc tính vật lý
  - Khả biến / Không khả biến (volatile / nonvolatile)
  - Xoá được / không xoá được
- Tổ chức

## 2. Phân cấp bộ nhớ



Từ trái sang phải:

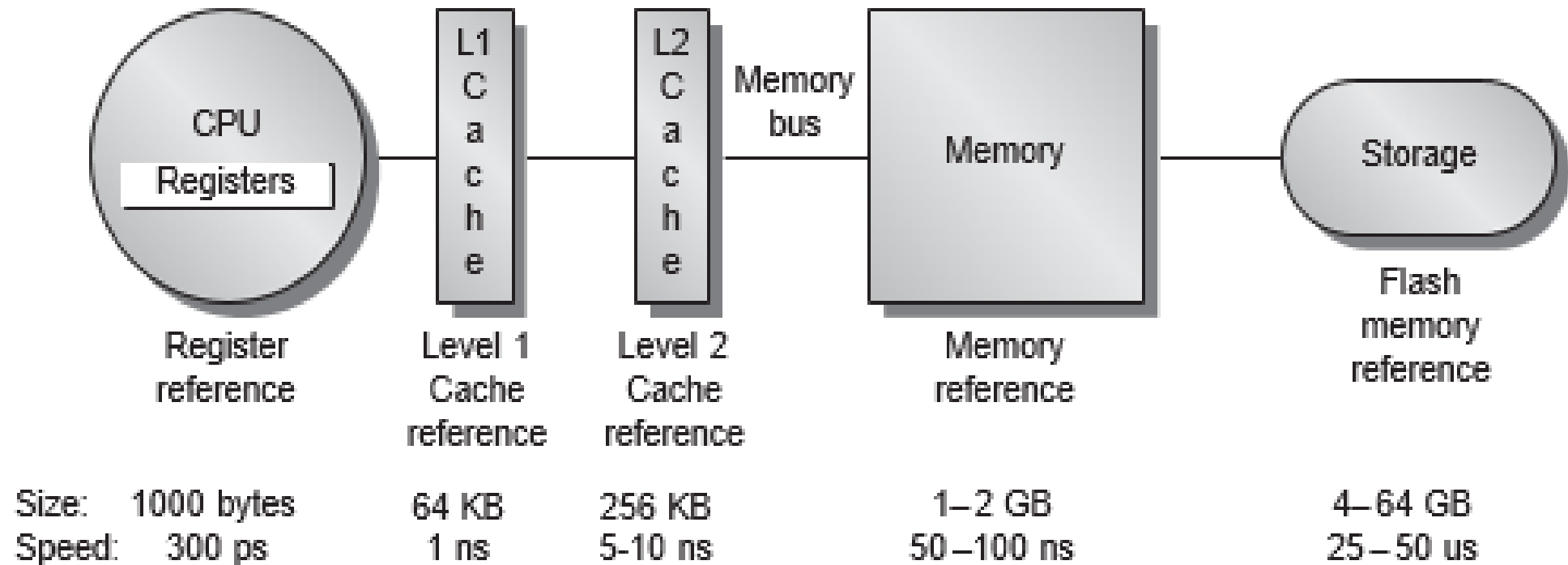
- dung lượng tăng dần
- tốc độ giảm dần
- giá thành cùng dung lượng giảm dần

# Công nghệ bộ nhớ

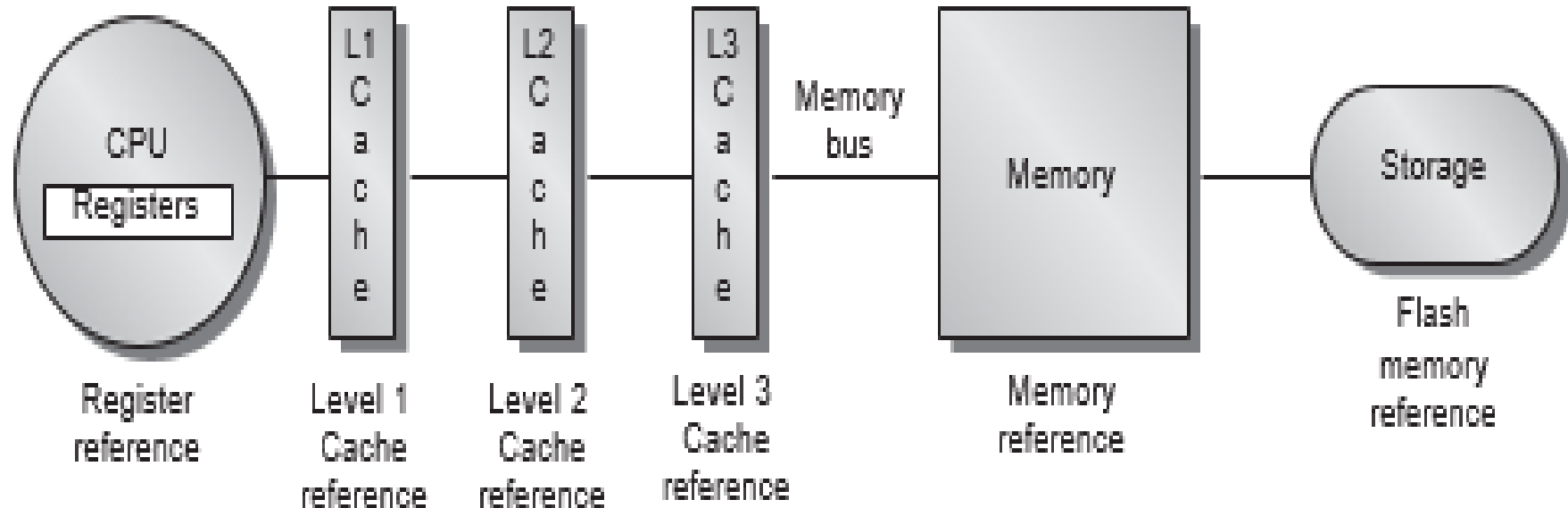
Công nghệ bộ nhớ	Thời gian truy nhập	Giá thành/GiB (2025)
SRAM	0,5 – 2,5 ns	\$500 – \$1000
DRAM	50 – 70 ns	\$10 – \$20
Flash memory	5000 – 50 000 ns	\$0,75 – \$1
HDD	5 – 20 ms	\$0,05 – \$0,1
SSD	30–80 $\mu$ s	\$0,11 – \$0,3

- Bộ nhớ lý tưởng
  - Thời gian truy nhập như SRAM
  - Dung lượng và giá thành như ổ đĩa cứng

# Phân cấp bộ nhớ cho thiết bị di động

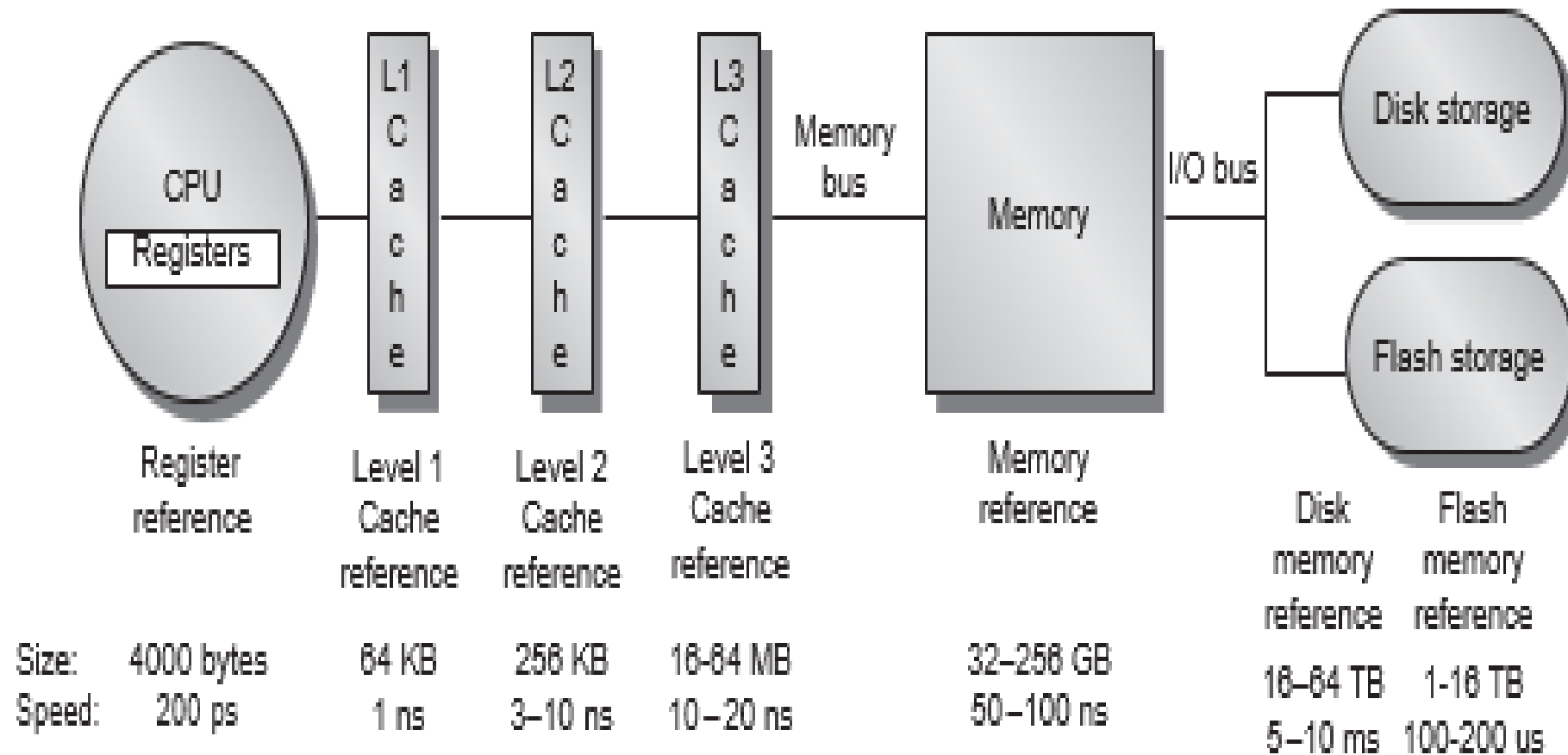


# Phân cấp bộ nhớ cho máy tính PC



Laptop	Size:	1000 bytes	64 KB	256 KB	4-8 MB	4-16 GB	256 GB-1 TB
	Speed:	300 ps	1 ns	3-10 ns	10-20 ns	50-100 ns	50-100 μs
Desktop	Size:	2000 bytes	64 KB	256 KB	8-32 MB	8-64 GB	256 GB-2 TB
	Speed:	300 ps	1 ns	3-10 ns	10-20 ns	50-100 ns	50-100 μs

# Phân cấp bộ nhớ cho máy chủ



## 4.2 Bộ nhớ bán dẫn

Kiểu bộ nhớ	Tiêu chuẩn	Khả năng xoá	Cơ chế ghi	Tính khả biến
Read Only Memory (ROM)	Bộ nhớ chỉ đọc	Không xoá được	Mặt nạ	Không khả biến
Programmable ROM (PROM)			Bảng điện	
Erasable PROM (EPROM)	bằng tia cực tím, cả chip			
Electrically Erasable PROM (EEPROM)	bằng điện, mức từng byte			
Flash memory	Bộ nhớ đọc-ghi	bằng điện, từng khối	Bảng điện	Khả biến
Random Access Memory (RAM)		bằng điện, mức từng byte		

# ROM (Read Only Memory)

- Bộ nhớ không khả biến
- Lưu trữ các thông tin sau:
  - Thư viện các chương trình con
  - Các chương trình điều khiển hệ thống (BIOS)
  - Các bảng chức năng
  - Vi chương trình

# Các kiểu ROM

- ROM mặt nạ:
  - thông tin được ghi khi sản xuất
- PROM (Programmable ROM)
  - Cần thiết bị chuyên dụng để ghi
  - Chỉ ghi được một lần
- EPROM (Erasable PROM)
  - Cần thiết bị chuyên dụng để ghi
  - Xóa được bằng tia tử ngoại
  - Ghi lại được nhiều lần
- EEPROM (Electrically Erasable PROM)
  - Có thể ghi theo từng byte
  - Xóa bằng điện

# Bộ nhớ Flash

- Ghi theo khối
- Xóa bằng điện
- Dung lượng lớn

# RAM (Random Access Memory)

- Bộ nhớ đọc-ghi (Read/Write Memory)
- Khả biến
- Lưu trữ thông tin tạm thời
- Có hai loại: SRAM và DRAM  
(Static and Dynamic RAM)

# SRAM (Static RAM) – RAM tĩnh

- Các bit được lưu trữ bằng các Flip-Flop  
→ thông tin ổn định
- Cấu trúc phức tạp
- Dung lượng chip nhỏ
- Tốc độ nhanh
- Đắt tiền
- Dùng làm bộ nhớ cache

# DRAM (Dynamic RAM) – RAM động

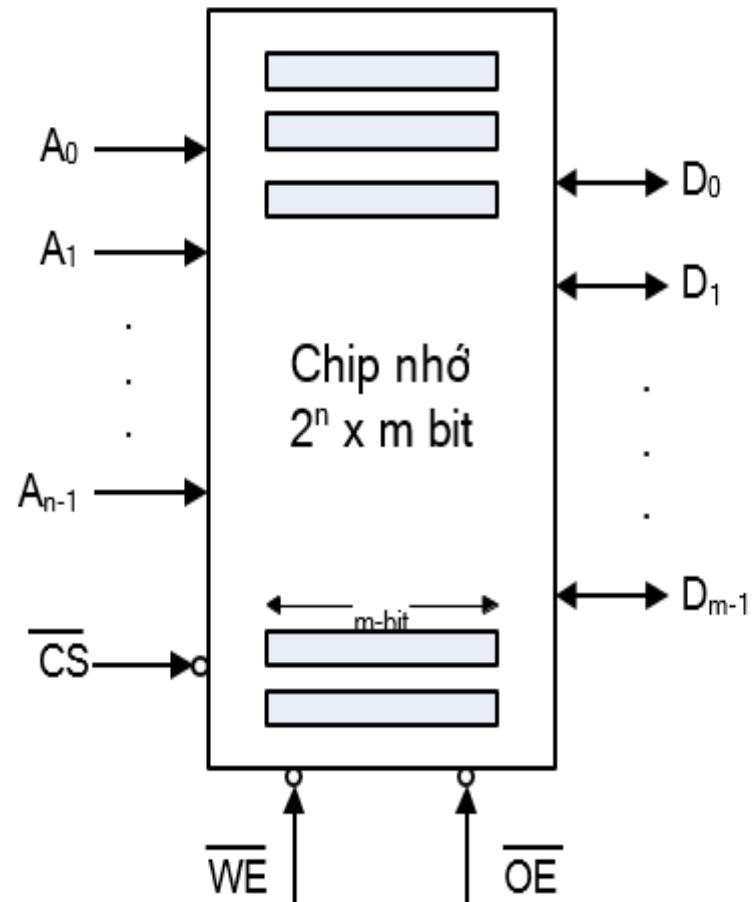
- Các bit được lưu trữ trên tụ điện  
→ cần phải có mạch làm tươi
- Cấu trúc đơn giản
- Dung lượng lớn
- Tốc độ chậm hơn
- Rẻ tiền hơn
- Dùng làm bộ nhớ chính

# Một số DRAM tiên tiến thông dụng

- Cải tiến để tăng tốc độ
- Synchronous DRAM (SDRAM): làm việc được đồng bộ bởi xung clock
- DDR-SDRAM (Double Data Rate SDRAM)
- DDR3, DDR4, DDR5

# Tổ chức của chip nhớ

## ■ Sơ đồ cơ bản của chip nhớ



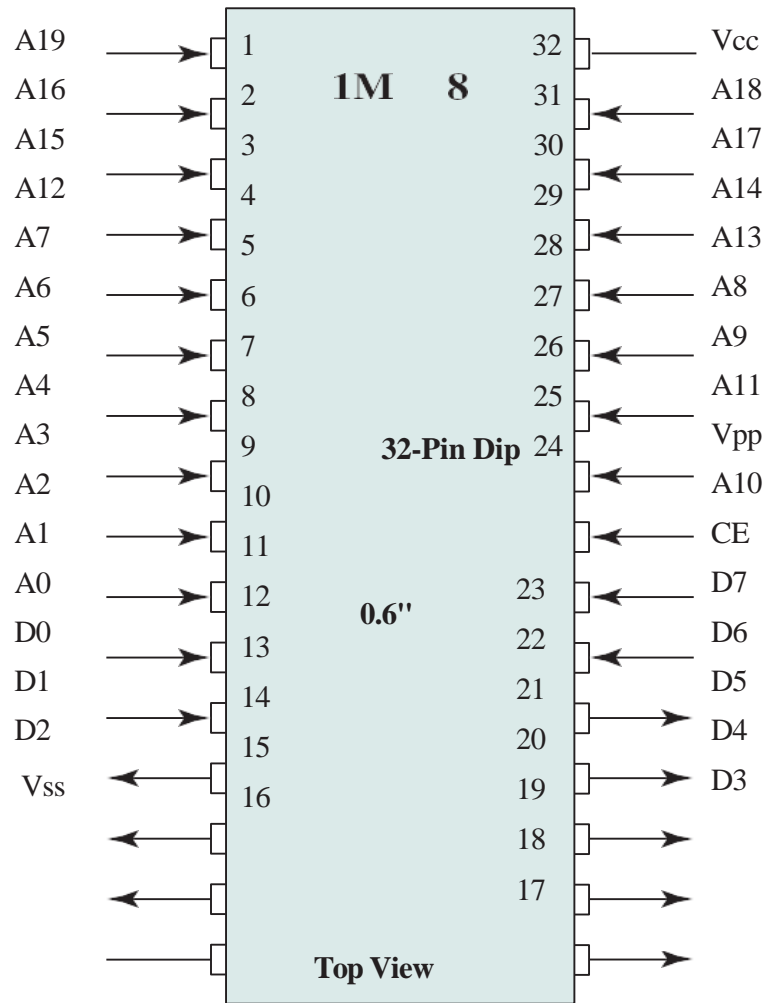
# Các tín hiệu của chip nhớ

- Các đường địa chỉ:  $A_{n-1} \div A_0 \rightarrow$  có  $2^n$  từ nhớ
- Các đường dữ liệu:  $D_{m-1} \div D_0 \rightarrow$  độ dài từ nhớ =  $m$  bit
- Dung lượng chip nhớ =  $2^n \times m$  bit
- Các đường điều khiển:
  - Tín hiệu chọn chip CS (Chip Select)
  - Tín hiệu điều khiển đọc OE (Output Enable)
  - Tín hiệu điều khiển ghi WE (Write Enable)(Các tín hiệu điều khiển thường tích cực với mức 0)

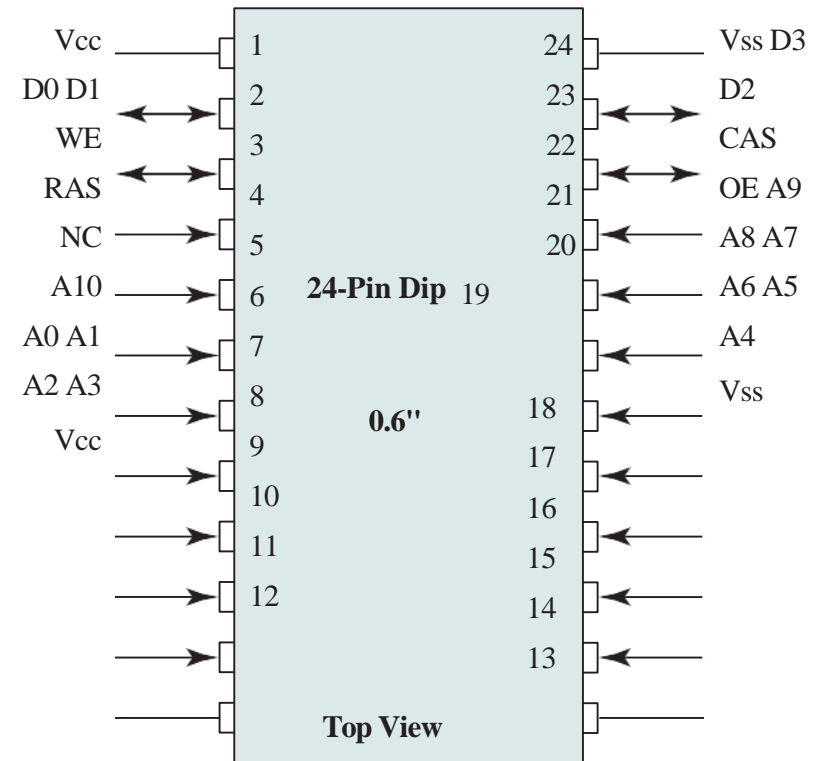
# Tổ chức của DRAM

- Dùng n đường địa chỉ dẫn kênh → cho phép truyền  $2^n$  bit địa chỉ
- Tín hiệu chọn địa chỉ hàng RAS (Row Address Select)
- Tín hiệu chọn địa chỉ cột CAS (Column Address Select)
- Dung lượng của DRAM =  $2^{2n} \times m$  bit

# Ví dụ chip nhớ



(a) 8-Mbit EPROM



(b) 16-Mbit DRAM

## 4.4 Thiết kế bộ nhớ

# Thiết kế mô-đun nhớ bán dẫn

- Dung lượng chip nhớ  $2^n \times m$  bit
- Cần thiết kế để tăng dung lượng:
  - Thiết kế tăng độ dài từ nhớ
  - Thiết kế tăng số lượng từ nhớ
  - Thiết kế kết hợp

# Tăng độ dài từ nhớ

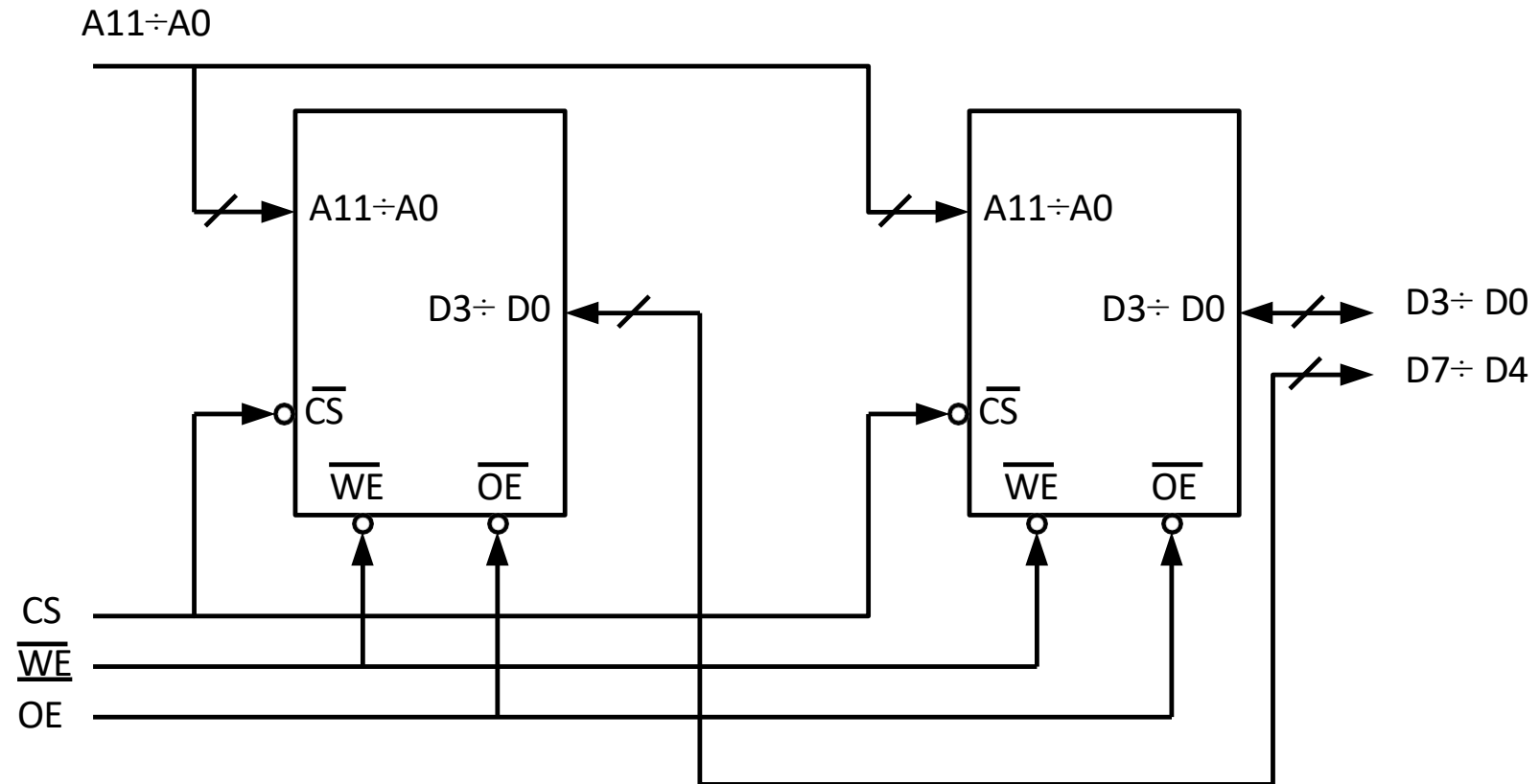
## Ví dụ 1:

- Cho chip nhớ SRAM 4K x 4 bit
- Thiết kế mô-đun nhớ 4K x 8 bit **Giải:**
- Dung lượng chip nhớ 4K x 4 bit =  $2^2 \times 2^{10} \times 4\text{bit} = 2^{12} \times 4\text{ bit}$
- chip nhớ có:
  - 12 chân địa chỉ
  - 4 chân dữ liệu
- mô-đun nhớ cần có:
  - 12 chân địa chỉ
  - 8 chân dữ liệu

# Bài toán tăng độ dài từ nhớ tổng quát

- Cho chip nhớ  $2^n \times \text{mbit}$
- Thiết kế mô-đun nhớ  $2^n \times (k.m)$  bit
- Dùng  $k$  chip nhớ

# Sơ đồ ví dụ tăng độ dài từ nhớ



# Tăng số lượng từ nhớ

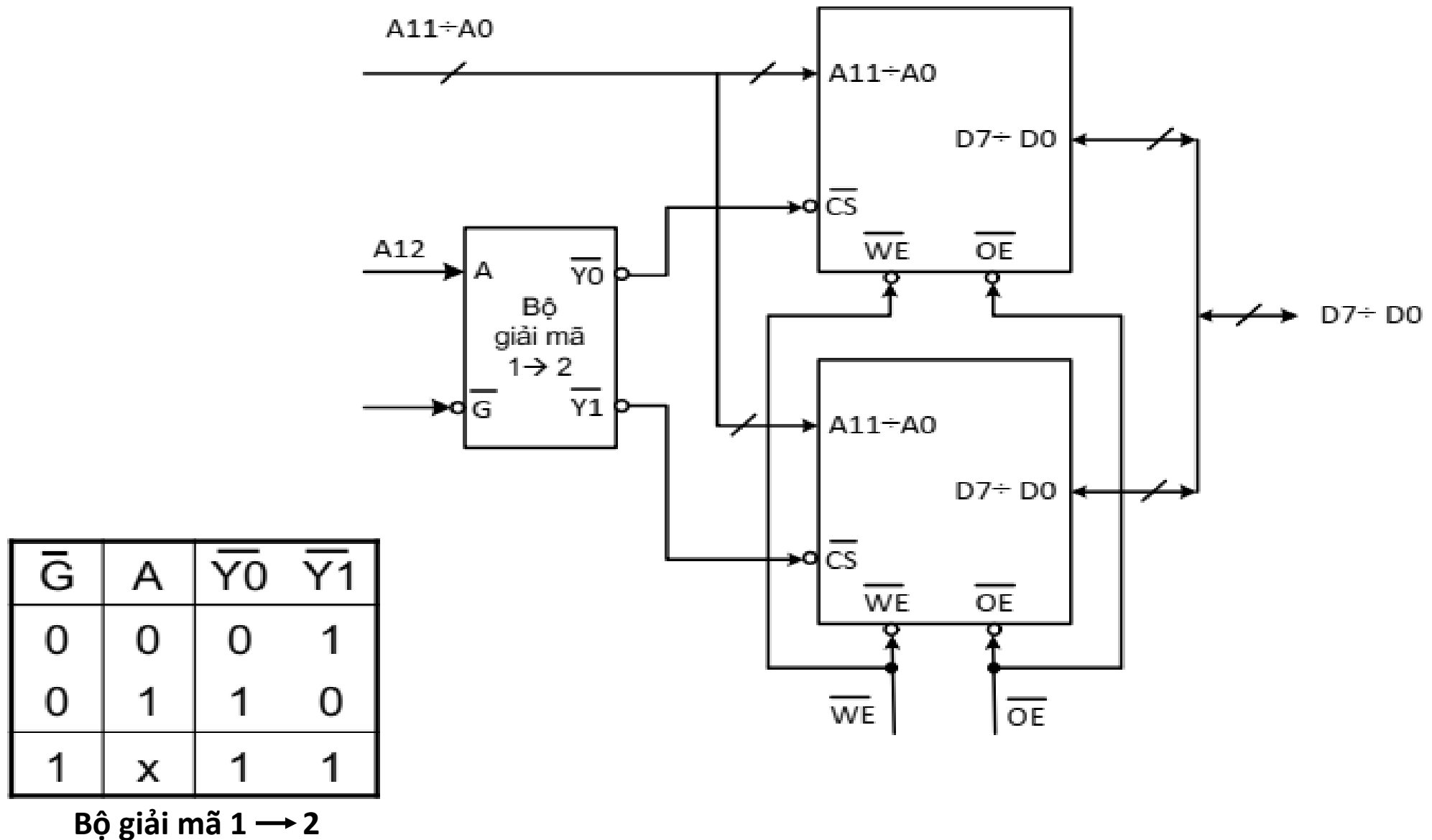
VD2:

- Cho chip nhớ SRAM 4K x 8 bit
- Thiết kế mô-đun nhớ 8K x 8 bit **Giải:**
- Dung lượng chip nhớ 4K x 8 bit =  $2^2 \times 2^{10} \times 8 \text{ bit} = 2^{12} \times 8 \text{ bit}$
- chip nhớ có:
  - 12 chân địa chỉ
  - 8 chân dữ liệu
- Dung lượng mô-đun nhớ =  $2^{13} \times 8 \text{ bit}$ 
  - 13 chân địa chỉ
  - 8 chân dữ liệu

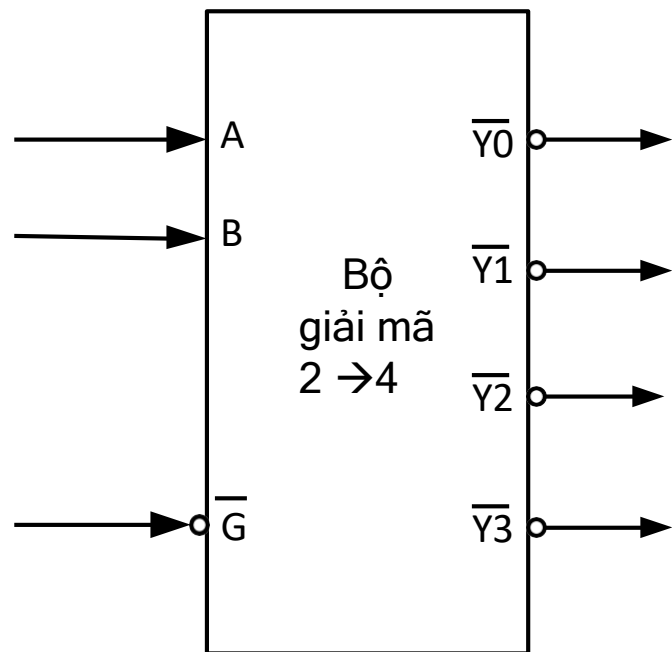
# Tăng số lượng và độ dài từ nhớ

- Bài toán tăng số lượng và độ dài tổng quát:
  - Cho chip nhớ  $2^n \times m$  bit
  - Cần ghép nối modul nhớ:  $2^{p+n} \times (q.m)$  bit
- Cần ghép nối  $q.2^p$  chip thành  $2^p$  bộ, mỗi bộ  $q$  chip và phải dùng bộ giải mã  $p: 2^p$  ( $p \rightarrow 2^p$ )

# Sơ đồ ví dụ tăng số lượng từ nhớ



# Bộ giải mã $2 \rightarrow 4$



$\overline{G}$	B	A	$\overline{Y0}$	$\overline{Y1}$	$\overline{Y2}$	$\overline{Y3}$
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0
1	x	x	1	1	1	1

# Tăng số lượng và độ dài ngăn nhớ

---

- Bài toán tăng số lượng và độ dài tổng quát:
- Cho chip nhớ  $2^n \times m$  bit
- Cần ghép nối modul nhớ:  $2^{p+n} \times (q.m)$  bit
- $\Rightarrow$  Cần ghép nối  $q.2^p$  chip thành  $2^p$  bộ, mỗi bộ  $q$  chip và phải dùng bộ giải mã  $p$ :  $2^p$  ( $p \rightarrow 2^p$ )

# Thiết kế kết hợp

Ví dụ 3:

- Cho chip nhớ SRAM 4K x 4 bit
- Thiết kế mô-đun nhớ 8K x 8 bit

Phương pháp thực hiện:

- Kết hợp ví dụ 1 và ví dụ 2

**Giải:**

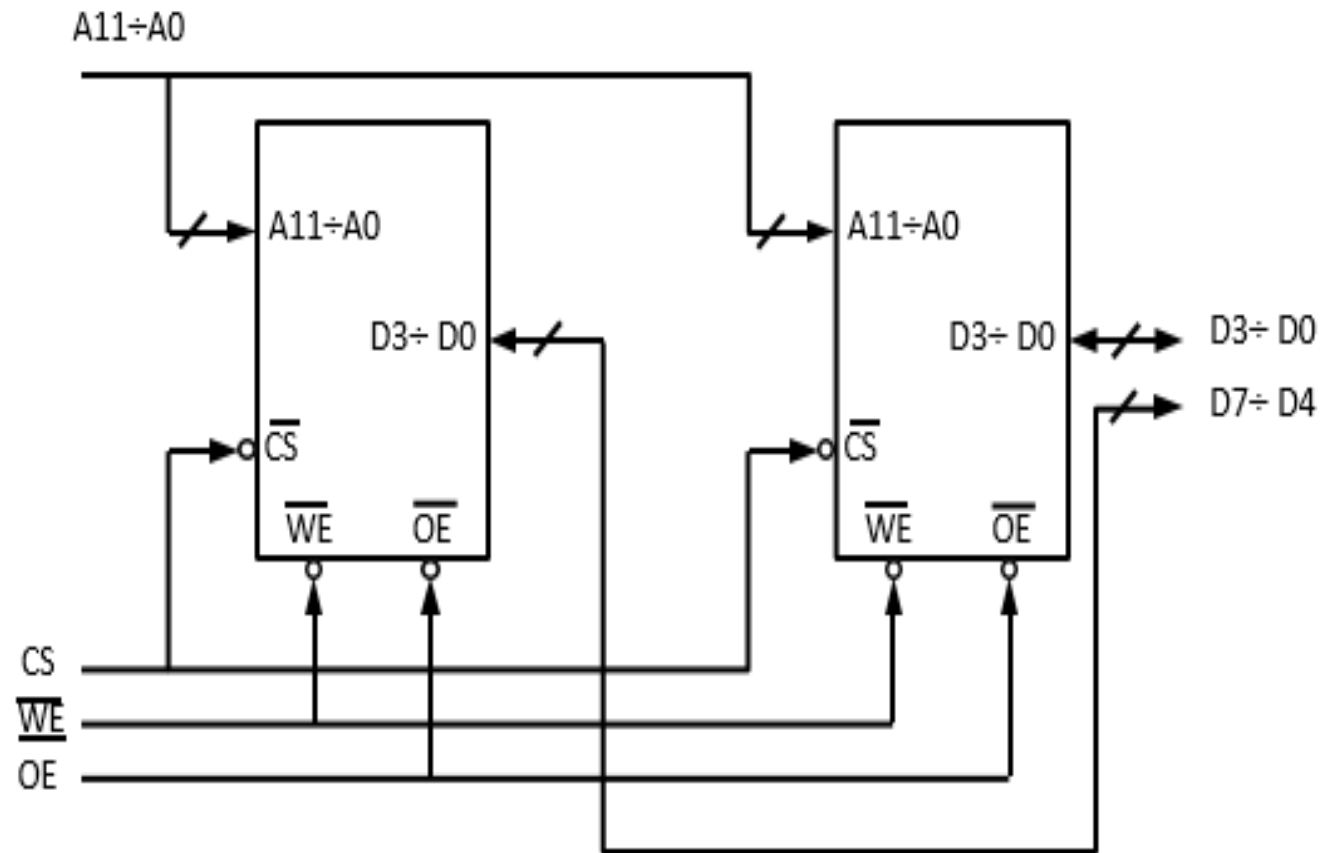
Chip nhớ:  $2^{12} \times 4$  bit

Module nhớ:  $2^1 \times 2^{12} \times (2 \times 4 \text{ bit})$

Cần ghép nối  $2 \times 2^1$  chip thành  $2^1$  bộ, mỗi bộ 2 chip và dùng bộ giải mã 1 : 2 (  $1 \rightarrow 2$  )

# Thiết kế kết hợp (ví dụ)

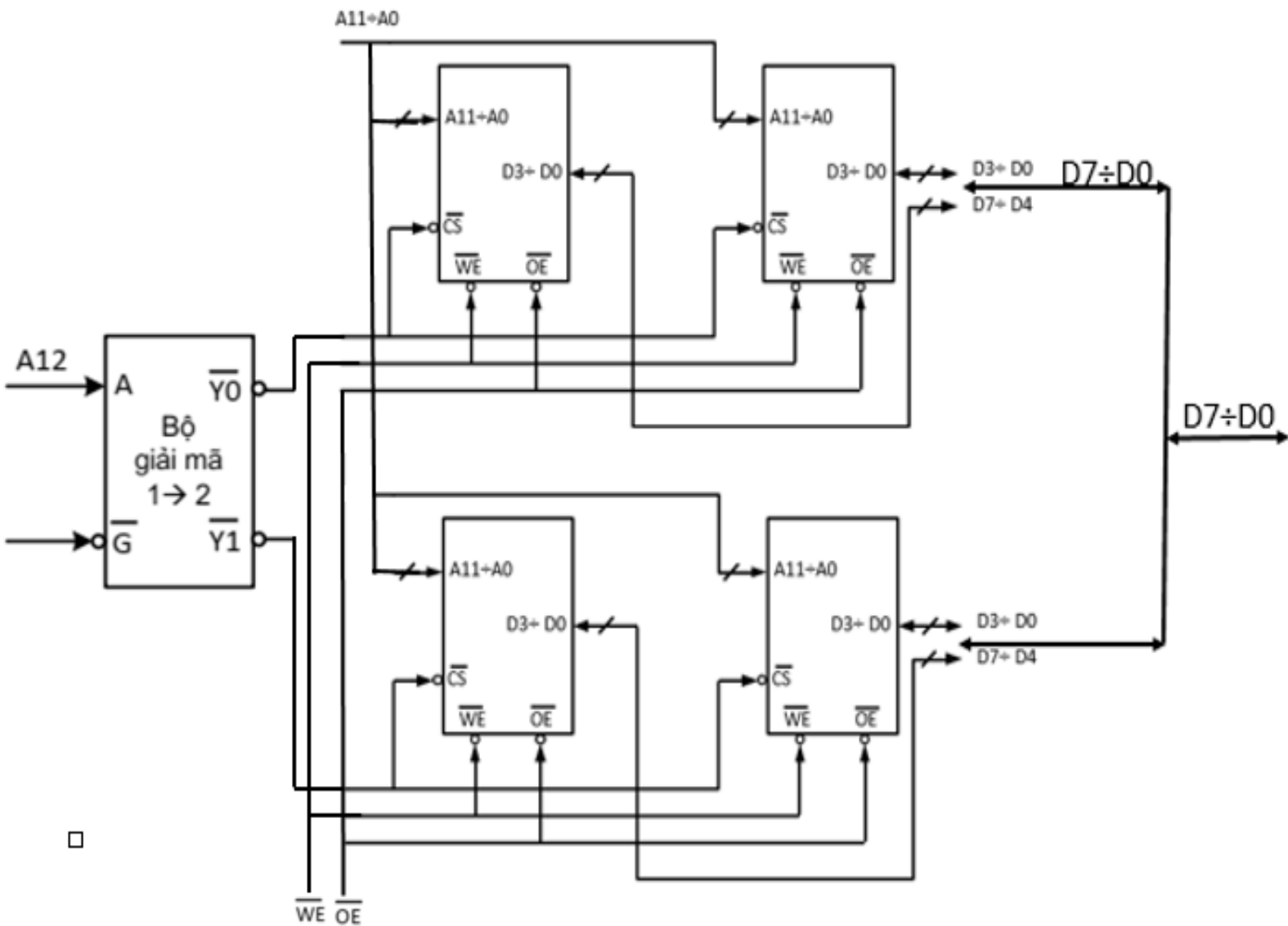
## Bước 1: Thiết kế module 4K x 8 bit



# Bước 2: Ghép 2 module ở bước 1 để có module 8K x 8 bit

$\overline{G}$	A	$\overline{Y0}$	$\overline{Y1}$
0	0	0	1
0	1	1	0
1	x	1	1

Bộ giải mã 1 → 2



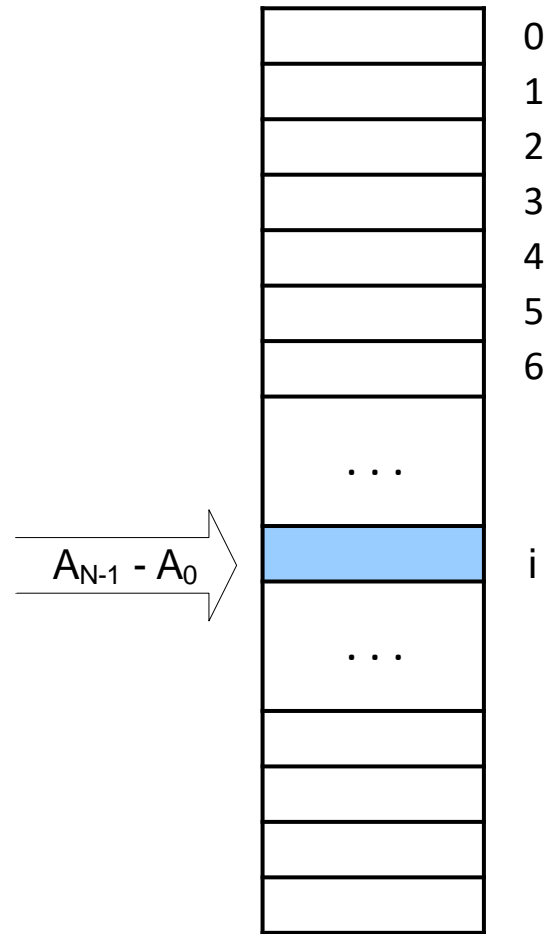
# Các đặc trưng cơ bản của bộ nhớ chính

- Chứa các chương trình đang thực hiện và các dữ liệu đang được sử dụng
- 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 nhỏ hơn không gian địa chỉ bộ nhớ mà CPU quản lý.
- Việc quản lý logic bộ nhớ chính tùy thuộc vào hệ điều hành

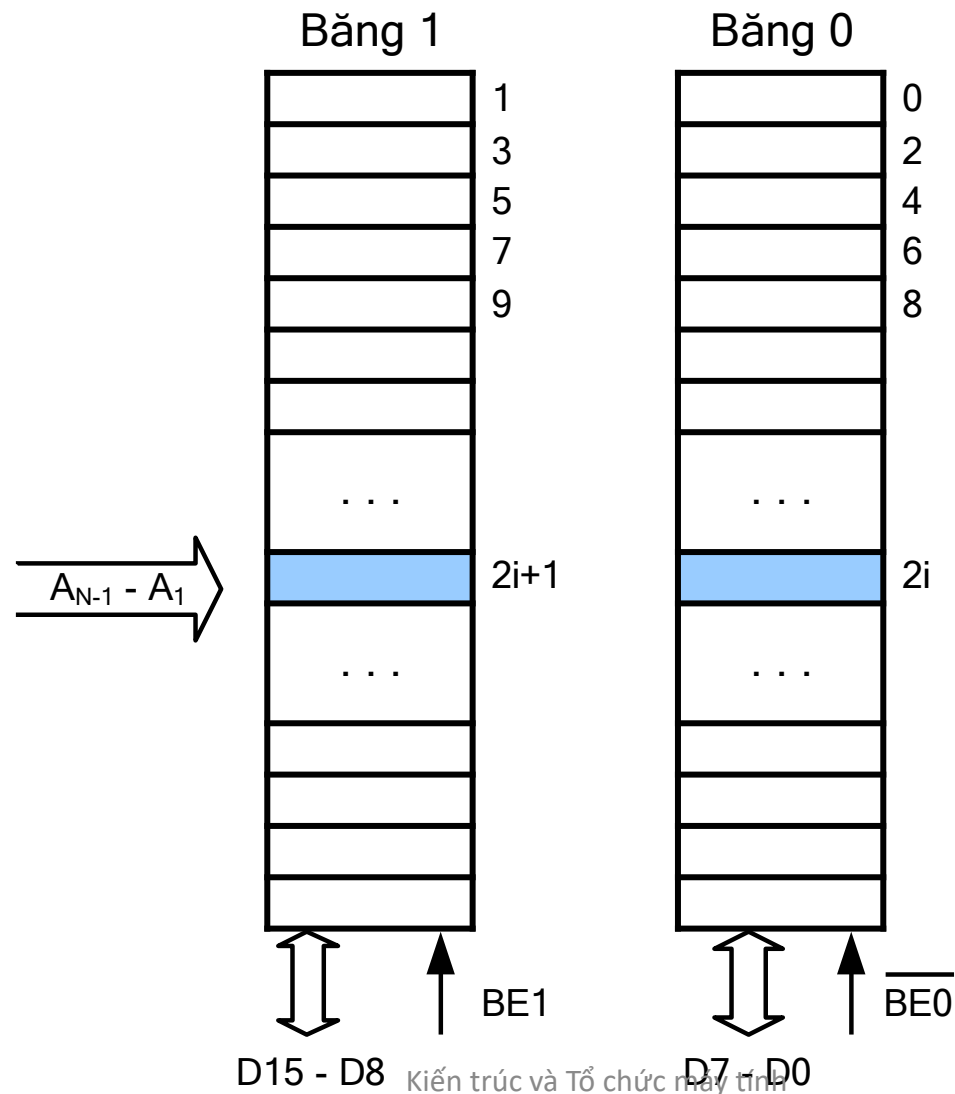
## Tổ chức bộ nhớ đan xen (interleaved memory)

- Độ rộng của bus dữ liệu để trao đổi với bộ nhớ:  $m = 8, 16, 32, 64, 128 \dots$  bit
- Các ngăn nhớ được tổ chức theo byte  
→ tổ chức bộ nhớ vật lý khác nhau

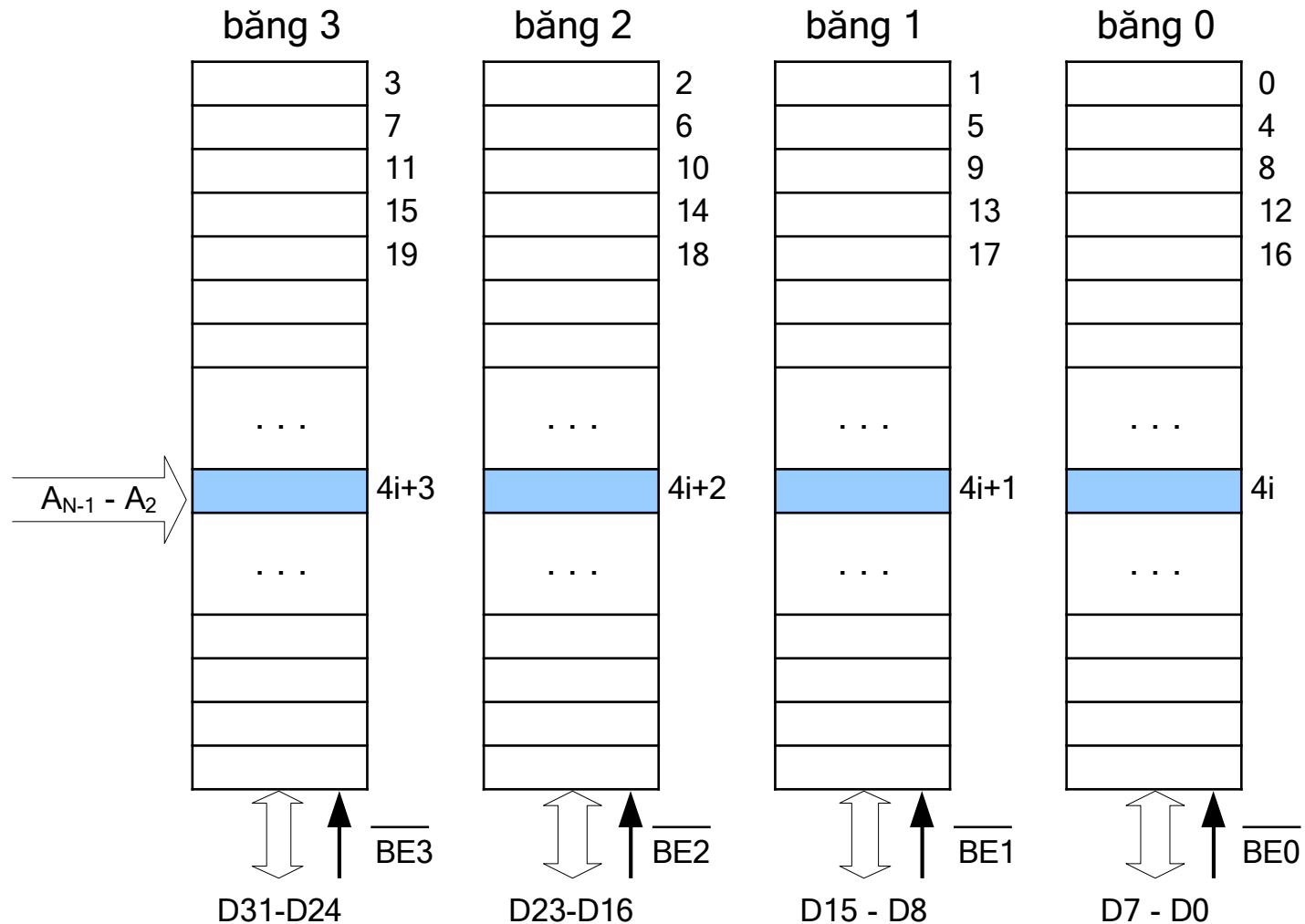
$m=8\text{bit} \rightarrow$  một bảng nhớ tuyến tính



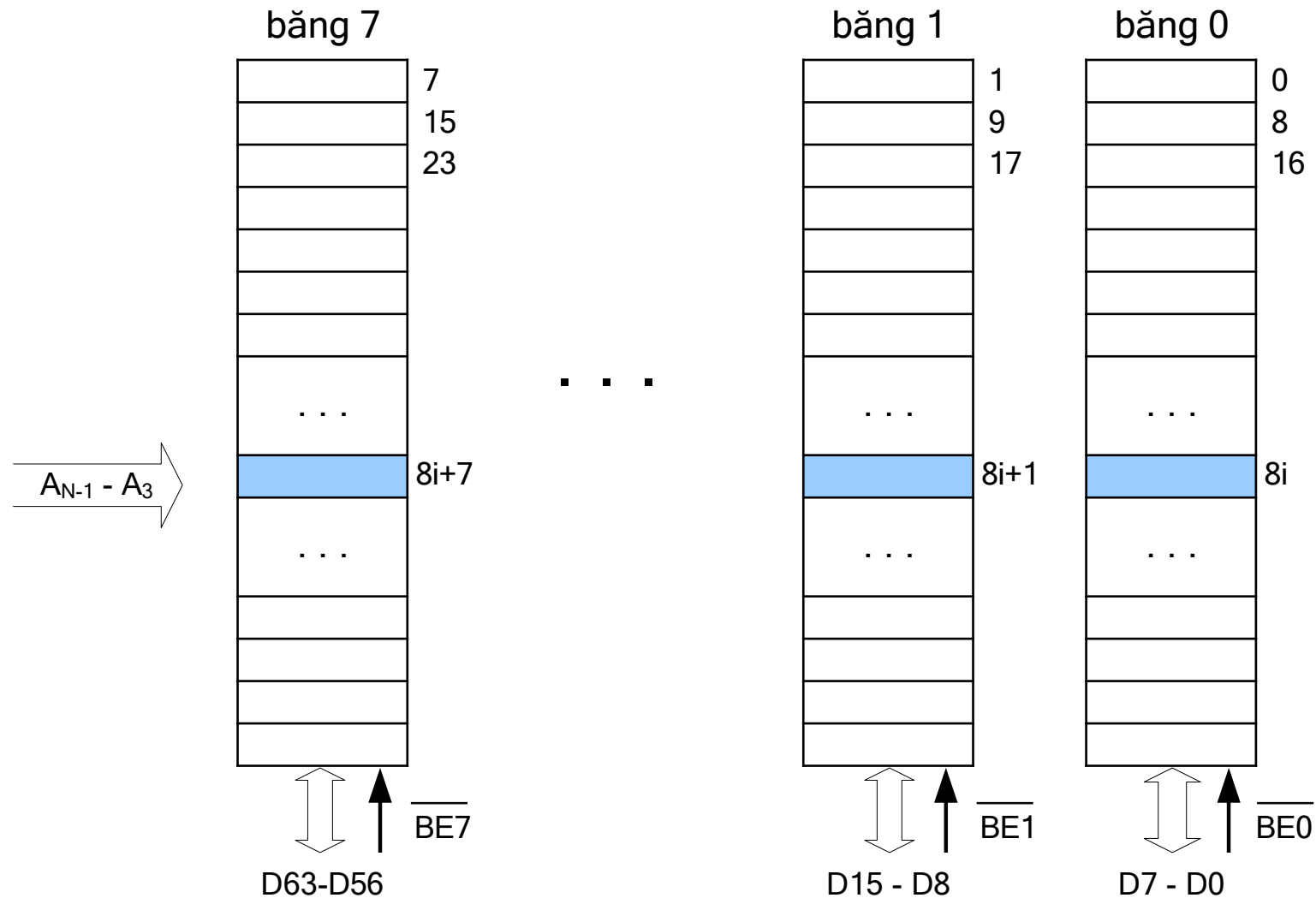
$m = 16\text{bit} \rightarrow$  hai bảng nhớ đan xen



**$m = 32\text{bit} \rightarrow$  bốn bảng nhớ đơn xen**



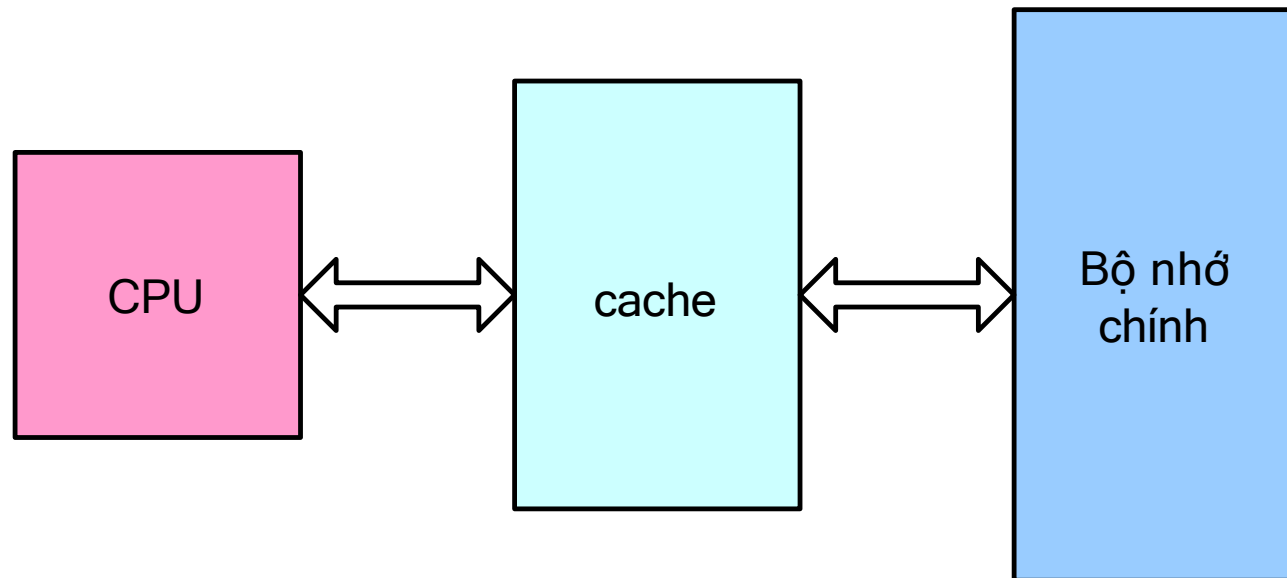
$m = 64\text{bit} \rightarrow$  tám băng nhớ đơn xen



# 4.5. Bộ nhớ đệm nhanh (cache memory)

## 1. Nguyên tắc chung của cache

- Cache có tốc độ nhanh hơn bộ nhớ chính
- Cache được đặt giữa CPU và bộ nhớ chính nhằm tăng tốc độ CPU truy cập bộ nhớ
- Cache có thể được đặt trên chip CPU



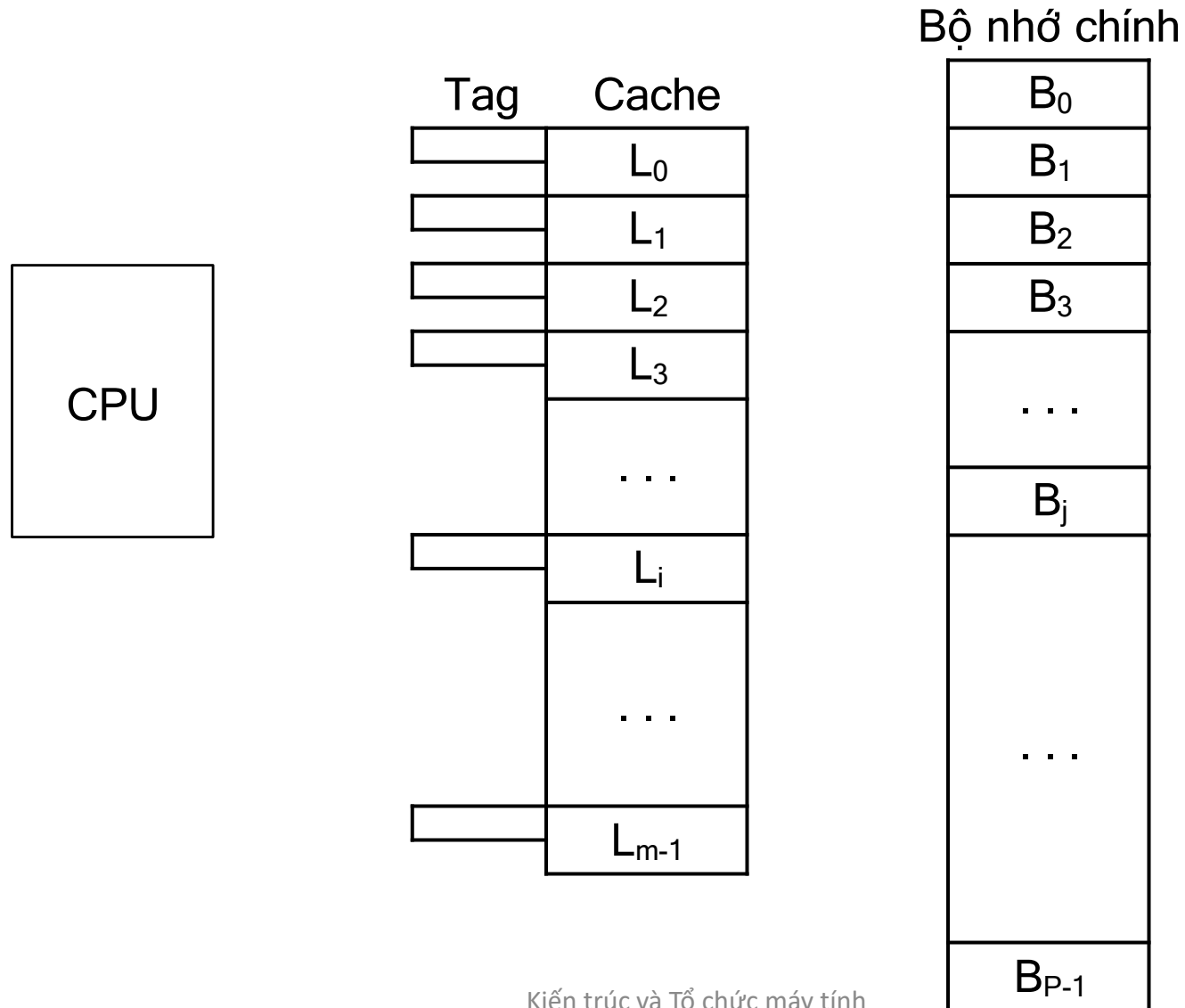
truyền theo từ nhớ

truyền theo block nhớ

# Ví dụ về thao tác của cache

- CPU yêu cầu nội dung của ngăn nhớ
- CPU kiểm tra trên cache với dữ liệu này
- Nếu có, CPU nhận dữ liệu từ cache (nhANH)
- Nếu không có, đọc Block nhớ chứa dữ liệu từ bộ nhớ chính vào cache
- Tiếp đó chuyển dữ liệu từ cache vào CPU

# Cấu trúc chung của cache / bộ nhớ chính



## Cấu trúc chung của cache / bộ nhớ chính (tiếp)

- Bộ nhớ chính có  $2^N$  byte nhớ
- Bộ nhớ chính và cache được chia thành các khối có kích thước bằng nhau
  - Bộ nhớ chính:  $B_0, B_1, B_2, \dots, B_{p-1}$  (p Blocks)
  - Bộ nhớ cache:  $L_0, L_1, L_2, \dots, L_{m-1}$  (m Lines)
  - Kích thước của Block (Line) = 8,16,32,64,128 byte
- Mỗi Line trong cache có một thẻ nhớ (Tag) được gắn vào

## Cấu trúc chung của cache / bộ nhớ chính (tiếp)

- 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
- Nội dung Tag được cập nhật mỗi khi Block từ bộ nhớ chính nạp vào Line đó
- Khi CPU truy nhập (đọc/ghi) một từ nhớ, có hai khả năng xảy ra:
  - Từ nhớ đó có trong cache (cache hit)
  - Từ nhớ đó không có trong cache (cache miss).

# Các phương pháp ánh xạ

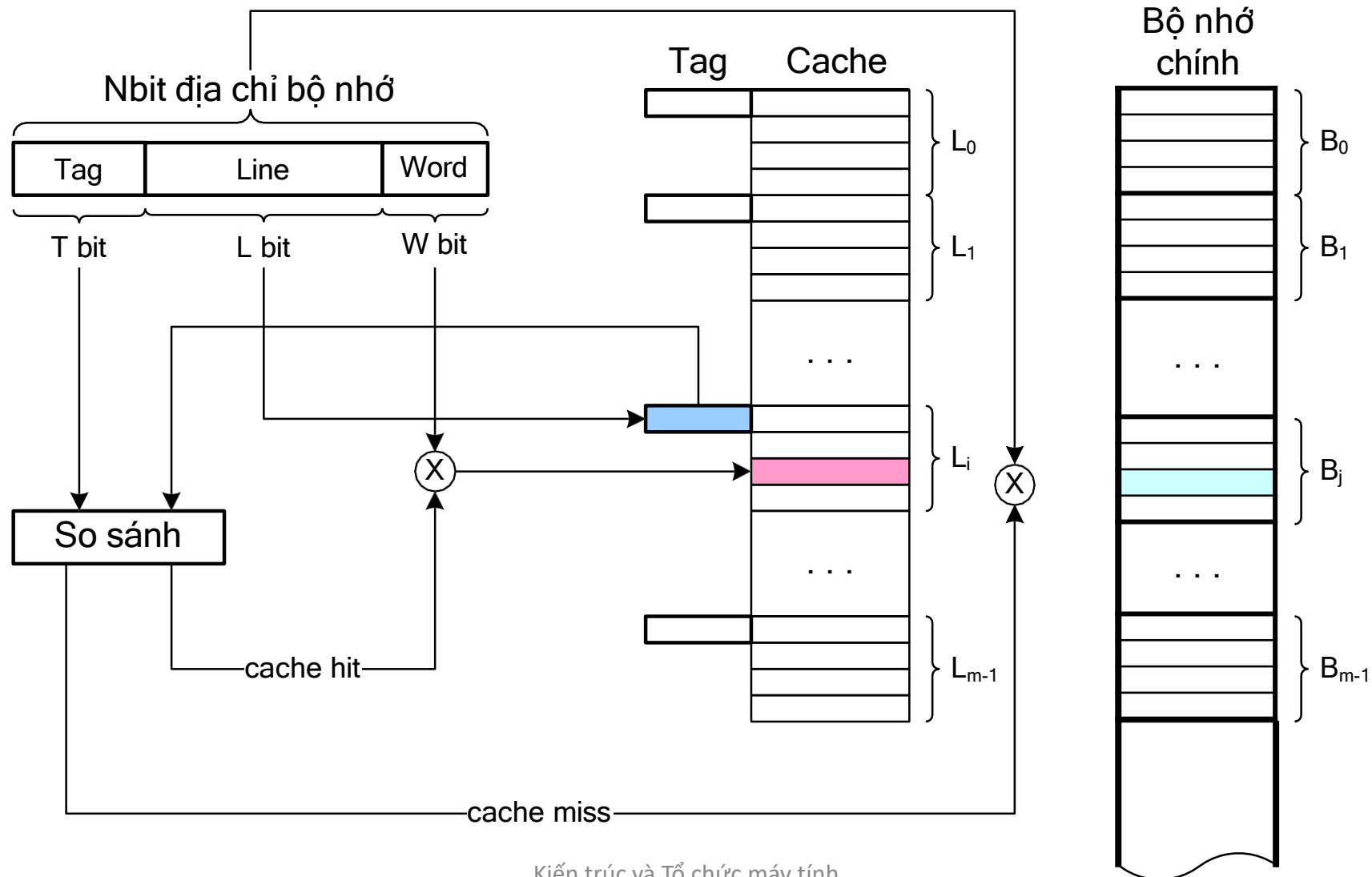
(Chính là các phương pháp tổ chức bộ nhớ cache)

- Ánh xạ trực tiếp (Direct mapping)
- Ánh xạ liên kết toàn phần (Fully associative mapping)
- Ánh xạ liên kết tập hợp (Set associative mapping)

# Ánh xạ trực tiếp

- Mỗi Block của bộ nhớ chính chỉ có thể được nạp vào một 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*.

# Ánh xạ trực tiếp (tiếp)



# Ánh xạ trực tiếp (tiếp)

- Địa chỉ N bit của bộ nhớ chính chia thành ba trường:
  - Trường *Word* gồm W bit xác định một từ nhớ trong *Block* hay *Line*:
$$2^W = \text{kích thước của } Block \text{ hay } Line$$
  - Trường *Line* gồm L bit xác định một trong số các *Line* trong *cache*:
$$2^L = \text{số } Line \text{ trong } cache = m$$
  - Trường *Tag* gồm T bit:
$$T = N - (W+L)$$

# Ánh xạ trực tiếp (tiếp)

- Mỗi thẻ nhớ (Tag) của một Line chứa được T bit
- Khi Block từ bộ nhớ chính được nạp vào Line của cache thì Tag ở đó được cập nhật giá trị là T bit địa chỉ bên trái của Block đó
- Khi CPU muốn truy nhập một từ nhớ thì nó phát ra một địa chỉ N bit cụ thể
  - Nhờ vào giá trị L bit của trường Line sẽ tìm ra Line tương ứng
  - Đọc nội dung Tag ở Line đó (T bit), rồi so sánh với T bit bên trái của địa chỉ vừa phát ra
    - Giống nhau: cache hit
    - Khác nhau: cache miss
- Ưu điểm: Bộ so sánh đơn giản
- Nhược điểm: Xác suất *cache hit* thấp

## Ví dụ

---

- Hệ thống có: bộ nhớ chính = 256 MB  
Cache = 128 KB  
Line = 16 Byte
- Xác định số bit của các trường địa chỉ

GIẢI:

$$2^N = 256.2^{20} = 2^{28} \rightarrow N = 28 \text{ bit}$$

- Tính cho trường Byte:

$$\text{Kích thước line} = 16 = 2^4 \text{ Byte} \rightarrow n_1 = 4 \text{ bit}$$

- Tính cho trường Line:

$$\text{Số line trong Cache: } 128.2^{10}/16 = 2^{13} \rightarrow n_2 = 13 \text{ bit}$$

- Tính cho trường Tag:

$$n_3 = N - (n_1 + n_2) = 28 - (4 + 13) = 11 \text{ bit}$$

## Ví dụ 2

Cho máy tính có dung lượng bộ nhớ chính: 256MB, cache: 128KB, line: 16B, độ dài ngăn nhớ: 2B. Trong trường hợp kỹ thuật ánh xạ trực tiếp, dạng địa chỉ do bộ xử lý phát ra để truy nhập cache là như thế nào

Giải:

$$2^N = 256 \times 2^{20} / 2 \text{ (vì mỗi ngăn nhớ 2B)} = 2^8 \times 2^{20} / 2 = 2^{27} \Rightarrow N = 27 \text{ bit}$$

Tính cho trường Byte:

$$\text{Kích thước Line} = 2^4 \text{B} / 2 \text{ (vì mỗi ngăn nhớ 2B)} = 2^3 \text{B}$$

$$\Rightarrow n_1 = 3 \text{ bit}$$

Tính cho trường Line:

$$\text{Số line trong cache} = 128 \times 2^{10} / 16 = 2^{17} / 2^4 = 2^{13} \text{ vậy } n_2 = 13$$

Tính cho trường Tag:

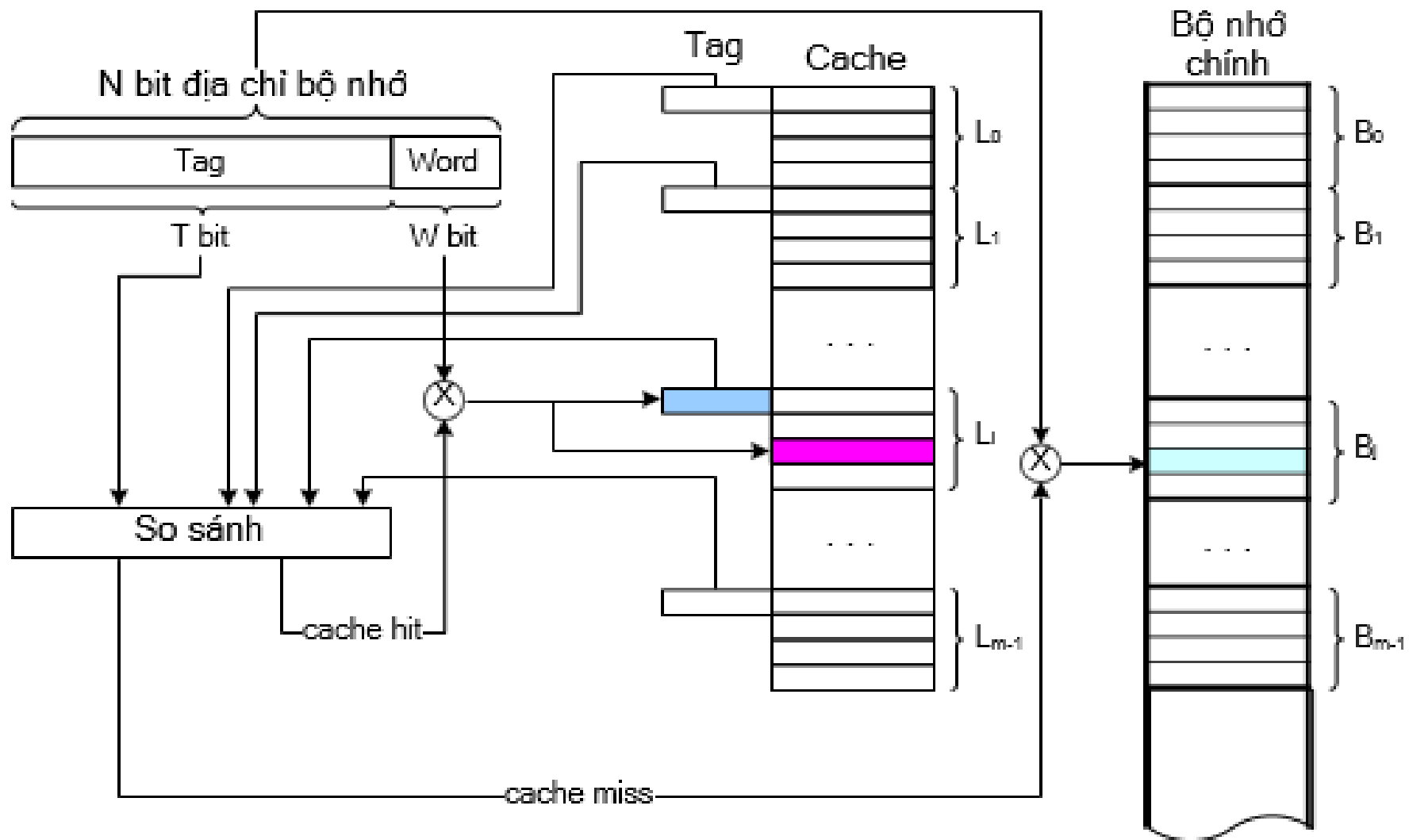
$$n_3 = N - n_1 - n_2 = 27 - 3 - 13 = 11$$

$$\text{Kết luận: } n_3 + n_2 + n_1 = 11 + 13 + 3$$

# Ánh xạ liên kết toàn phần

- Mỗi *Block* có thể nạp vào bất kỳ *Line* nào của *cache*
- Địa chỉ của bộ nhớ chính chia thành hai trường:
  - Trường *Word*
  - Trường *Tag* dùng để xác định *Block* của bộ nhớ chính
- Tag xác định Block đang nằm ở Line đó

## Ảnh xạ liên kết toàn phần (tiếp)



## Ảnh xạ liên kết toàn phần (tiếp)

- Mỗi thẻ nhớ (Tag) của một Line chứa được T bit
- Khi Block từ bộ nhớ chính được nạp vào Line của cache thì Tag ở đó được cập nhật giá trị là T bit địa chỉ bên trái của Block đó
- Khi CPU muốn truy nhập một từ nhớ thì nó phát ra một địa chỉ N bit cụ thể
  - So sánh T bit bên trái của địa chỉ vừa phát ra với lần lượt nội dung của các Tag trong cache
    - Nếu gặp giá trị bằng nhau: cache hit xảy ra ở Line đó
    - Nếu không có giá trị nào bằng: cache miss
- Ưu điểm: Xác suất *cache hit* cao
- Nhược điểm:
  - So sánh đồng thời với tất cả các Tag → mất nhiều thời gian
  - Bộ so sánh phức tạp
- Ít sử dụng

## Ánh xạ liên kết toàn phần (ví dụ)

Cho máy tính có dung lượng bộ nhớ chính: 256MB, cache: 64KB, line: 16 byte, độ dài ngăn nhớ: 4 byte. Trong trường hợp kỹ thuật ánh xạ liên kết toàn phần, dạng địa chỉ do bộ xử lý phát ra để truy nhập cache là?

Giải:

Bộ nhớ chính  $2^N = 256 \times 2^{20} / 2^2 = 2^{26}$ , số bit địa chỉ cần  $N=26$  do ngăn nhớ 4B

Line 16B =  $2^4$ B, do 4B/ngăn nhớ  $2^4 / 2^2 \Rightarrow n1=2$

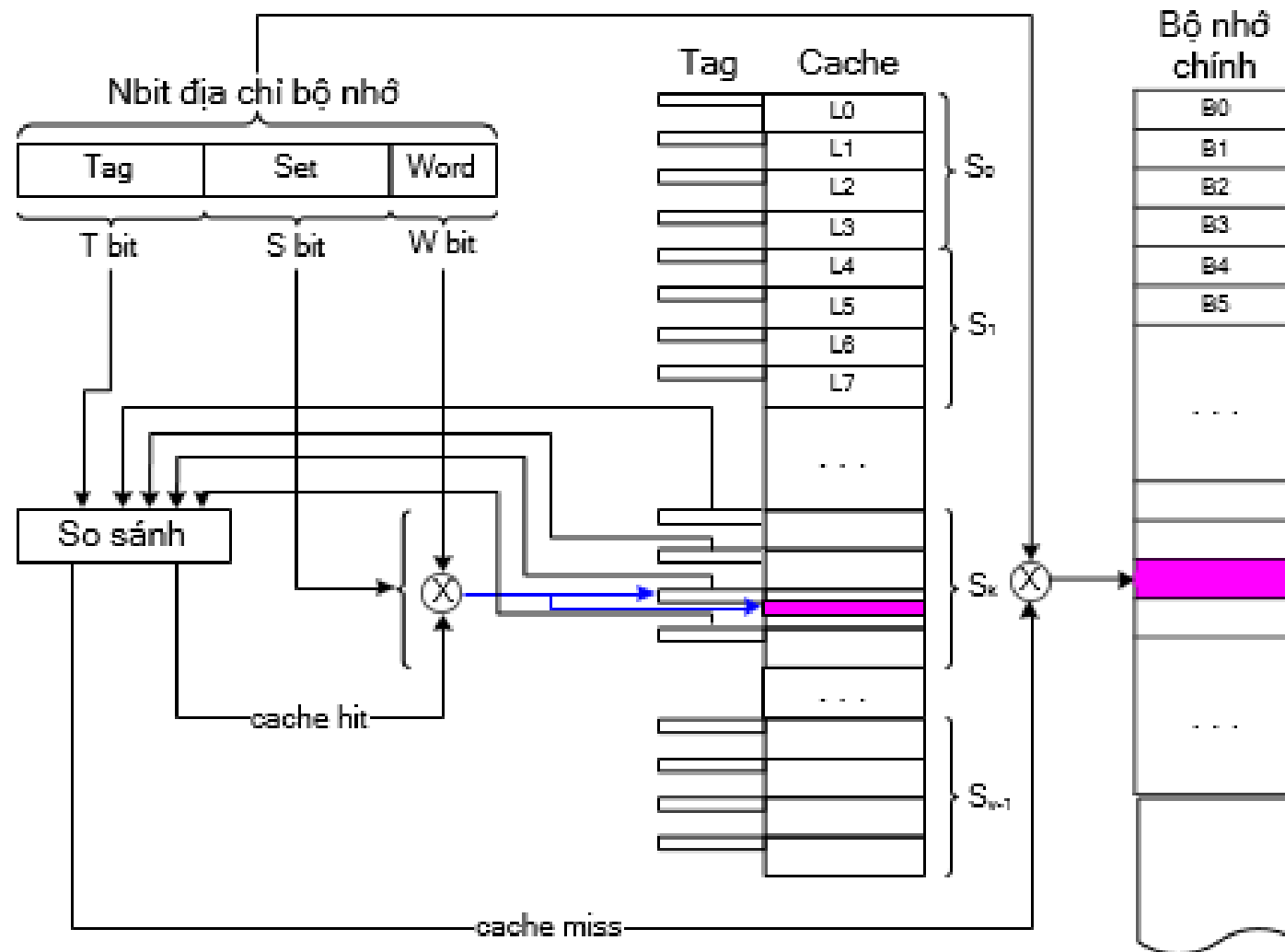
$n2 = N - n1 = 26 - 2 = 24$

Vậy dạng địa chỉ là: 24+2

# Ánh xạ liên kết tập hợp

- Dung hòa cho hai phương pháp trên
- Cache được chia thành các Tập (Set)
- Mỗi một Set chứa một số Line
- Ví dụ:
  - 4 Line/Set  $\rightarrow$  4-way associative mapping
- Ánh xạ theo nguyên tắc sau:
  - $B_0 \rightarrow S_0$
  - $B_1 \rightarrow S_1$
  - $B_2 \rightarrow S_2$
  - .....

# Ánh xạ liên kết tập hợp (tiếp)



# Ánh xạ liên kết tập hợp (tiếp)

- Kích thước  $Block = 2^W$  Word
- Trường *Set* có  $S$  bit dùng để xác định một trong số các Set trong cache.  $2^S = \text{Số Set trong cache}$
- Trường *Tag* có  $T$  bit:  $T = N - (W+S)$
- Khi CPU muốn truy nhập một từ nhớ thì nó phát ra một địa chỉ  $N$  bit cụ thể
  - Nhờ vào giá trị  $S$  bit của trường *Set* sẽ tìm ra Set tương ứng
  - So sánh  $T$  bit bên trái của địa chỉ vừa phát ra với lần lượt nội dung của các *Tag* trong Set đó
    - Nếu gặp giá trị bằng nhau: cache hit xảy ra ở Line tương ứng
    - Nếu không có giá trị nào bằng: cache miss
- Tổng quát cho cả hai phương pháp trên
- Thông dụng với: 2,4,8,16Lines/Set

## Ví dụ

---

- Hệ thống có: bộ nhớ chính = 256 MB
  - Cache = 128 KB
  - Line = 16 Byte
- Xác định số bit của các trường địa chỉ khi
  - Ánh xạ liên kết tập hợp 4 Line/Set

GIẢI

- Tính cho trường Byte:

$$\text{Kích thước line} = 16 = 2^4 \text{ Byte} \rightarrow n_1 = 4 \text{ bit}$$

- Trường Set:

$$\text{Số Set} = \text{Số line}/4 = 2^{13}/4 = 2^{11} \rightarrow n_2 = 11 \text{ bit}$$

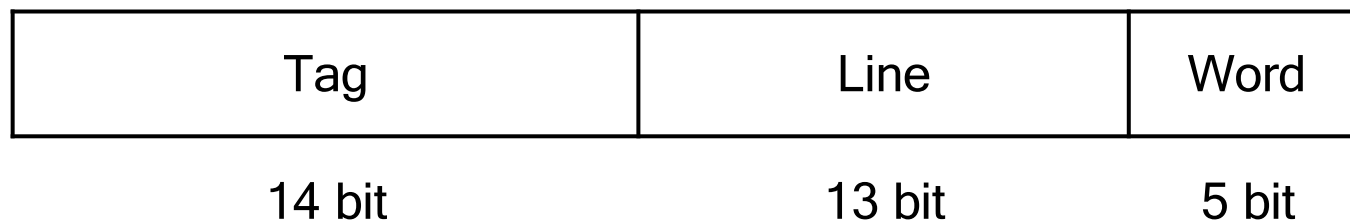
- Trường Tag:  $n_3 = N - (n_1 + n_2) = 28 - (4 + 11) = 13 \text{ bit}$

# Các ví dụ khác về ánh xạ địa chỉ

- Giả sử máy tính đánh địa chỉ cho từng byte
- Không gian địa chỉ bộ nhớ chính = 4GiB
- Dung lượng bộ nhớ *cache* là 256KiB
- Kích thước *Line (Block)* = 32byte.
- Xác định số bit của các trường địa chỉ cho ba trường hợp tổ chức:
  - Ánh xạ trực tiếp
  - Ánh xạ liên kết toàn phần
  - Ánh xạ liên kết tập hợp 4 đường

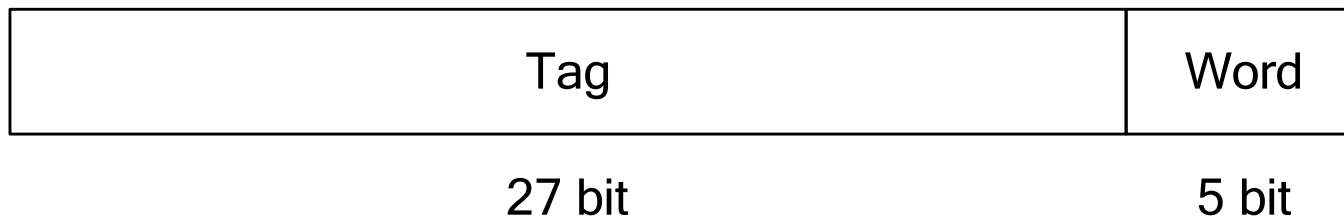
# Với ánh xạ trực tiếp

- Bộ nhớ chính = 4GiB =  $2^{32}$  byte  $\rightarrow$  Số bit địa chỉ của bộ nhớ chính là:  $N = 32$  bit
- *Cache* = 256 KiB =  $2^{18}$  byte
- Kích thước *Line* = 32 byte =  $2^5$  byte  $\rightarrow$  số bit địa chỉ của trường Word là:  $W = 5$  bit
- Số *Line* trong *cache* =  $2^{18} / 2^5 = 2^{13}$  *Line*  $\rightarrow$  số bit địa chỉ trường Line là:  $L = 13$  bit
- Số bit địa chỉ của trường Tag là:  
 $T = 32 - (13 + 5) = 14$  bit



# Với ánh xạ liên kết toàn phần

- Bộ nhớ chính = 4GiB =  $2^{32}$  byte  $\rightarrow$  số bit địa chỉ của bộ nhớ chính là:  $N = 32$  bit
- Kích thước *Line* = 32 byte =  $2^5$  byte  $\rightarrow$  số bit địa chỉ của trường Word là:  $W = 5$  bit
- Số bit địa chỉ của trường *Tag* là:  
 $T = 32 - 5 = 27$  bit



# Với ánh xạ liên kết tập hợp 4 đường

- Bộ nhớ chính = 4GiB =  $2^{32}$  byte  $\rightarrow$  số bit địa chỉ của bộ nhớ chính là:  $N = 32$  bit
- Kích thước *Line* = 32 byte =  $2^5$  byte  $\rightarrow$  số bit địa chỉ của trường Word là:  $W = 5$  bit
- Số *Line* trong *cache* =  $2^{18} / 2^5 = 2^{13}$  *Line*
- Một *Set* có 4 *Line* =  $2^2$  *Line*  
 $\rightarrow$  số *Set* trong *cache* =  $2^{13} / 2^2 = 2^{11}$  *Set*  
 $\rightarrow$  số bit địa chỉ của trường Set là:  $S = 11$  bit
- Số bit địa chỉ của trường *Tag* là:

$$T = 32 - (11 + 5) = 16 \text{ bit}$$



# Thay thế block trong cache

Với ánh xạ trực tiếp:

- Không phải lựa chọn
- Mỗi Block chỉ ánh xạ vào một Line xác định
- Thay thế Block ở Line đó

# Thay thế block trong cache (tiếp)

Với ánh xạ liên kết: cần có thuật giải thay thế:

- **Random**: Thay thế ngẫu nhiên
- **FIFO** (First In First Out): Thay thế *Block* nào nằm lâu nhất ở trong *Set* đó
- **LFU** (Least Frequently Used): Thay thế *Block* nào trong *Set* có số lần truy nhập ít nhất trong cùng một khoảng thời gian
- **LRU** (Least Recently Used): Thay thế *Block* ở trong *Set* tương ứng có thời gian lâu nhất không được tham chiếu tới
- Tối ưu nhất: **LRU**

# Phương pháp ghi dữ liệu khi cache hit

- Ghi xuyên qua (Write-through):
  - ghi cả cache và cả bộ nhớ chính
  - tốc độ chậm
- Ghi trả sau (Write-back):
  - chỉ ghi ra cache
  - tốc độ nhanh
  - khi Block trong cache bị thay thế cần phải ghi trả cả Block về bộ nhớ chính