



Chương 4: Hệ thống bộ nhớ

Chương 4: Nội dung chính

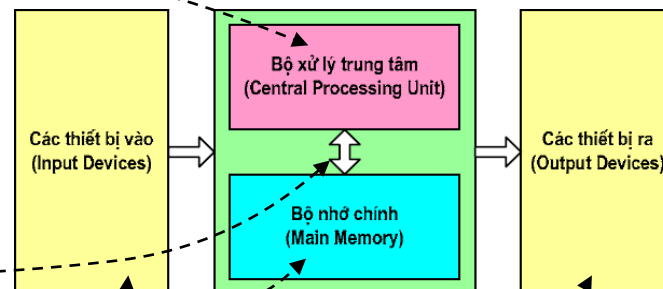
1. Bộ nhớ trong và mô hình phân cấp bộ nhớ
2. Phân loại bộ nhớ và tổ chức mạch nhớ
3. ROM
4. RAM
5. Bộ nhớ cache

Phân cấp bộ nhớ máy tính

Các đặc trưng của bộ nhớ máy tính

Vị trí

- Bên trong CPU:
 - tập thanh ghi
- Bộ nhớ trong:
 - bộ nhớ cache
 - bộ nhớ chính
- Bộ nhớ ngoài: các thiết bị nhớ



Dung lượng

- Độ dài từ nhớ (tính bằng bit)
- Số lượng từ nhớ



Các đặc trưng của hệ thống nhớ (tiếp)

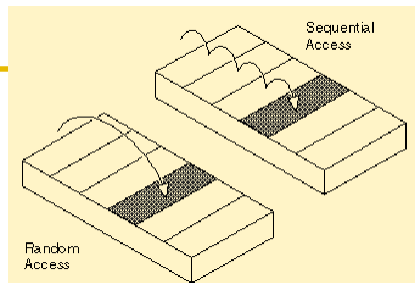
Đơn vị truyền

- 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 hệ thống nhớ (tiếp)

Hiệu năng (performance)

- Thời gian truy nhập
- Chu kỳ nhớ
- Tốc độ truyền

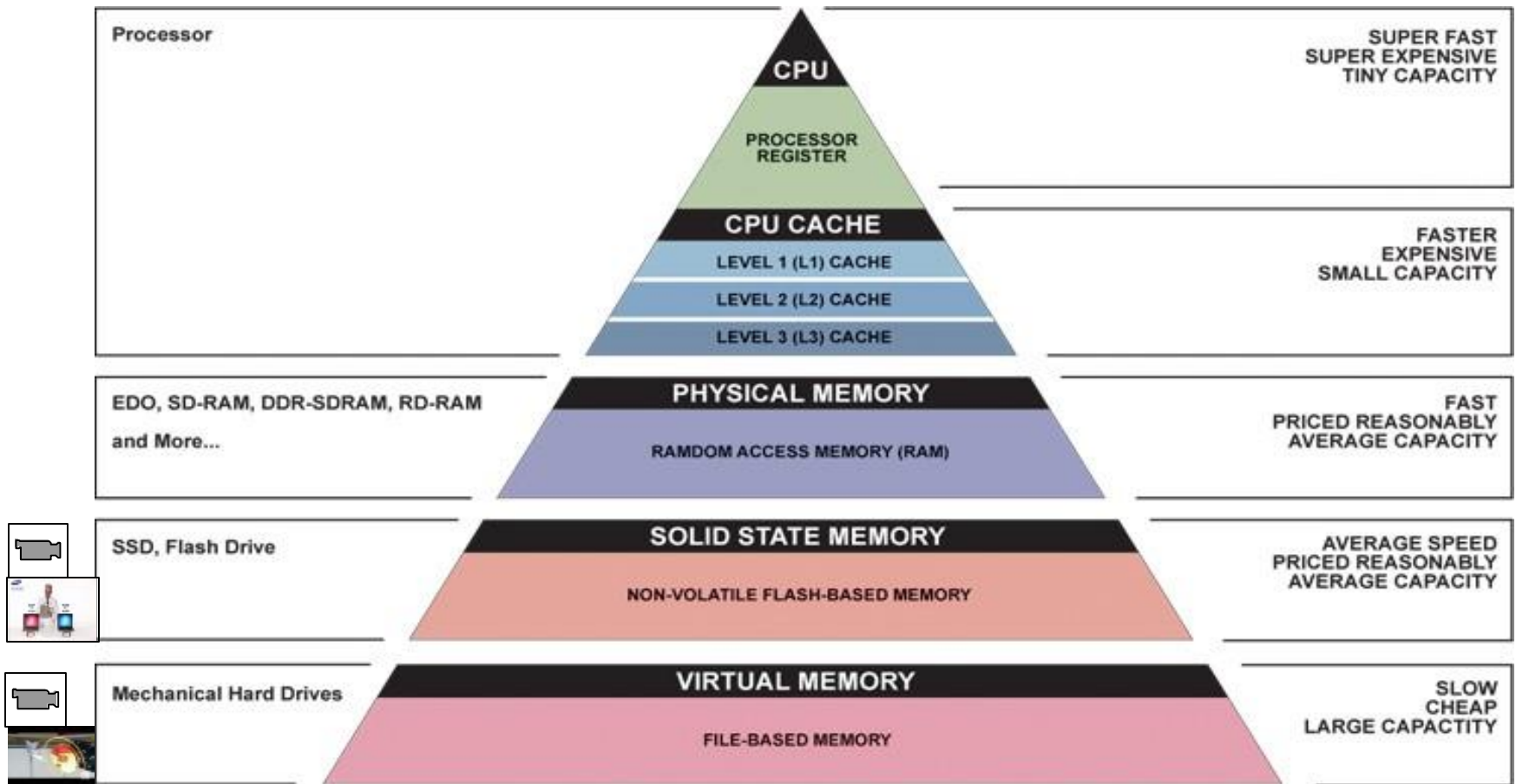
Kiểu vật lý

- Bộ nhớ bán dẫn
- Bộ nhớ từ
- Bộ nhớ quang

Các đặc trưng của hệ thống nhớ (tiếp)

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

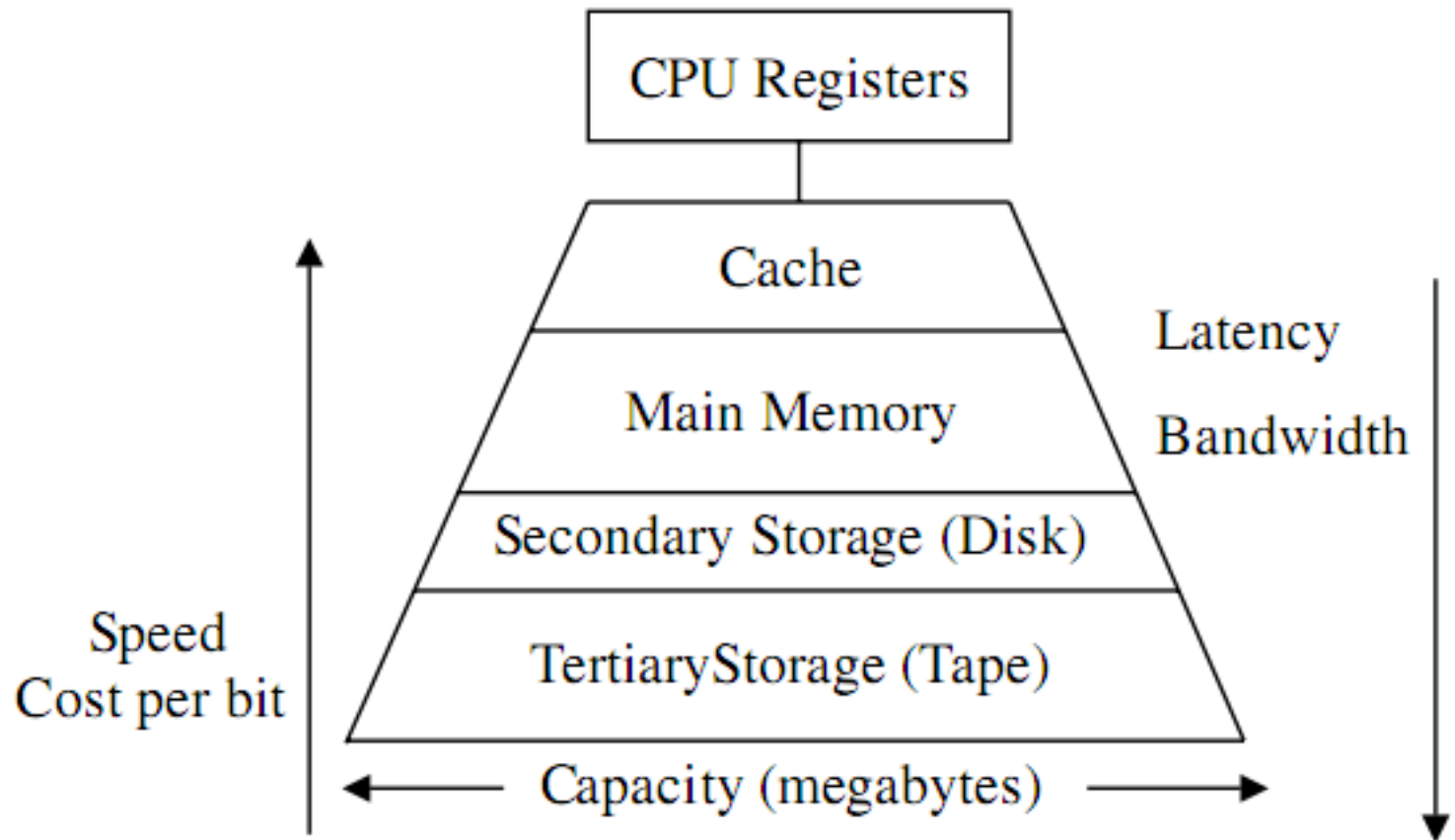
Phân cấp bộ nhớ



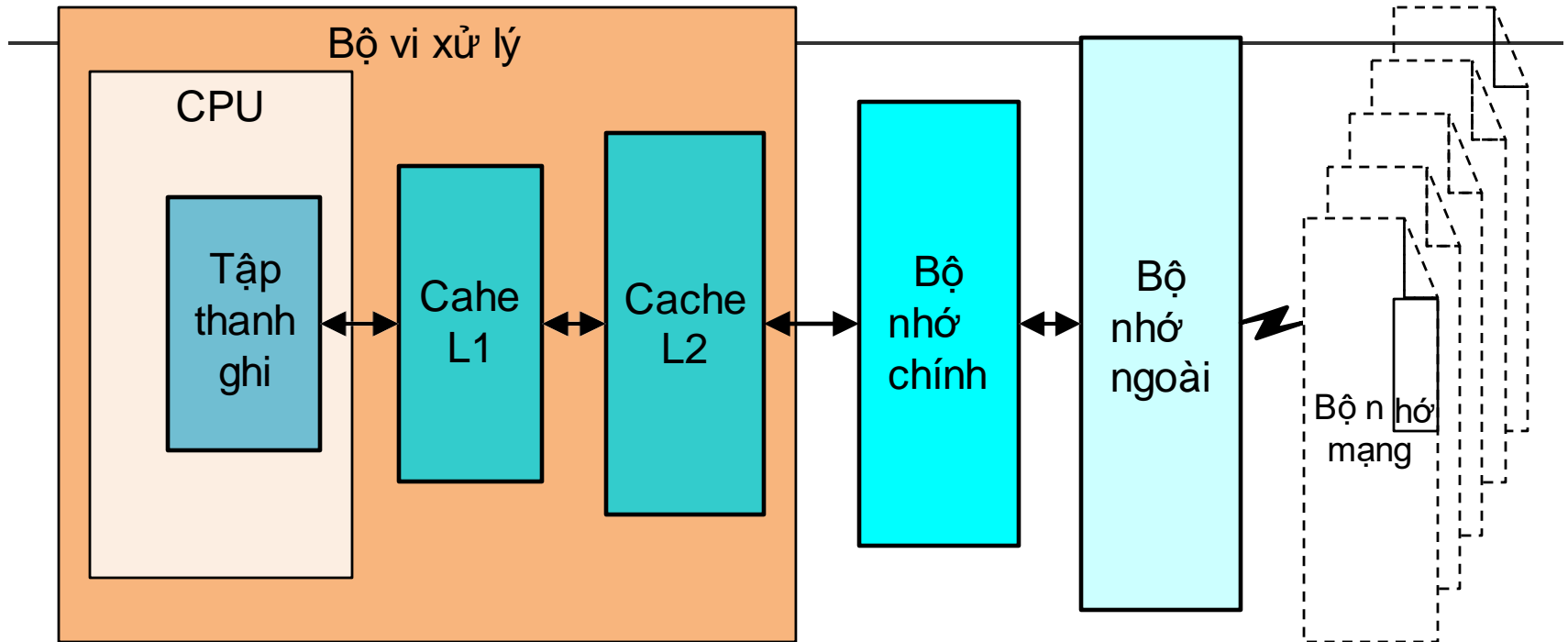
▲ Simplified Computer Memory Hierarchy



Mô hình phân cấp hệ thống bộ nhớ



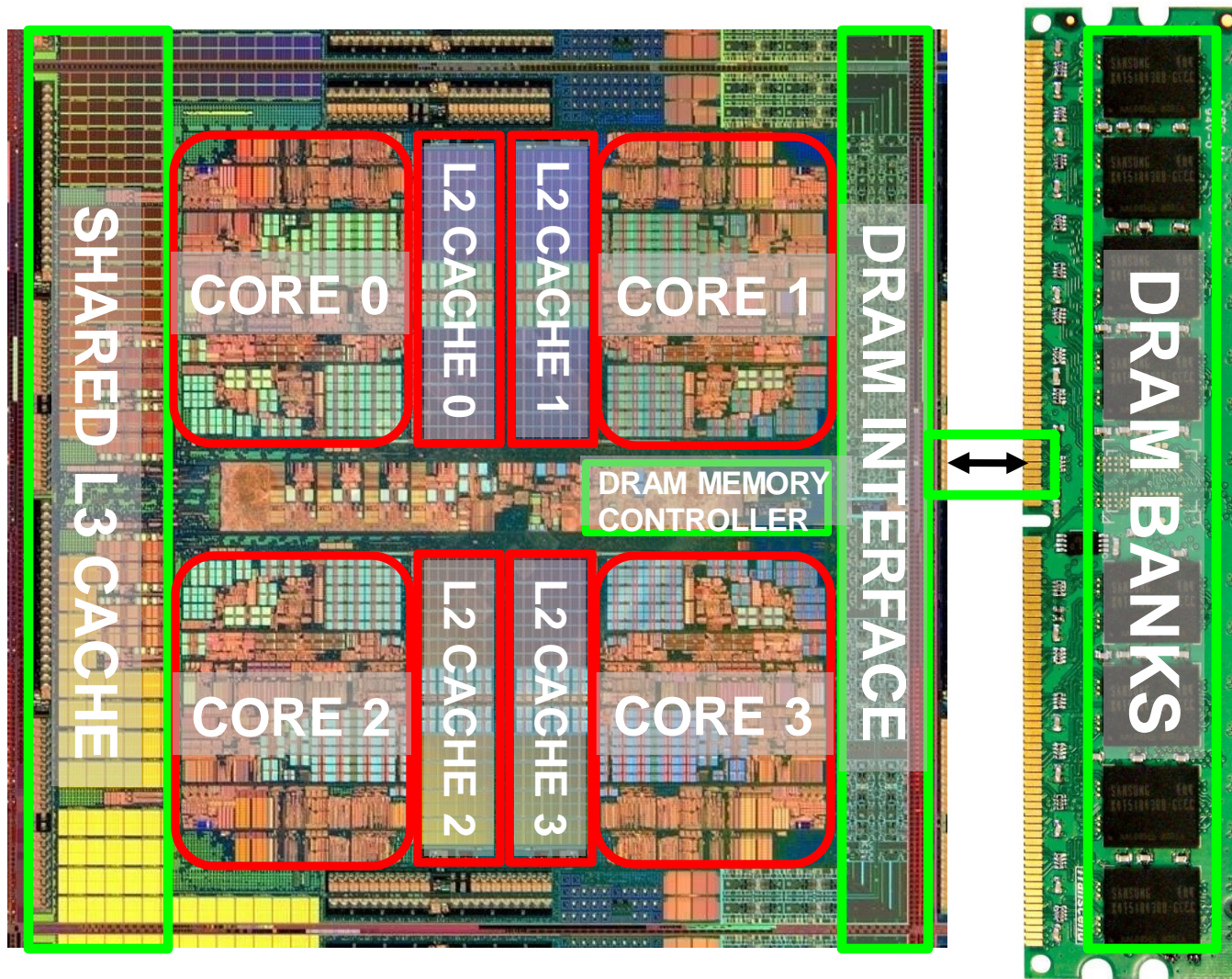
Ví dụ hệ thống nhớ thông dụng



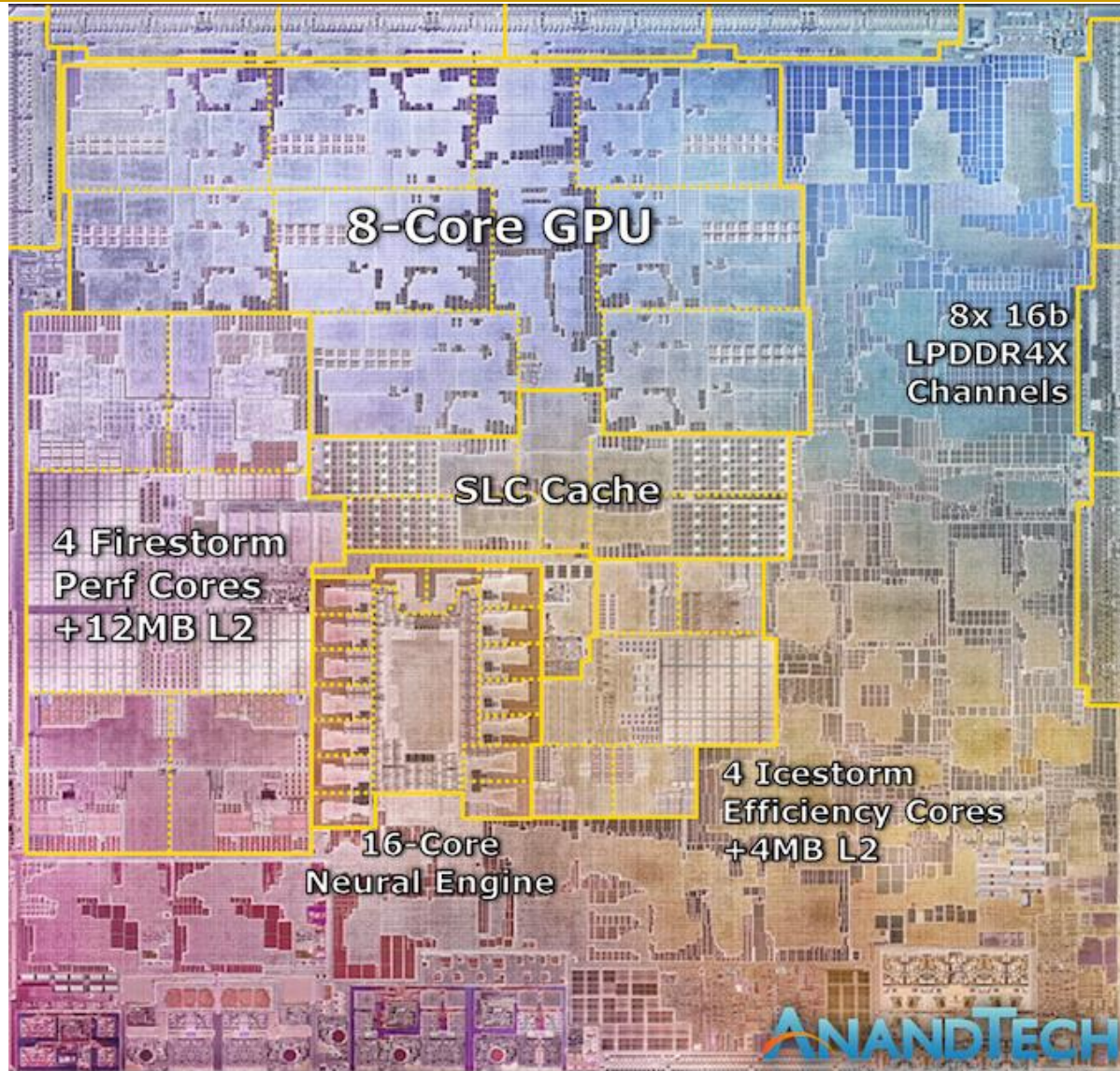
Từ trái sang phải:

- dung lượng tăng dần
- tốc độ giảm dần
- giá thành/1bit giảm dần

Hệ thống phân cấp bộ nhớ

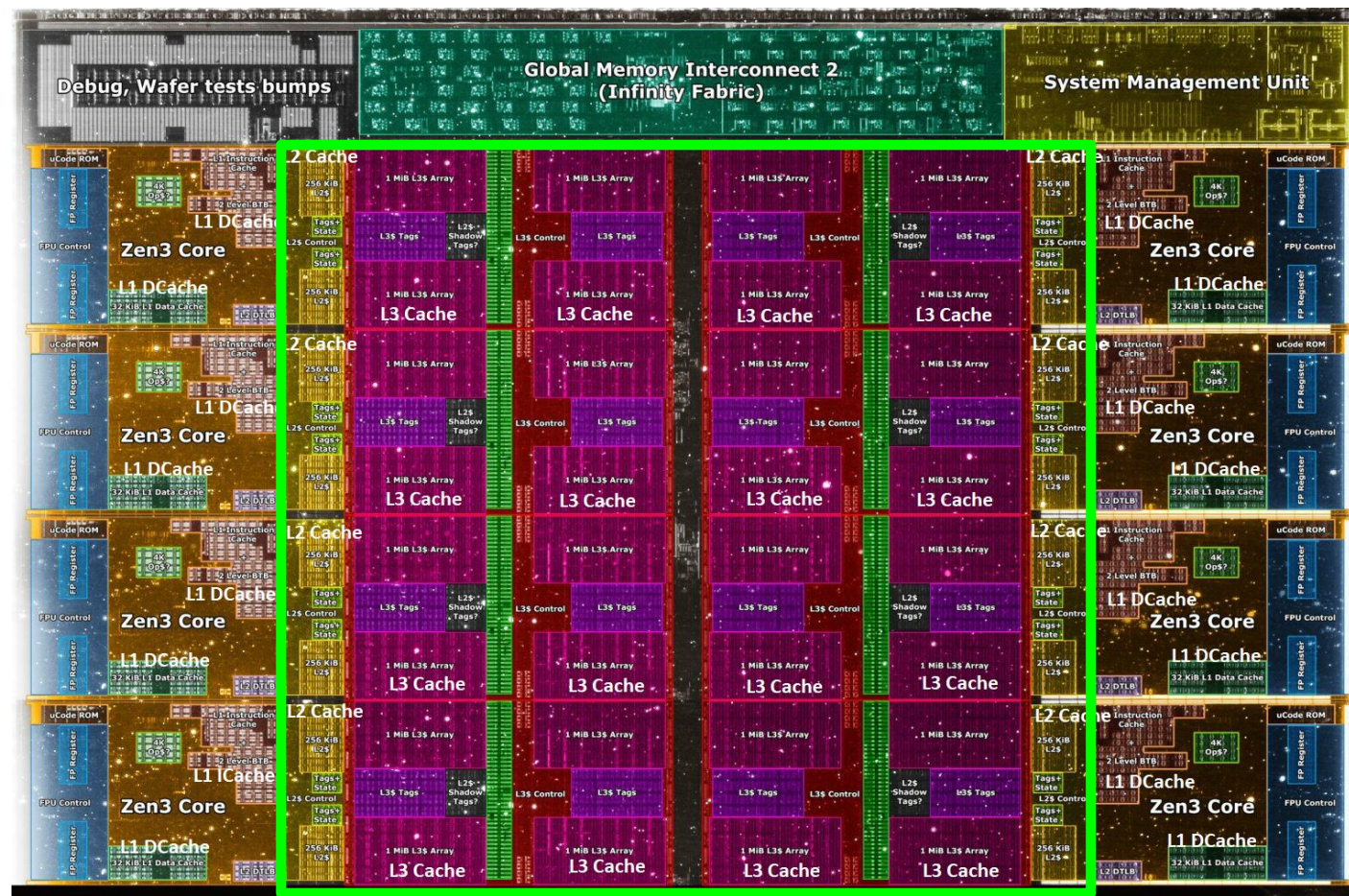


A Large Fraction of Modern Chips is Memory



Apple M1,
2021

Hệ thống phân cấp bộ nhớ



Core Count:

8 cores/16 threads

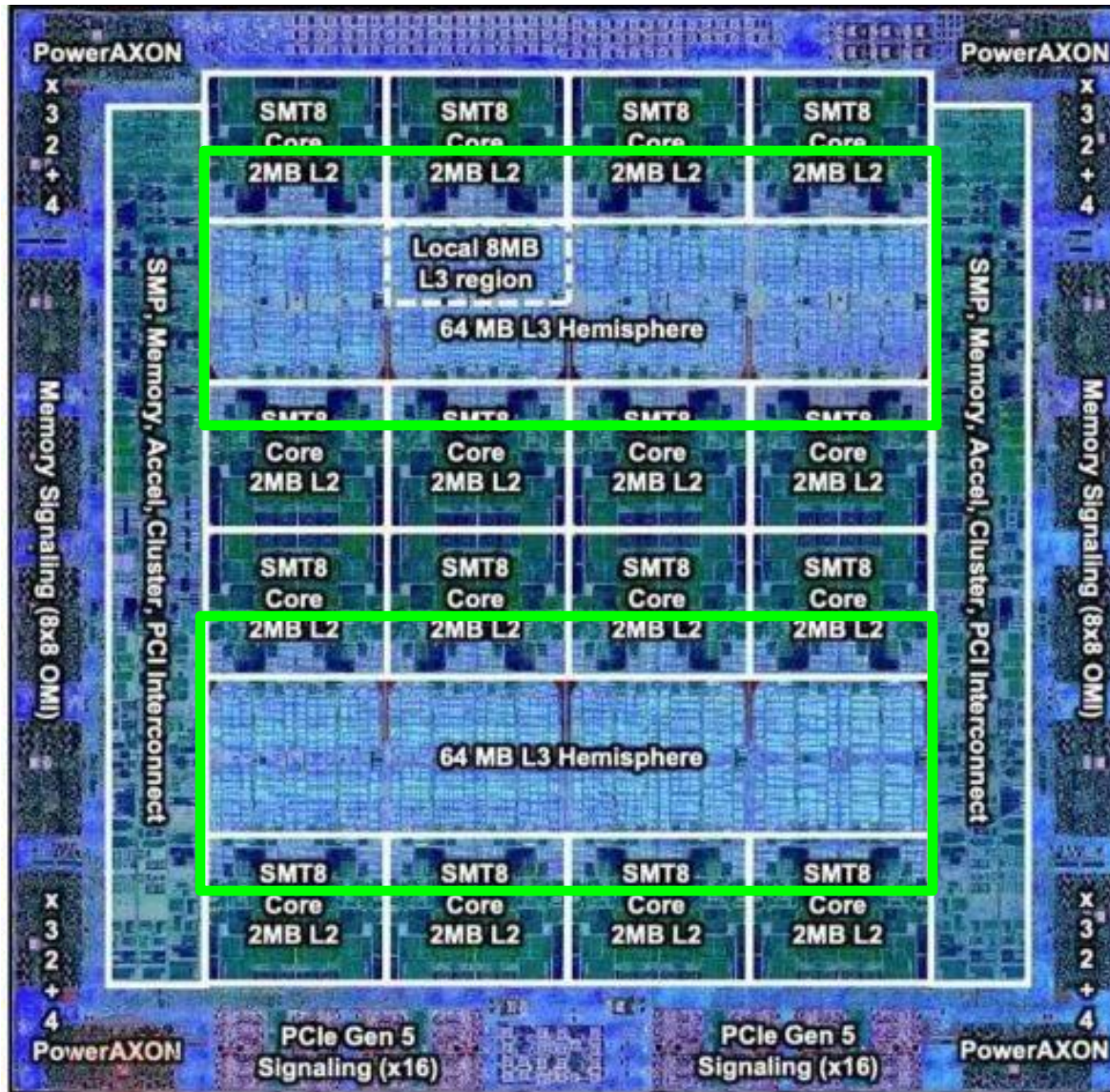
L1 Caches:
32 KB per core

L2 Caches:
512 KB per core

L3 Cache:
32 MB shared

AMD Ryzen 5000, 2020

Hệ thống phân cấp bộ nhớ



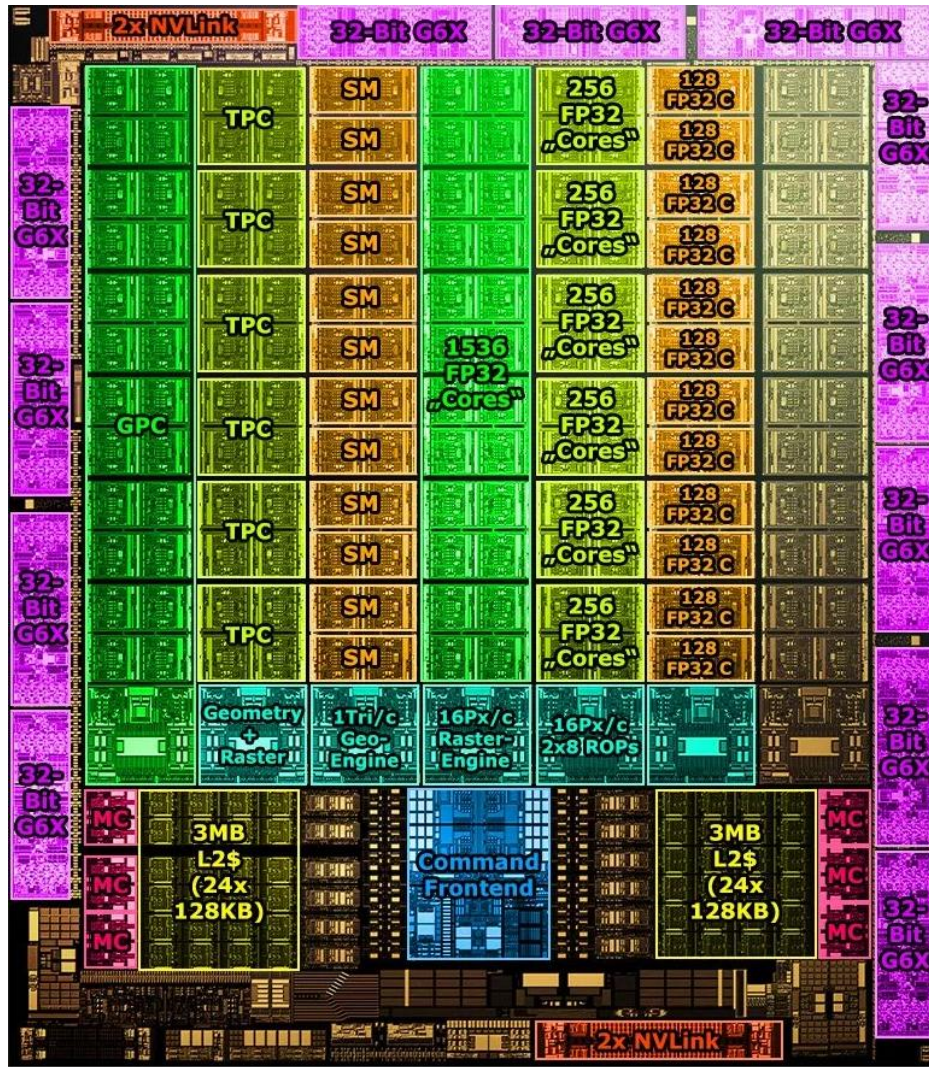
IBM POWER10,
2020

Cores:
15-16 cores,
8 threads/core

L2 Caches:
2 MB per core

L3 Cache:
120 MB shared

Hệ thống phân cấp bộ nhớ



Nvidia Ampere, 2020

Cores:

128 Streaming Multiprocessors

L1 Cache or Scratchpad:

192KB per SM

Can be used as L1 Cache and/or Scratchpad

L2 Cache:

40 MB shared

Ý tưởng về bộ nhớ

- Thời gian truy cập bằng không (độ trễ)
- Hiệu năng vô hạn
- Chi phí bằng không
- Băng thông vô hạn (để hỗ trợ nhiều truy cập song song)

Bài toán

- Các yêu cầu của bộ nhớ lý tưởng có sự đối lập
- Lớn hơn thì chậm hơn Bigger is slower
 - Bộ nhớ lớn → Mất nhiều thời gian để xác định vị trí
- Nhanh hơn thì đắt hơn Faster is more expensive
 - Công nghệ bộ nhớ : SRAM vs. DRAM vs. SSD vs. Disk vs. Tape
- Băng thông cao thì càng đắt
 - Cần nhiều nhiều cổng hơn, nhiều kênh hơn, tần số cao hơn hoặc công nghệ nhanh hơn

Bài toán

■ Bigger is slower

- ❑ SRAM, 512 Bytes, sub-nanosec
- ❑ SRAM, KByte~MByte, ~nanosec
- ❑ DRAM, Gigabyte, ~50 nanosec
- ❑ PCM-DIMM (Intel Optane DC DIMM), Gigabyte, ~200 nanosec
- ❑ PCM-SSD (Intel Optane SSD), Gigabyte, ~10 μ s
- ❑ Flash memory, Gigabyte~Terabyte, ~100 μ s
- ❑ Hard Disk, Terabyte, ~10 millisec

■ Faster is more expensive (dollars and chip area)

- ❑ SRAM, < 0.3\$ per Megabyte
- ❑ DRAM, < 0.03\$ per Megabyte
- ❑ PCM-DIMM (Intel Optane DC DIMM), < 0.004\$ per Megabyte
- ❑ PCM-SSD, < 0.001\$ per Megabyte
- ❑ Flash memory, < 0.00008\$ per Megabyte
- ❑ Hard Disk, < 0.00003\$ per Megabyte
- ❑ Các giá trị này (~ năm 2021) sẽ thay đổi tỷ lệ theo thời gian

■ Các công nghệ khác

- ❑ MRAM, RRAM, STT-MRAM, ... (chưa hoàn chỉnh)

Vấn đề

Bigger is slower

Memory Device	Capacity	Latency	Cost per Megabyte
SRAM	512 Bytes	sub-nanosec	
SRAM	KByte~MByte	~nanosec	< 0.3\$
DRAM	Gigabyte	~50 nanosec	< 0.03\$
PCM-DIMM (Intel Optane DC DIMM)	Gigabyte	~200 nanosec	< 0.004\$
PCM-SSD (Intel Optane SSD)	Gigabyte ~Terabyte	~10 μ s	< 0.001\$
Flash memory	Gigabyte ~Terabyte	~100 μ s	< 0.00008\$
Hard Disk	Terabyte	~10 millisec	< 0.00003\$

Faster is more expensive
(dollars and chip area)

These sample values (circa ~2021) scale with time

Các thành phần phân cấp bộ nhớ

□ Thanh ghi của CPU:

- Kích thước rất nhỏ (vài chục byte tới vài KB)
- Tốc độ rất nhanh, thời gian truy cập khoảng 0.25 ns
- Giá thành đắt
- Lưu trữ tạm thời dữ liệu đầu vào và ra cho các lệnh

□ Cache:

- Kích thước nhỏ (64KB tới 16MB)
- Tốc độ nhanh, thời gian truy cập khoảng 1 – 5ns
- Giá thành đắt
- Lưu trữ lệnh và dữ liệu cho CPU
- Còn được gọi là “bộ nhớ thông minh” (smart memory)

Các thành phần phân cấp bộ nhớ

□ Bộ nhớ chính:

- Kích thước lớn, dung lượng từ 256MB tới 4GB cho các hệ 32bits
- Tốc độ chậm, thời gian truy cập từ 50 – 70ns
- Lưu trữ lệnh và dữ liệu cho hệ thống và người dùng
- Giá thành rẻ

□ Bộ nhớ phụ:

- Kích thước rất lớn, dung lượng từ 20GB tới 1000GB
- Tốc độ rất chậm, thời gian truy cập khoảng 5ms
- Lưu trữ lượng dữ liệu lớn dưới dạng file trong thời gian lâu dài
- Giá thành rất rẻ

Vai trò của mô hình phân cấp

□ Nâng cao hiệu năng hệ thống:

- Dung hòa được CPU có tốc độ cao với bộ nhớ chính và bộ nhớ phụ có tốc độ thấp
- Thời gian truy cập dữ liệu trung bình của CPU từ hệ thống bộ nhớ gần bằng thời gian truy cập cache

□ Giảm giá thành sản xuất:

- Các thành phần đắt tiền sẽ được sử dụng với dung lượng nhỏ hơn
- Các thành phần rẻ hơn được sử dụng với dung lượng lớn hơn

=> Tổng giá thành của hệ thống nhớ theo mô hình phân cấp sẽ rẻ hơn so với hệ thống nhớ không phân cấp cùng tốc độ

2. Phân loại bộ nhớ

□ Dựa vào kiểu truy cập:

- Bộ nhớ truy cập ngẫu nhiên (RAM: Random Access Memory)
- Bộ nhớ truy cập tuần tự (SAM: Serial Access Memory)
- Bộ nhớ chỉ đọc (ROM: Read Only Memory)

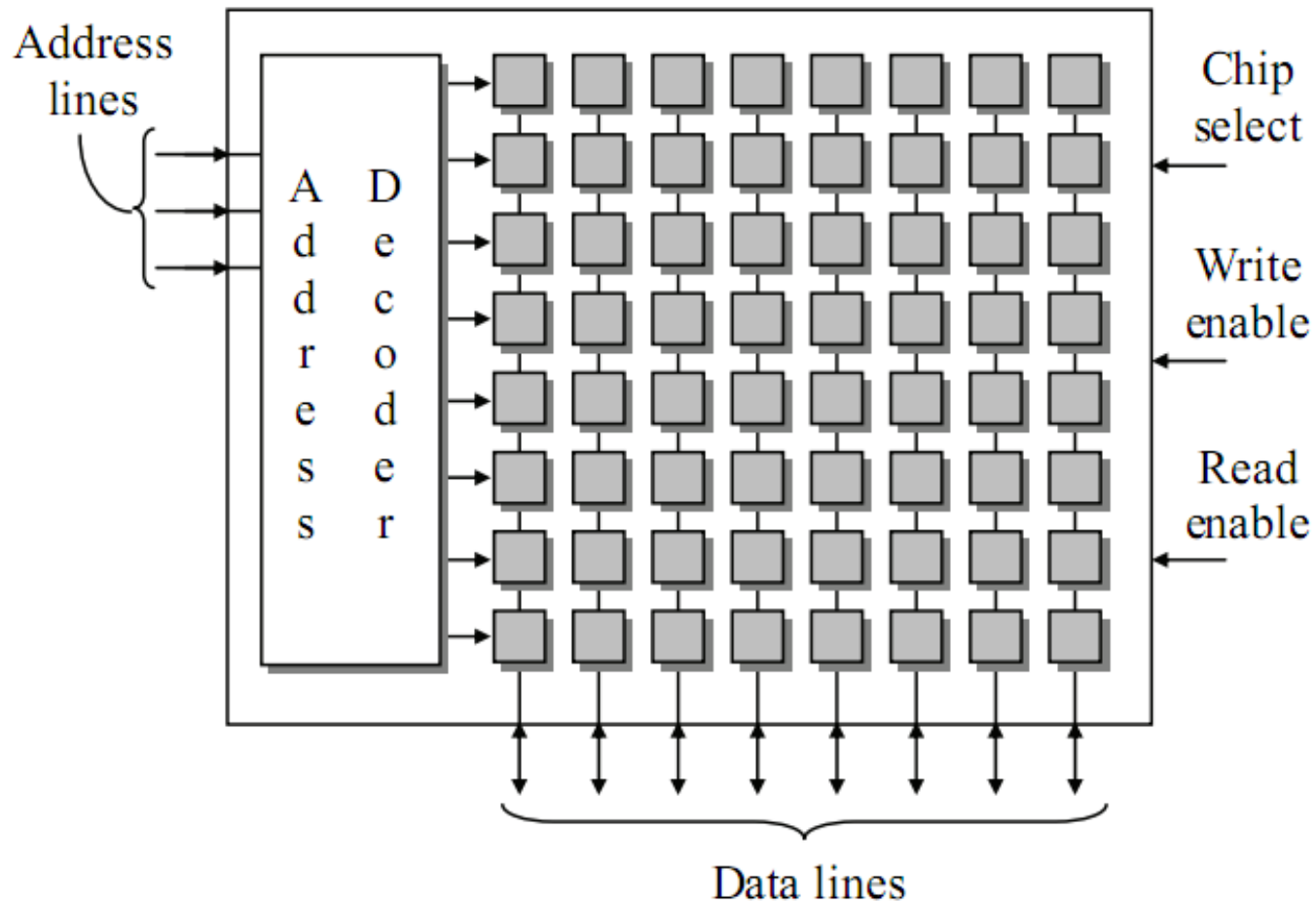
□ Dựa vào khả năng chịu đựng/ lưu giữ thông tin:

- Bộ nhớ không ổn định (volatile memory): thông tin lưu trữ bị mất khi tắt nguồn
- Bộ nhớ ổn định: thông tin lưu trữ được giữ lại khi tắt nguồn

□ Dựa vào công nghệ chế tạo:

- Bộ nhớ bán dẫn: ROM, RAM, SSD, USB, ...
- Bộ nhớ từ: HDD, FDD, tape
- Bộ nhớ quang: CD, DVD

Tổ chức mạch nhớ



Tổ chức của thiết bị nhớ

□ Address lines:

- Các đường địa chỉ nối tới bus A,
- Truyền tín hiệu địa chỉ từ CPU tới mạch nhớ

□ Address decoder:

- Bộ giải mã địa chỉ
- Sử dụng địa chỉ để chọn ra và kích hoạt ô nhớ/dòng nhớ cần truy nhập

□ Data lines:

- Các đường dữ liệu kết nối với bus D
- Truyền dữ liệu từ bộ nhớ về CPU và ngược lại

Tổ chức của thiết bị nhớ

□ Chip select CS:

- Chân tín hiệu chọn chip
- Chip nhớ được kích hoạt khi $CS=0$. Thông thường CPU chỉ làm việc với 1 chip nhớ tại 1 thời điểm

□ Write enable WE:

- Chân tín hiệu cho phép ghi
- Cho phép ghi vào đường nhớ khi $WE = 0$

□ Read enable RE:

- Chân tín hiệu cho phép đọc
- Cho phép đọc dữ liệu từ đường nhớ khi $RE = 0$

Bộ nhớ chính

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)	Bộ nhớ hầu như chỉ đọc	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		
Random Access Memory (RAM)		bằng điện, mức từng byte	Bảng điện	Khả biến

3. Bộ nhớ ROM

- ROM (Read Only Memory): là bộ nhớ chỉ đọc
 - Ghi thông tin vào ROM bằng cách sử dụng các thiết bị hoặc phương pháp đặc biệt
- ROM là bộ nhớ ổn định: tất cả thông tin vẫn được duy trì khi mất nguồn nuôi
- Là bộ nhớ bán dẫn: mỗi ô nhớ là một cổng bán dẫn
- Thường dùng để lưu trữ thông tin hệ thống: thông tin phần cứng và BIOS

Các loại ROM

- ROM nguyên thủy (Ordinary ROM):
 - ROM các thế hệ đầu tiên
 - Sử dụng tia cực tím để ghi thông tin
- PROM (Programmable ROM)
 - ROM có thể lập trình
 - Thông tin có thể được ghi vào PROM nhờ thiết bị đặc biệt gọi là bộ lập trình PROM



Các loại ROM

- EPROM (Erasable programmable read-only memory)
 - Là ROM có thể lập trình và xóa được
 - Thông tin trong EPROM có thể xóa bằng cách chiếu các tia cực tím có cường độ cao
- EEPROM (Electrically Erasable PROM :
 - Là PROM có thể xóa được thông tin bằng điện
 - Có thể ghi được thông tin sử dụng phần mềm chuyên dụng

4. Bộ nhớ RAM

- RAM (Random Access Memory) là bộ nhớ truy cập ngẫu nhiên
 - Mỗi ô nhớ có thể được truy cập một cách ngẫu nhiên không theo trật tự nào
 - Tốc độ truy cập các ô nhớ là tương đương
- Là bộ nhớ không ổn định (dễ bay hơi): mọi thông tin lưu trữ sẽ bị mất khi tắt nguồn
- Là bộ nhớ bán dẫn. Mỗi ô nhớ là một cổng bán dẫn
- Sử dụng để lưu trữ thông tin hệ thống và của người dùng
 - Thông tin hệ thống: thông tin phần cứng & hệ điều hành
 - Thông tin người dùng: mã lệnh và dữ liệu các chương trình ứng dụng

Các loại RAM

□ RAM tĩnh (SRAM):

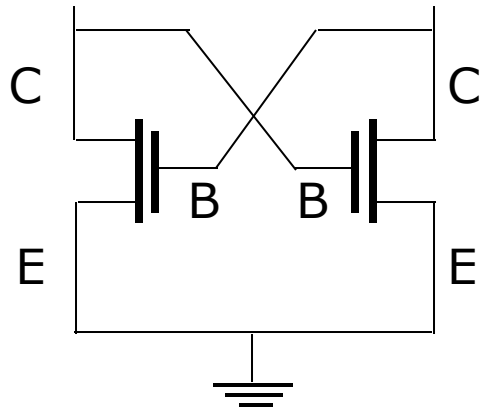
- Mỗi bit của SRAM là một mạch lật flip-flop
- Thông tin lưu trong các bit SRAM luôn ổn định, không cần phải làm tươi định kỳ
- SRAM nhanh nhưng đắt hơn DRAM

□ RAM động (DRAM):

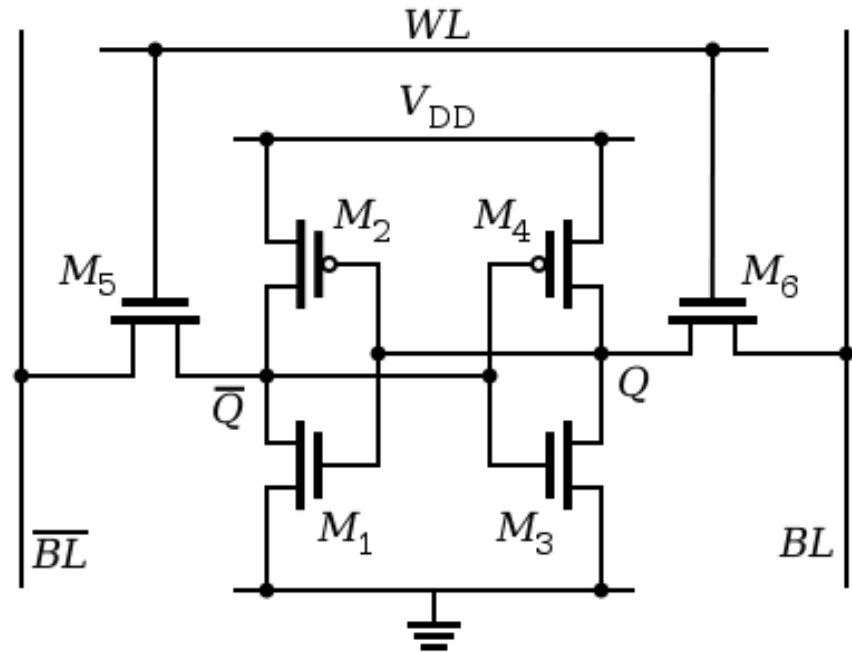
- Mỗi bit của DRAM dựa trên tụ điện
- Thông tin lưu trong bit DRAM không được ổn định, và phải được làm tươi định kỳ
- DRAM chậm hơn SRAM nhưng rẻ hơn



Cấu tạo SRAM



Một mạch lật đơn giản

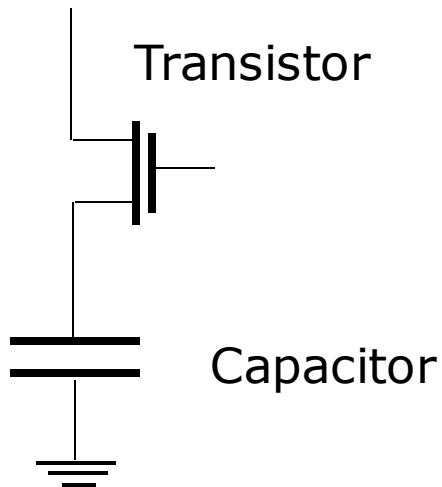


Một ô nhớ SRAM loại 6T

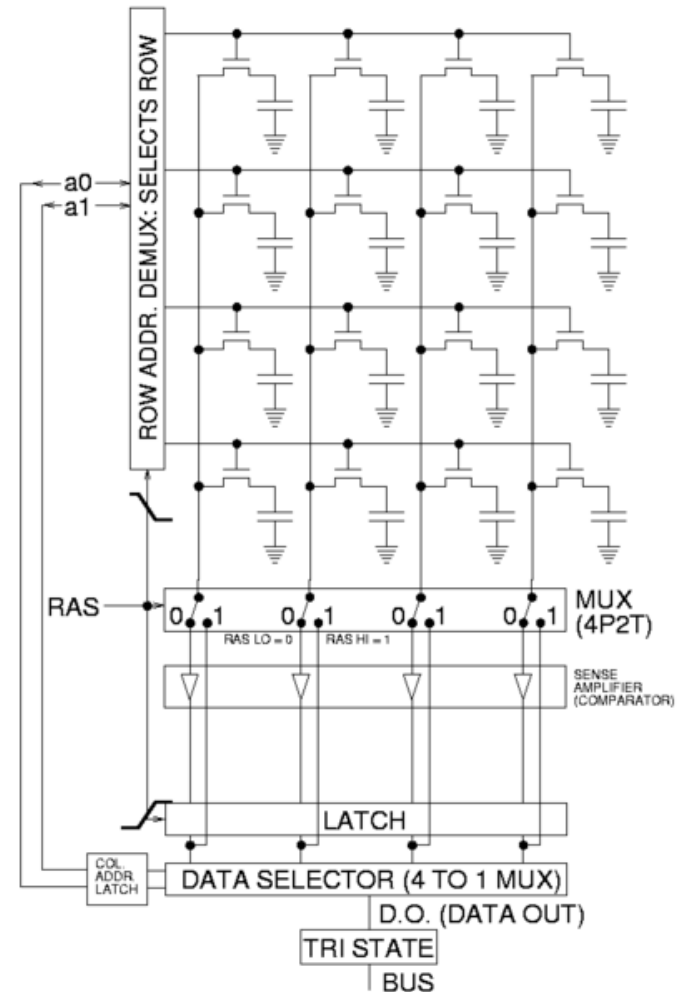
Các đặc điểm của SRAM

- SRAM sử dụng mạch lật trigơ lưỡng ổn (bistable latching circuit) để lưu 1 bit thông tin
- Mỗi mạch lật lưu 1 bit thường sử dụng 6, 8, 10 transistor (gọi là mạch 6T, 8T, 10T)
- SRAM thường có tốc độ truy cập nhanh vì:
 - Các bit có cấu trúc đối xứng
 - Các mạch nhớ SRAM chấp nhận tất cả các chân địa chỉ tại một thời điểm (không dồn kênh)
- SRAM đắt vì:
 - Nó sử dụng nhiều transistor cho một bit hơn DRAM
 - Vì cấu trúc bên trong phức tạp hơn, mật độ của SRAM thấp hơn DRAM

Cấu tạo DRAM



Một bit DRAM



Các đặc điểm của DRAM

- Bit của DRAM dựa trên 1 tụ điện và 1 transistor.
 - 2 mức tích điện của tụ sẽ biểu diễn 2 mức logic:
 - Không tích điện: mức 0
 - Tích đầy điện: mức 1
- Do tụ thường tự phóng điện, điện tích trong tụ điện có xu hướng bị tổn hao
 - Cần nạp lại thông tin trong tụ thường xuyên để tránh mất thông tin
 - Việc nạp lại thông tin cho tụ là quá trình làm tươi (refresh), phải theo định kỳ

Các đặc điểm của DRAM

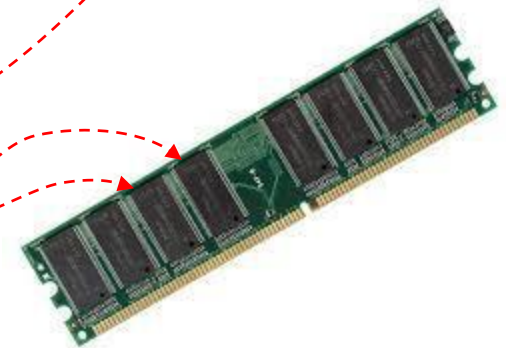
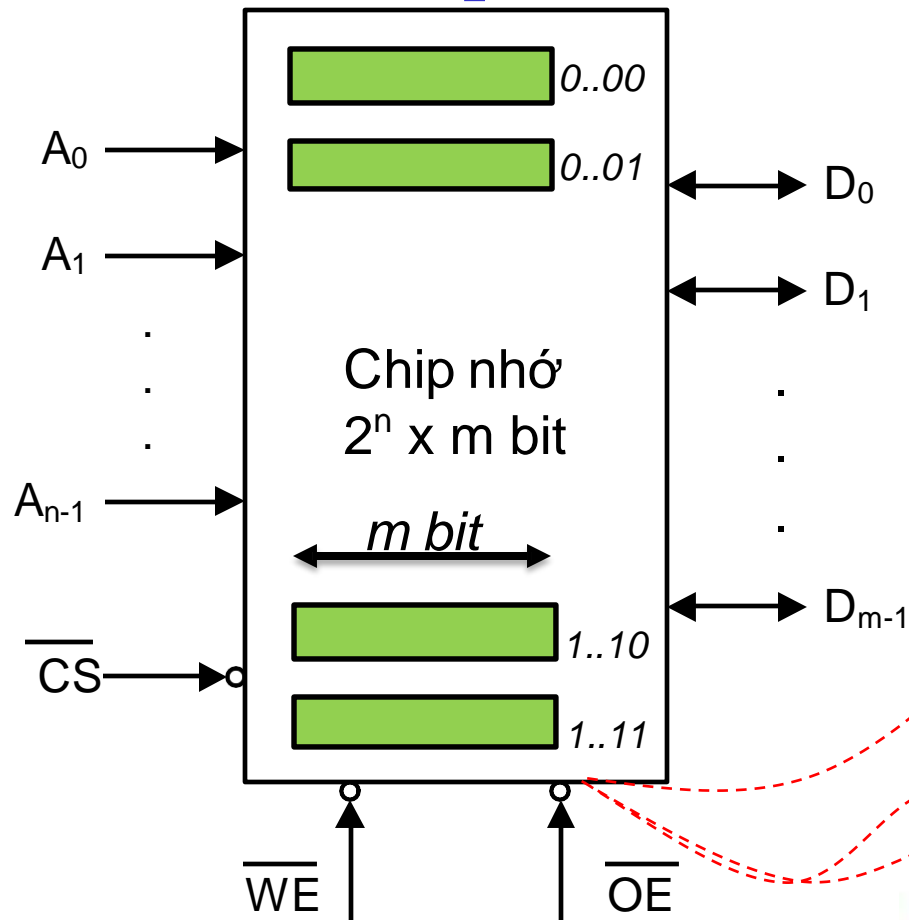
- Các bit nhớ của DRAM thường được sắp xếp thành ma trận:
 - Một tụ + một transistor \rightarrow một bit
 - Các bit được tập hợp thành các cột và dòng
- DRAM chậm hơn SRAM vì:
 - Cần làm tươi định kỳ
 - Quá trình nạp điện tụ mất thời gian
 - Các mạch DRAM thường dùng kỹ thuật dồn kênh (địa chỉ cột/ hàng) để tiết kiệm đường địa chỉ
- DRAM rẻ hơn SRAM vì:
 - Cấu trúc đơn giản, sử dụng ít transistor
 - Mật độ cấy cao hơn

Các loại DRAM

- SDRAM: Synchronous DRAM
- SRD SDRAM: (Single Data Rate SDRAM) chấp nhận 1 thao tác đọc/ ghi và chuyển 1 từ dữ liệu trong 1 chu kỳ đồng hồ; tốc độ 100MHz, 133MHz
- DDR SDRAM: Double Data Rate SDRAM
 - DDR1 SDRAM: DDR 266, 333, 400: có khả năng chuyển 2 từ dữ liệu trong 1 chu kỳ
 - DDR2 SDRAM: DDR2 400, 533, 800: có khả năng chuyển 4 từ dữ liệu trong 1 chu kỳ
 - DDR3 SDRAM: DDR3 800, 1066, 1333, 1600: có khả năng chuyển 8 từ dữ liệu trong 1 chu kỳ

Tổ chức của chip nhớ

- Sơ đồ cơ bản của chip nhớ



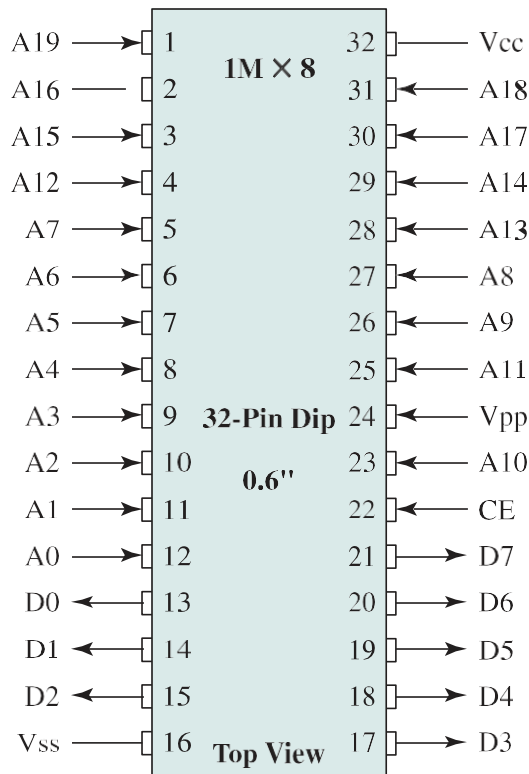
Các tín hiệu của chip nhớ

- 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)

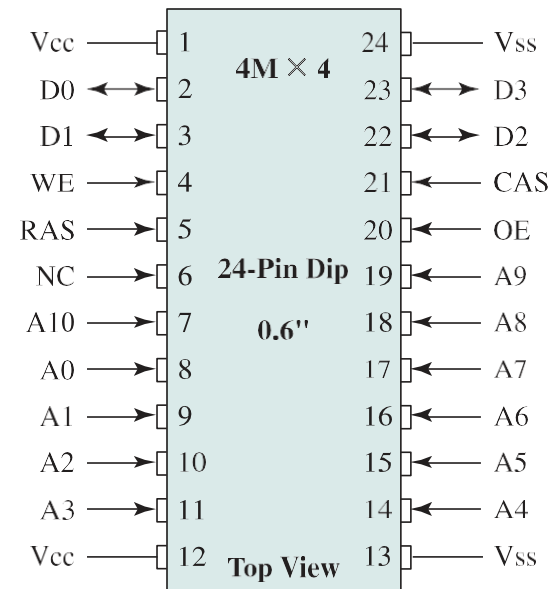
Tổ chức của DRAM

- Dùng n đường địa chỉ dồn kênh \rightarrow 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

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

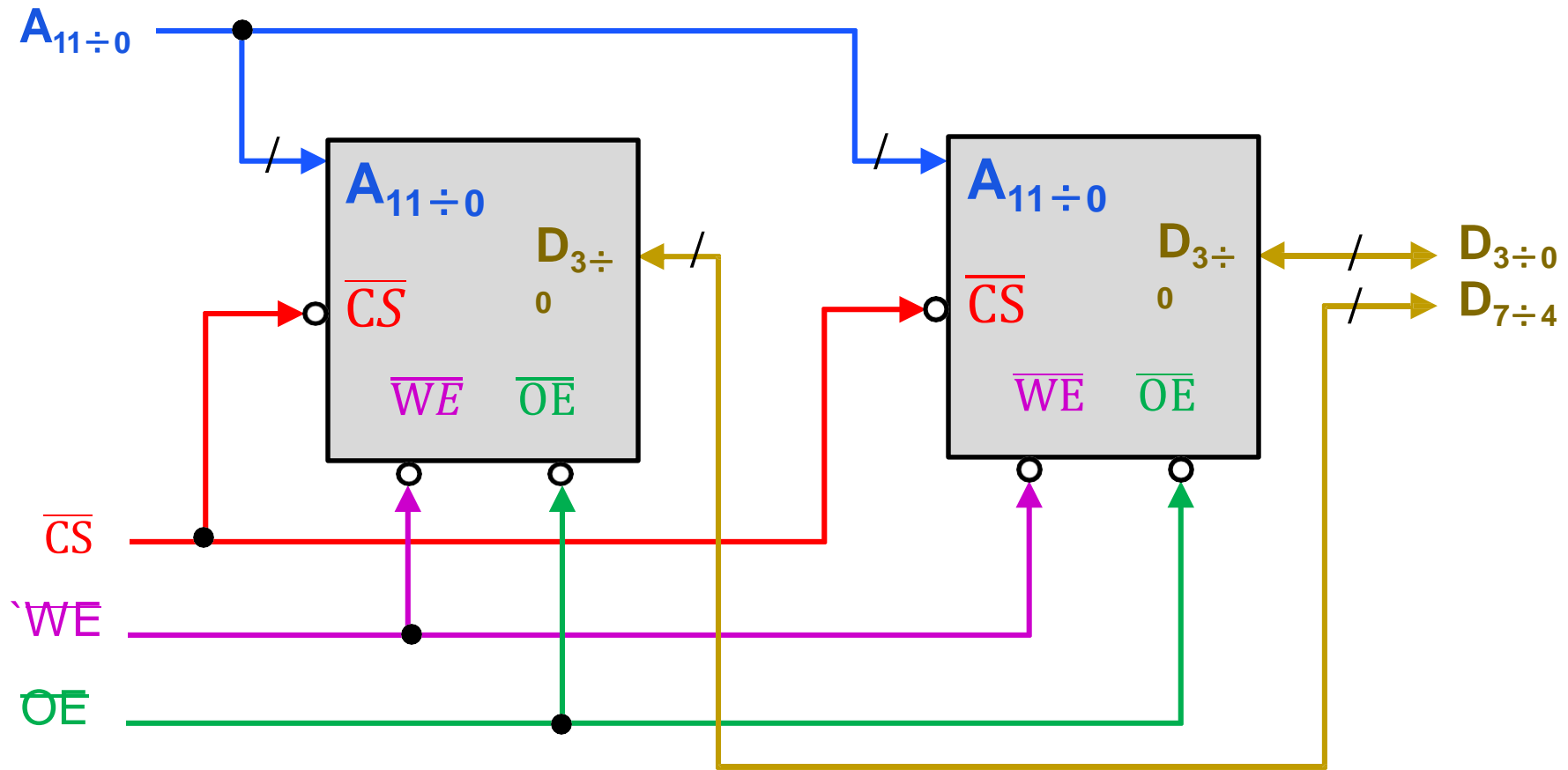
VD1:

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

Giải:

- Dung lượng chip nhớ = $2^{12} \times 4$ bit
- chip nhớ có:
 - 12 chân địa chỉ
 - 4 chân dữ liệu
- mô-đun nhớ cần có: ($= 2^{12} \times 8$ bit)
 - 12 chân địa chỉ
 - 8 chân dữ liệu

Ví dụ tăng độ dài từ nhớ



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

Tăng số lượng từ nhớ

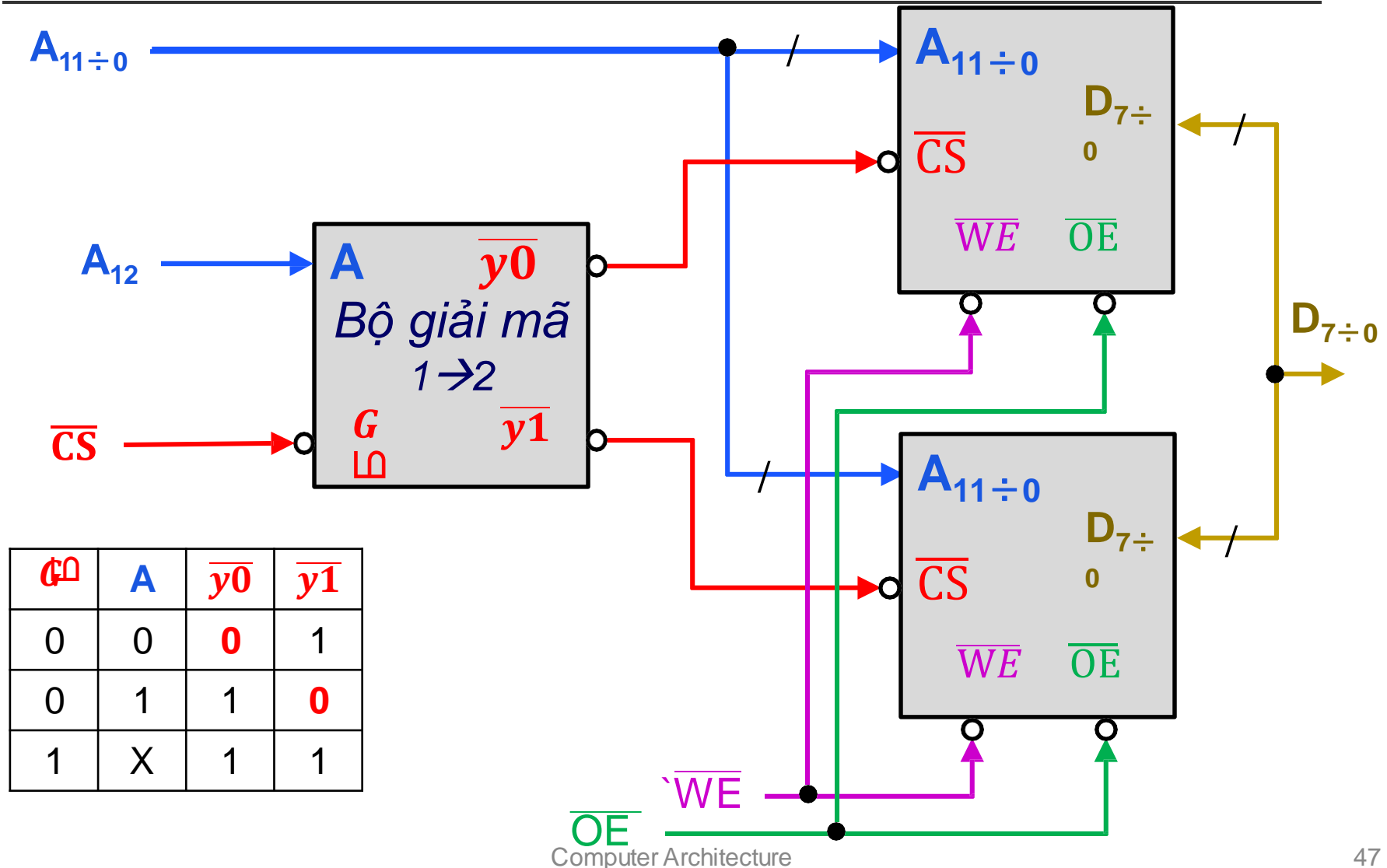
VD2:

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

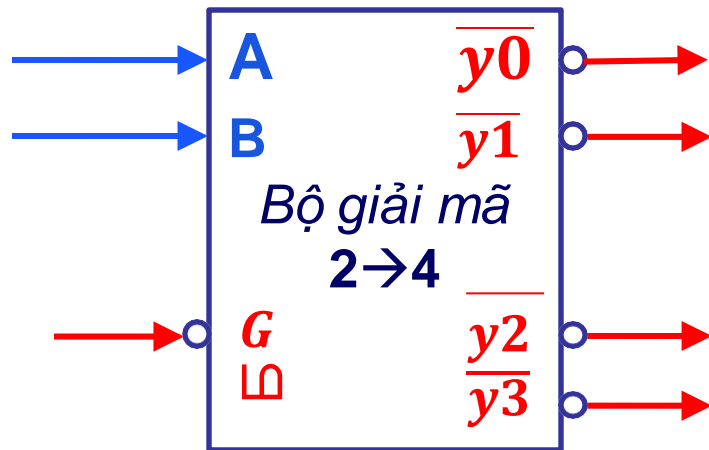
Giải:

- Dung lượng chip nhớ = $2^{12} \times 8$ bit
- chip nhớ có:
 - 12 chân địa chỉ
 - 8 chân dữ liệu
- Dung lượng mô-đun nhớ = $2^{13} \times 8$ bit
 - 13 chân địa chỉ
 - 8 chân dữ liệu

Tăng số lượng từ nhớ



Bộ giải mã $2 \rightarrow 4$



G \overline{G}	Vào		Ra			
	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

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
- Tự vẽ sơ đồ thiết kế

Đặc trưng cơ bản của bộ nhớ

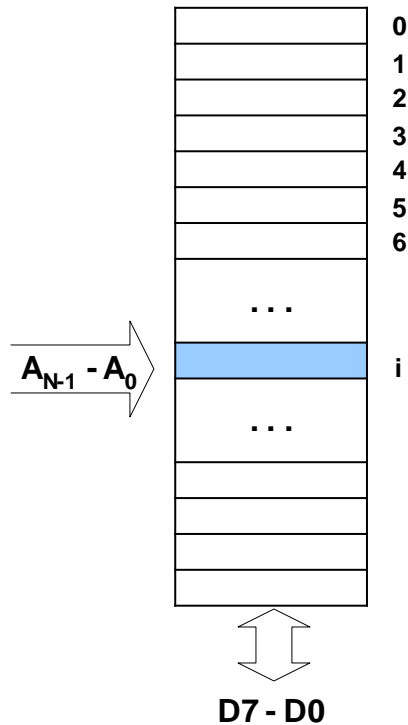
- Chứa các chương trình đang thực hiện và các dữ liệu đang được sử dụng
- Tồn tại trên mọi hệ thống máy tính
- Bao gồm các ngăn nhớ được đánh địa chỉ trực tiếp bởi CPU
- Dung lượng của bộ nhớ chính nhỏ hơn không gian địa chỉ bộ nhớ mà CPU quản lý.
- Việc quản lý logic bộ nhớ chính tùy thuộc vào hệ điều hành

Tổ chức bộ nhớ đan xen (interleaved memory)

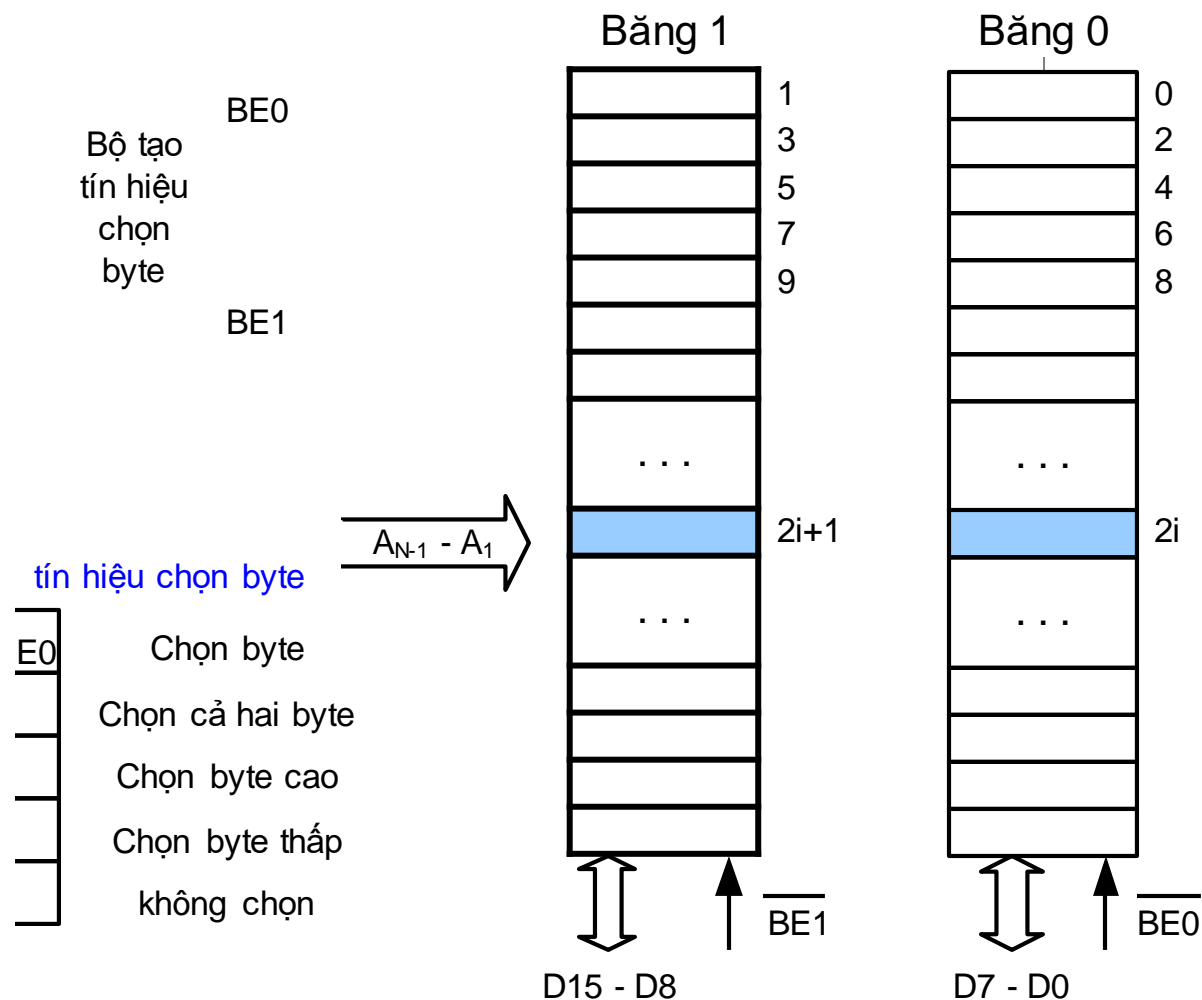
- Độ rộng của bus dữ liệu để trao đổi với bộ nhớ: $m = 8, 16, 32, 64, 128 \dots$ bit
- Các ngăn nhớ được tổ chức theo byte
→ tổ chức bộ nhớ vật lý khác nhau

Tổ chức bộ nhớ đơn xen

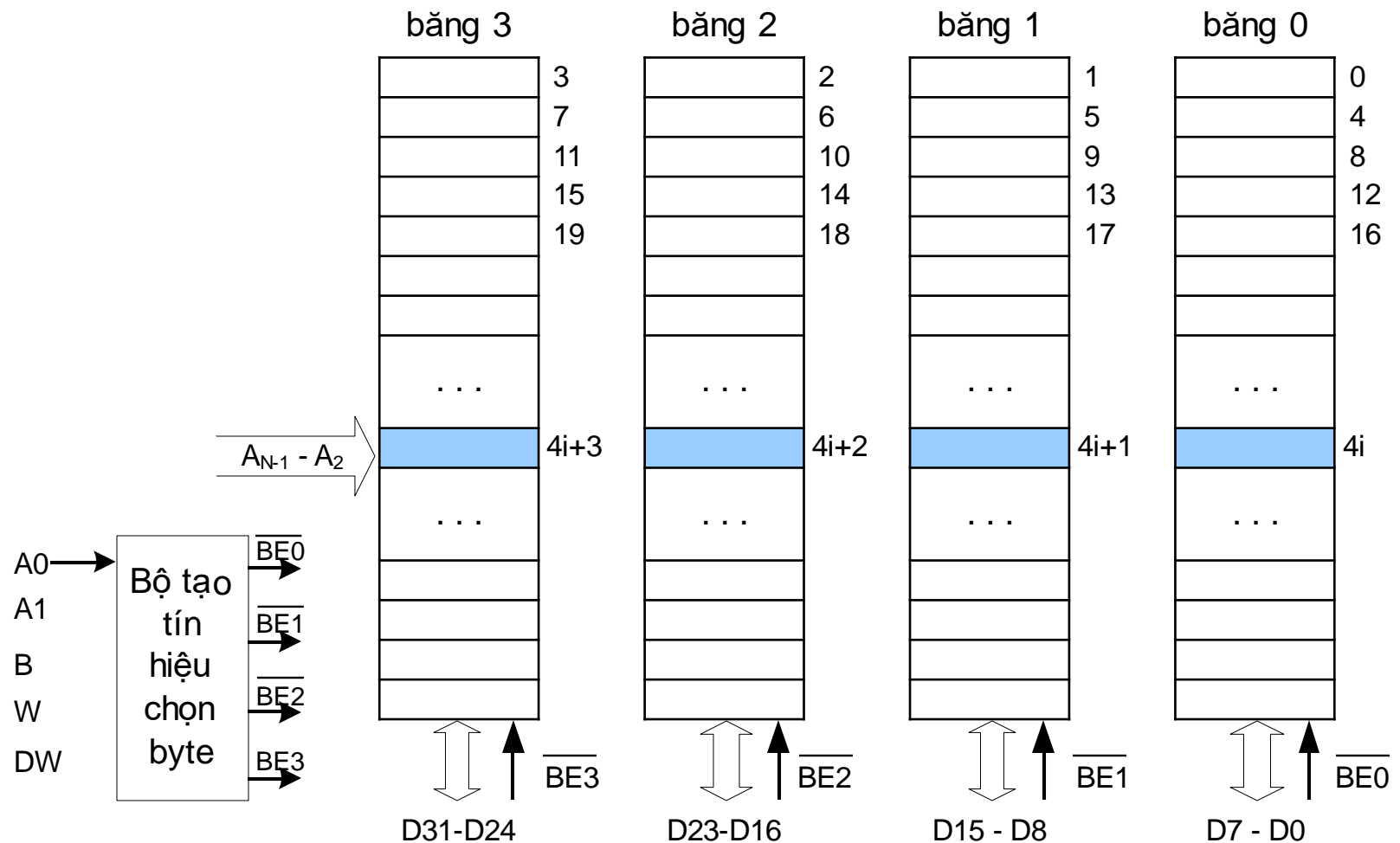
$m=8\text{bit} \rightarrow$ một băng nhớ tuyến tính



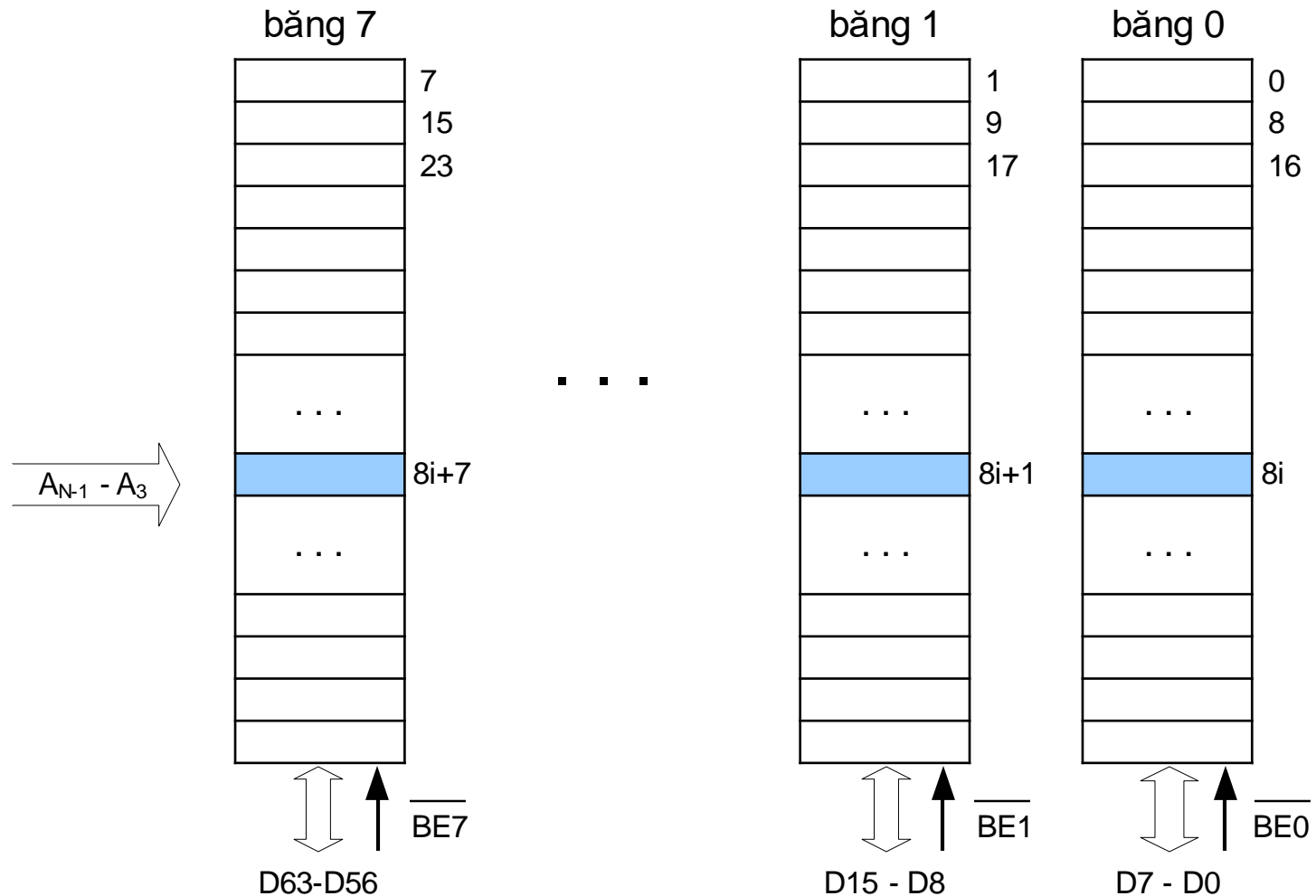
$m = 16\text{bit} \rightarrow$ hai băng nhớ đan xen



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



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



HWiNFO32 v4.48-2350

Program
Report
Benchmarks
Monitoring
Help

Summary

Save Report

Benchmarks

Sensors

About

Driver Update

BIOS Update

TiENND10

Central Processor(s)

Motherboard

Memory

Row: 0 - 4096 MB PC3-12800 DDR3 SDRAM Samsung M47

Row: 1 - 4096 MB PC3-12800 DDR3 SDRAM Samsung M47

Bus

Video Adapter

Monitor

Drives

Audio

Network

Ports

Smart Battery

Feature

General Module Information

Module Number:

Module Size:

Memory Type:

Module Type:

Memory Speed:

Module Manufacturer:

Module Part Number:

Module Revision:

Module Serial Number:

Module Manufacturing Date:

Module Manufacturing Location:

SDRAM Manufacturer:

Error Check/Correction:

Description

0

4096 MBytes

DDR3 SDRAM

SO-DIMM

800.0 MHz (DDR3-1600 / PC3-12800)

Samsung

M471B5173DB0-YK0

0

395094995

Year: 2014, Week: 48

2

Samsung

None

Module characteristics

Row Address Bits:

Column Address Bits:

Number Of Banks:

Module Density:

Number Of Ranks:

Device Width:

Bus Width:

Module Nominal Voltage (VDD):

16

10

8

4096 Mb

1

8 bits

64 bits

1.5 V, 1.35 V

Module timing

Minimum SDRAM Cycle Time (tCKmin):

CAS# Latencies Supported:

Minimum CAS# Latency Time (tAmin):

Minimum RAS# to CAS# Delay (tRCDmin):

Minimum Row Precharge Time (tRpmn):

Minimum Active to Precharge Time (tRASmin):

1.250 ns

5, 6, 7, 8, 9, 10, 11

13.125 ns

13.125 ns

13.125 ns

35.000 ns

Supported Module Timing at 800.0 MHz:

Supported Module Timing at 733.3 MHz:

Supported Module Timing at 666.7 MHz:

Supported Module Timing at 600.0 MHz:

11-11-11-28

10-10-10-26

9-9-9-24

8-8-8-21

CPU-Z

CPU

Caches

Mainboard

Memory

SPD

Graphics

About

General

Type

Channel #

Size

DC Mode

NB Frequency

DDR3

Dual

8 GBytes

2089.7 MHz

Timings

DRAM Frequency

FSB:DRAM

CAS# Latency (CL)

RAS# to CAS# Delay (tRCD)

RAS# Precharge (tRP)

Cycle Time (tRAS)

Row Refresh Cycle Time (tRFC)

Command Rate (CR)

DRAM Idle Timer

Total CAS# (tRDRAM)

Row To Column (tRCD)

796.1 MHz

1:6

11.0 clocks

11 clocks

11 clocks

28 clocks

208 clocks

1T

CPU-Z

Ver. 1.71.1.x32

Tools

Validate

OK

sung M471B5173DB0-YK0

5. Bộ nhớ cache

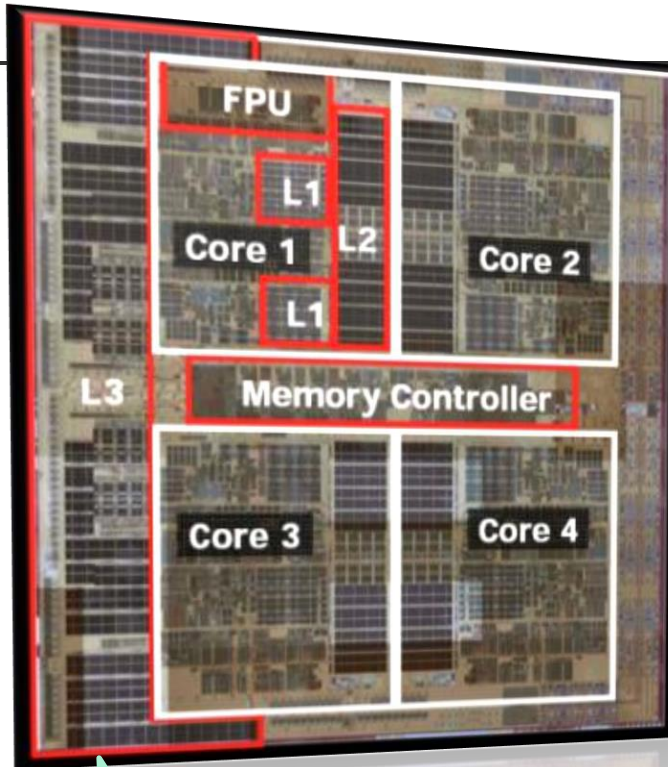
- Cache là gì?
- Vai trò của cache
- Các nguyên lý cơ bản hoạt động của cache
- Kiến trúc cache
- Tổ chức cache
- Đọc/ ghi trong cache
- Các chính sách thay thế
- Cách thức để nâng cao hiệu năng cache

Cache là gì?

- Cache là thành phần nhớ trong sơ đồ phân cấp bộ nhớ máy tính.
 - Nó hoạt động như thành phần trung gian, trung chuyển dữ liệu từ bộ nhớ chính về CPU và ngược lại
- Vị trí của cache:
 - Với các hệ thống cũ, cache thường nằm ngoài CPU
 - Với các CPU mới, cache thường được tích hợp vào trong CPU



Lịch sử hình thành cache



AMD Athlon

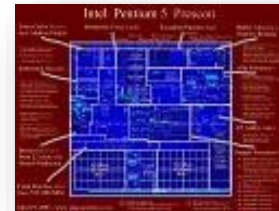
Cache chiếm
nhiều diện tích



■ Intel Quad Core



■ Intel Core i7



■ Intel Pentium 5

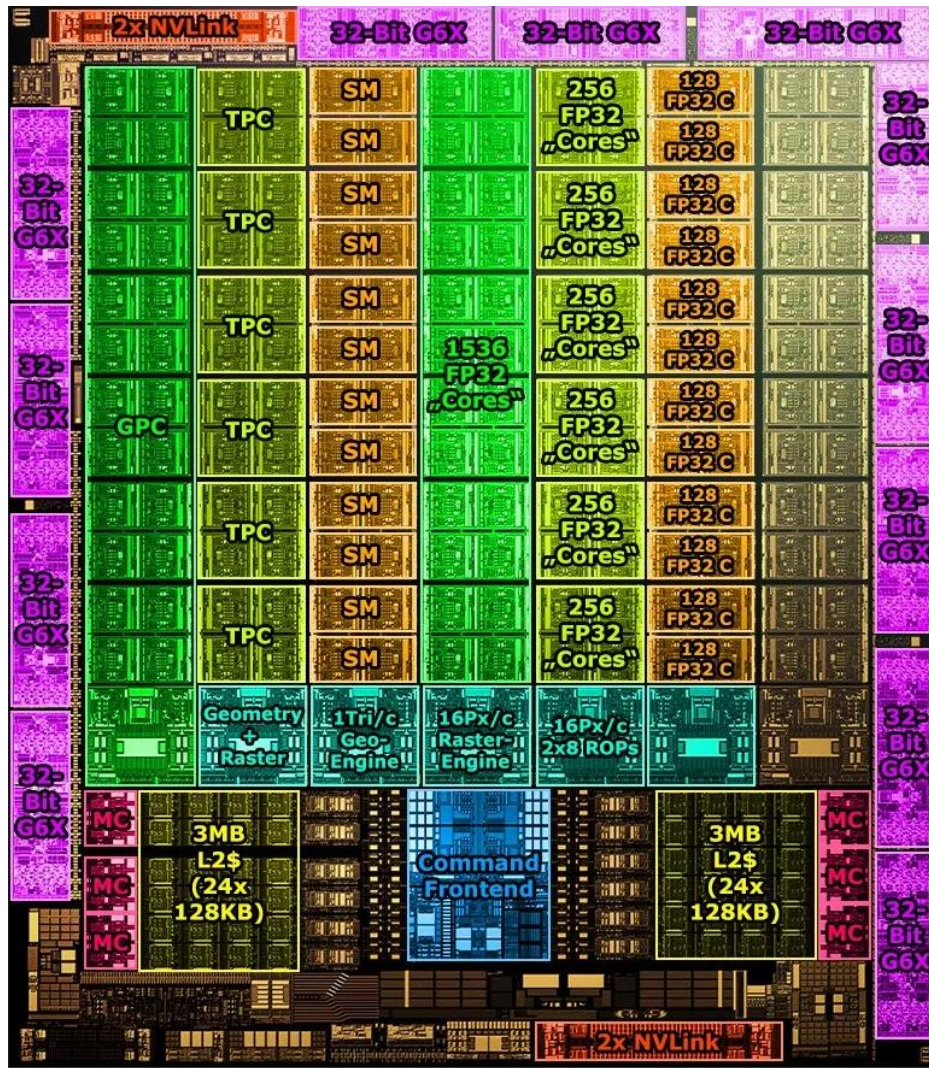


■ AMD Quad Core

```
Intel SNB 2295 MHz
L1 Cache: 32K 76495 MB/s
L2 Cache: 256K 34770 MB/s
L3 Cache: 3072K 26077 MB/s
Memory : 8170M 16047 MB/s
IMC : Intel(R) Core(TM) i5-24
Settings: RAM : 0 MHz (DDR3- 0)

752K e820
```

Caches and Scratchpad in a Modern GPU



Nvidia Ampere, 2020

Cores:

128 Streaming Multiprocessors

L1 Cache or Scratchpad:

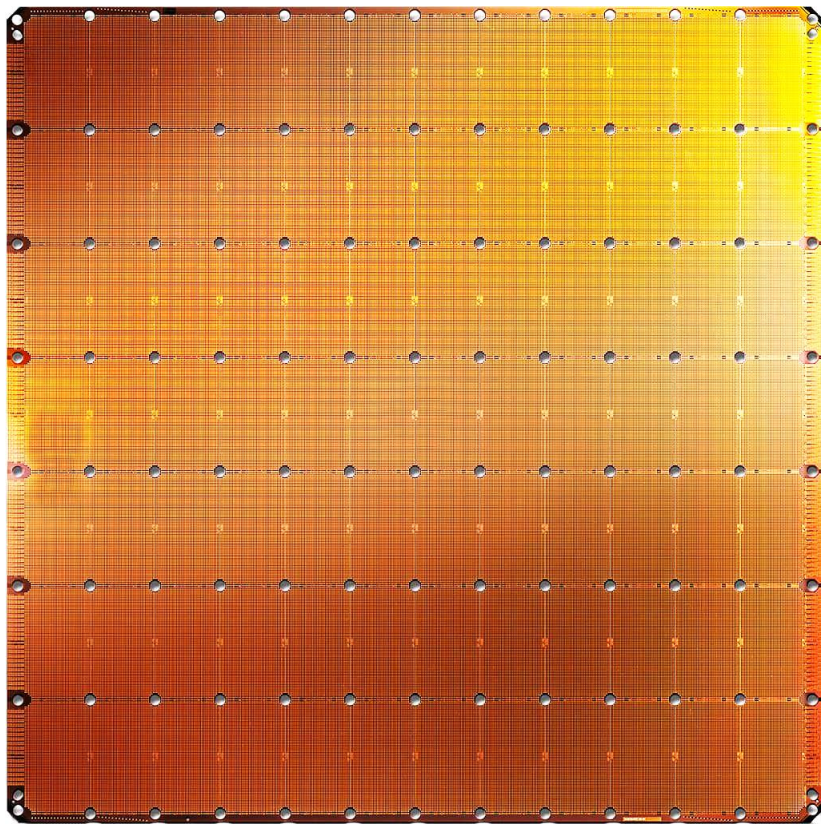
192KB per SM

Can be used as L1 Cache and/or Scratchpad

L2 Cache:

40 MB shared

Cerebras's Wafer Scale Engine (2019)



Cerebras WSE

1.2 Trillion transistors
46,225 mm²

- The largest ML accelerator chip
- 400,000 cores
- **18 GB of on-chip memory**
- **9 PB/s memory bandwidth**



Largest GPU

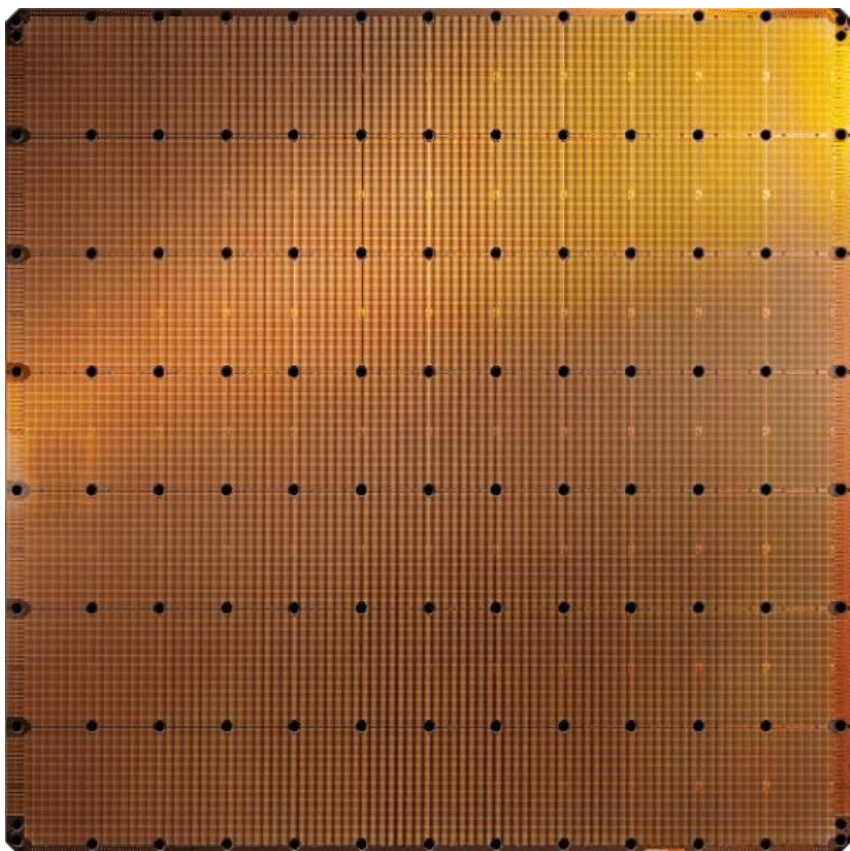
21.1 Billion transistors
815 mm²

NVIDIA TITAN V

<https://www.anandtech.com/show/14758/hot-chips-31-live-blogs-cerebras-wafer-scale-deep-learning>

<https://www.cerebras.net/cerebras-wafer-scale-engine-why-we-need-big-chips-for-deep-learning>

Cerebras's Wafer Scale Engine-2 (2021)



Cerebras WSE-2

2.6 Trillion transistors
46,225 mm²

<https://cerebras.net/product/#overview>

- The largest ML accelerator chip
- 850,000 cores
- **40 GB of on-chip memory**
- **20 PB/s memory bandwidth**



Largest GPU

54.2 Billion transistors
826 mm²

NVIDIA Ampere GA100

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

Cache là gì?

- Dung lượng cache thường nhỏ:
 - Với các hệ thống cũ: 16K, 32K,..., 128K
 - Với các hệ thống mới: 256K, 512K, 1MB, 2MB, ...
- Tốc độ truy nhập của cache nhanh hơn so với tốc độ bộ nhớ chính
- Giá thành cache (tính trên bit) thường đắt hơn so với bộ nhớ chính
- Với các hệ thống CPU mới, cache thường được chia thành nhiều mức:
 - Mức 1: 16 – 32 KB có tốc độ rất cao
 - Mức 2: 1 -16MB có tốc độ khá cao

Vai trò của Cache

□ Nâng cao hiệu năng hệ thống:

- Dung hòa giữa CPU có tốc độ cao và bộ nhớ chính tốc độ thấp (giảm số lượng truy cập trực tiếp của CPU vào bộ nhớ chính)
- Thời gian trung bình CPU truy cập hệ thống bộ nhớ gần bằng thời gian truy cập cache

□ Giảm giá thành sản xuất:

- Nếu 2 hệ thống có cùng hiệu năng thì hệ thống có cache sẽ rẻ hơn
- Nếu 2 hệ thống cùng giá thành, hệ thống có cache sẽ nhanh hơn

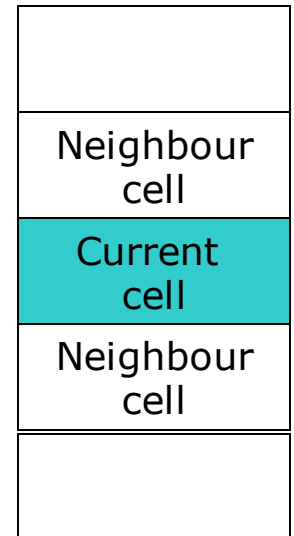
Các nguyên lý hoạt động của Cache

- Cache được coi là bộ nhớ thông minh:
 - Cache có khả năng đoán trước yêu cầu về lệnh và dữ liệu của CPU
 - Dữ liệu và lệnh cần thiết được chuyển trước từ bộ nhớ chính về cache -> CPU chỉ truy nhập cache -> giảm thời gian truy nhập bộ nhớ
- Cache hoạt động dựa trên 2 nguyên lý cơ bản:
 - Nguyên lý cục bộ/ lân cận về không gian (spatial locality)
 - Nguyên lý cục bộ/ lân cận về thời gian (temporal locality)

Các nguyên lý hoạt động của Cache

□ Cục bộ (lân cận) về không gian:

- Nếu một vị trí bộ nhớ được truy cập, thì khả năng/ xác suất các vị trí gần đó được truy cập trong thời gian gần tới là cao
- Áp dụng với các mục dữ liệu và các lệnh có thứ tự tuần tự theo chương trình
- Hầu hết các lệnh trong chương trình có thứ tự tuần tự, do đó cache đọc một khối lệnh trong bộ nhớ, mà bao gồm cả các phần tử xung quanh vị trí phần tử hiện tại được truy cập



Các nguyên lý cơ bản của Cache

□ Cục bộ (lân cận) về thời gian:

- Nếu một vị trí bộ nhớ được truy cập, thì khả năng nó sẽ được truy cập **lại** trong thời gian gần tới là cao
- Áp dụng với các mục dữ liệu và các lệnh trong vòng lặp
- Cache đọc khối dữ liệu trong bộ nhớ bao gồm tất cả các thành phần trong vòng lặp

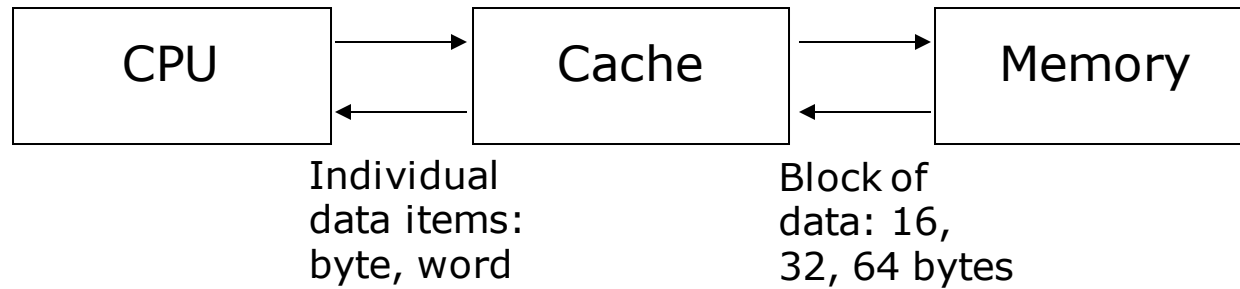
Start of
loop

Instruction 1
Instruction 2
Instruction 3
Instruction 4
Instruction 5

End of
loop

Trao đổi dữ liệu

- CPU đọc/ ghi từng mục dữ liệu riêng biệt từ/ vào cache
- Cache đọc/ ghi khối dữ liệu từ/ vào bộ nhớ



Hệ số Hit và Miss của Cache

- Hit là sự kiện CPU truy cập tới mục dữ liệu/ mục tin mà tìm được trong cache
 - Xác suất xảy ra Hit được gọi là hệ số hit H
 - $0 \leq H \leq 1$
 - H càng cao thì cache càng tốt
- Miss là sự kiện CPU truy cập tới mục dữ liệu mà không tìm thấy nó trong cache
 - Khả năng xảy ra Miss gọi là hệ số miss hay $1-H$
 - $0 \leq (1-H) \leq 1$
 - Hệ số miss thấp thì hiệu quả cache cao

Kiến trúc Cache – look aside

- Cache và bộ nhớ cùng được kết nối tới bus hệ thống
- Cache và bộ nhớ chính “thấy” chu kỳ bus CPU tại cùng một thời điểm

- Ưu:

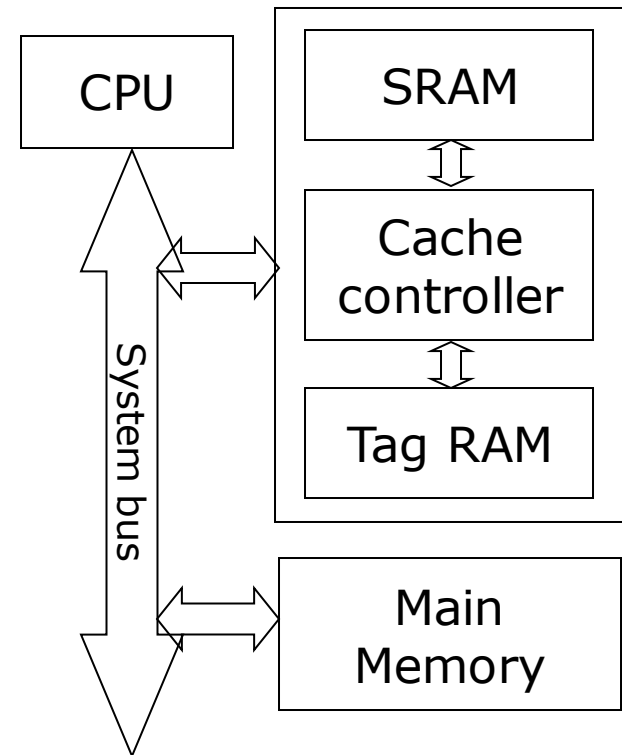
- Thiết kế đơn giản
- Miss nhanh

- Nhược:

- Hit chậm

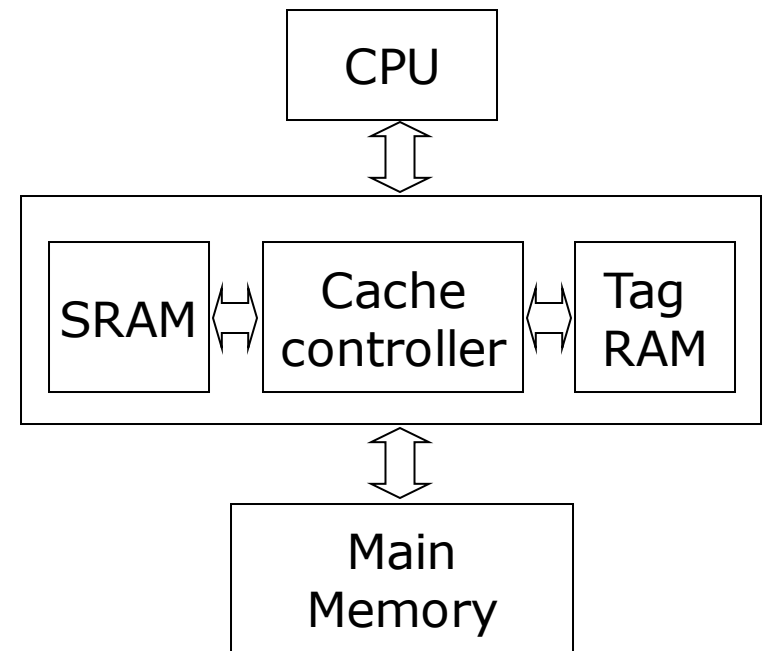
SRAM: RAM lưu dữ liệu cache

Tag RAM: RAM lưu địa chỉ bộ nhớ



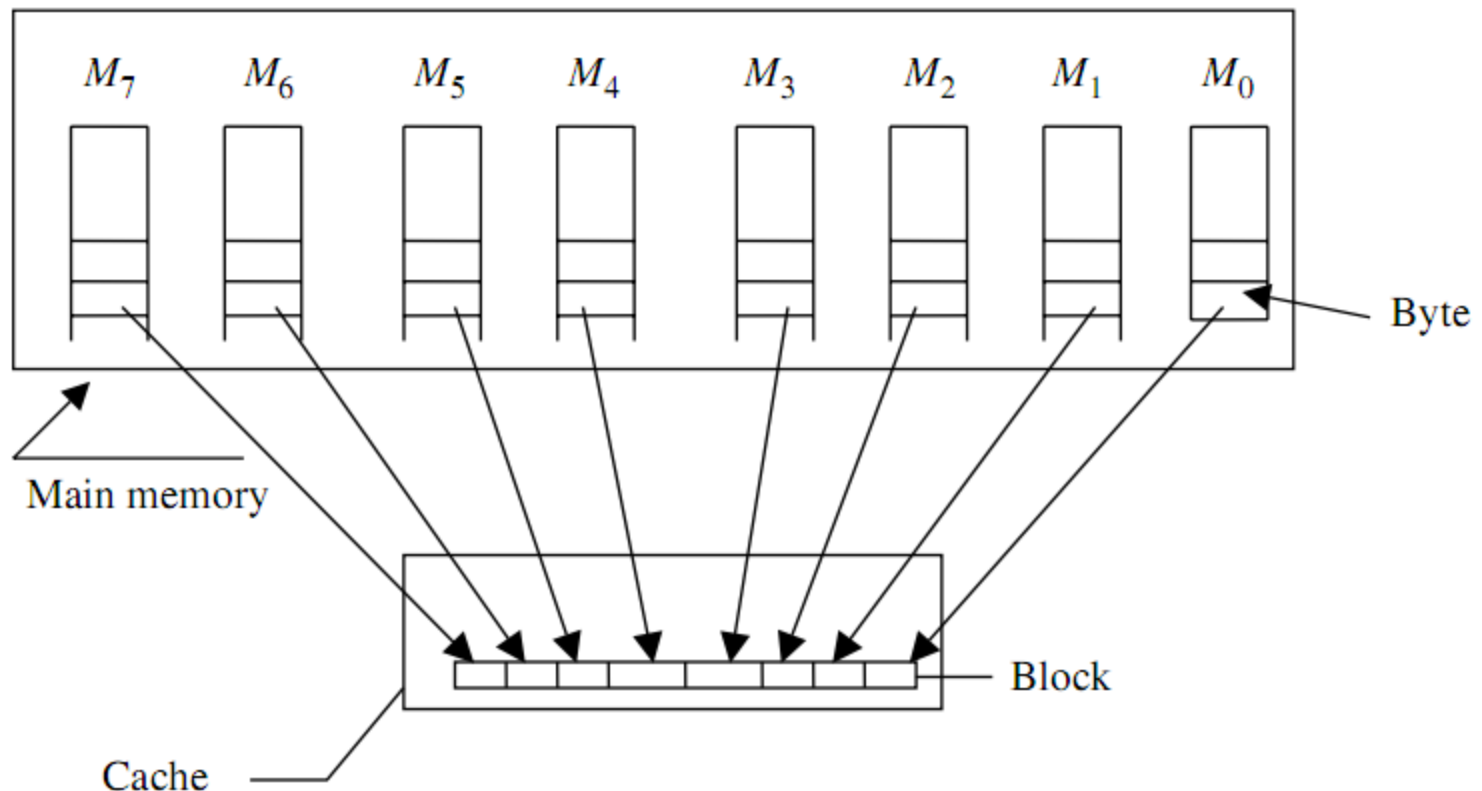
Kiến trúc Cache – look through

- Cache nằm giữa CPU và bộ nhớ chính
- Cache “thấy” chu kỳ bus CPU trước sau đó nó “truyền” lại cho bộ nhớ chính
- Ưu:
 - Hit nhanh
- Nhược:
 - Thiết kế phức tạp
 - Đắt
 - Miss chậm



Tổ chức Cache

- Tổ chức cache giải quyết vấn đề cache và bộ nhớ chính phối hợp làm việc với nhau như thế nào



Các kỹ thuật tổ chức Cache

- Ánh xạ trực tiếp (direct mapping):
 - Đơn giản, nhanh
 - Ánh xạ cố định
- Ánh xạ kết hợp đầy đủ (fully associative mapping):
 - Phức tạp, chậm
 - Ánh xạ linh hoạt
- Ánh xạ tập kết hợp/ theo bộ (set): (set associative mapping)
 - Phức tạp
 - Nhanh, ánh xạ linh hoạt

Ánh xạ trực tiếp

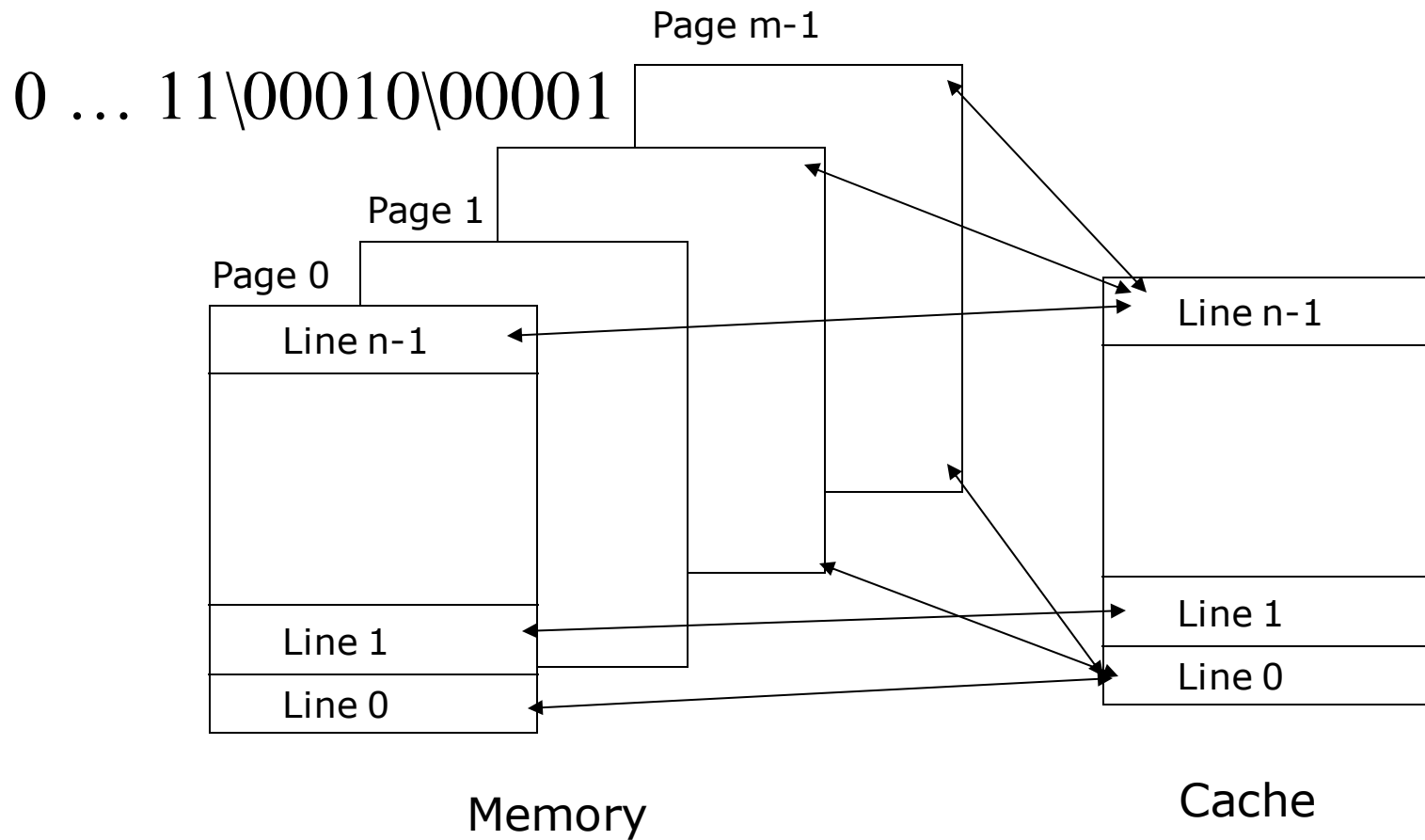
- Cache: Được chia thành n khối hoặc dòng (block or line), từ Line_0 tới Line_{n-1}
- Bộ nhớ:
 - Được chia thành m trang (page), từ page_0 tới page_{m-1}
 - Mỗi trang bộ nhớ có kích thước bằng cache
 - Mỗi trang có n lines, từ Line_0 tới Line_{n-1}
- Ánh xạ:
 - Line_0 của (page_0 tới page_{m-1}) được ánh xạ tới Line_0 của cache
 - Line_1 của (page_0 tới page_{m-1}) được ánh xạ tới Line_1 của cache
 -
 - Line_{n-1} của (page_0 tới page_{m-1}) được ánh xạ tới Line_{n-1} của cache

Địa chỉ ánh xạ trực tiếp

Tag	Line	Word
-----	------	------

- *Tag* (bit): là địa chỉ của trang trong bộ nhớ
- *Line* (bit): là địa chỉ của line trong cache
- *Word* (bit): là địa chỉ của word trong line

Ảnh xạ trực tiếp

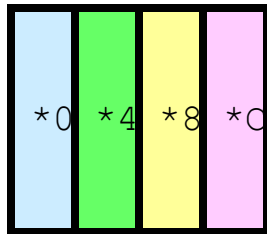


Địa chỉ ánh xạ trực tiếp

- Ví dụ: Tìm kích thước trường địa chỉ (32 bit) dùng direct mapping
 - Đầu vào:
 - Kích thước bộ nhớ: 4GB
 - Kích thước cache: 1KB
 - Kích thước line: 32 byte
 - Đầu ra:
 - Kích thước line: 32 byte = 2^5 -> Word = 5 bit
 - Kích thước cache: 1 KB = 2^{10} -> có $2^{10} / 2^5 = 2^5$ lines -> Line = 5 bit
 - Tag = địa chỉ 32 bit – Line – Word = $32 - 5 - 5 = 22$ bit

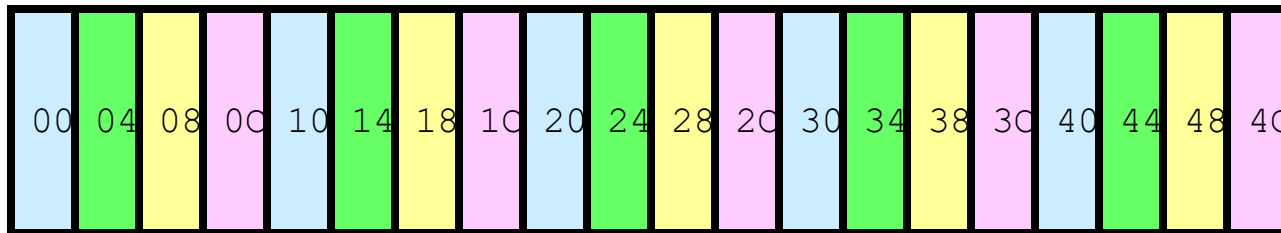
Ví dụ: Direct Mapped block

Cache



address maps to block:

location = (block address MOD # blocks in cache)



Memory

Accessing A Direct-Mapped Cache

- DM cache contains 4 1-word blocks. Find the # Misses for each cache given this sequence of memory block accesses: 0, 8, 0, 6, 8

DM Memory Access 1: Mapping: $0 \bmod 4 = 0$

Mem Block	DM Hit/Miss
0	

Block 0

Block 1

Block 2

Block 3

Accessing A Direct-Mapped Cache

DM cache contains 4 1-word blocks. Find the # Misses for each cache given this sequence of memory block accesses: 0, 8, 0, 6, 8

DM Memory Access 1: Mapping: $0 \bmod 4 = 0$

Mem Block	DM Hit/Miss	Block 0	Mem[0]
0	miss	Block 1	
		Block 2	
		Block 3	

Set 0 is empty: write Mem[0]

Accessing A Direct-Mapped Cache

- DM cache contains 4 1-word blocks. Find the # Misses for each cache given this sequence of memory block accesses: 0, 8, 0, 6, 8

DM Memory Access 2: Mapping: $8 \bmod 4 = 0$

Mem Block	DM Hit/Miss	Block 0
0	miss	Block 1
8		Block 2
		Block 3

Mem[0]

Accessing A Direct-Mapped Cache

- DM cache contains 4 1-word blocks. Find the # Misses for each cache given this sequence of memory block accesses: 0, 8, 0, 6, 8

DM Memory Access 2: Mapping: $8 \bmod 4 = 0$

Mem Block	DM Hit/Miss	Block 0
0	miss	
8	miss	Block 1
		Block 2
		Block 3

Mem[8]

Set 0 contains Mem[0]. Overwrite with Mem[8]

Accessing A Direct-Mapped Cache

- DM cache contains 4 1-word blocks. Find the # Misses for each cache given this sequence of memory block accesses: 0, 8, 0, 6, 8

DM Memory Access 3: Mapping: $0 \bmod 4 = 0$

Mem Block	DM Hit/Miss
0	miss
8	miss
0	

Block 0

Block 1

Block 2

Block 3

Mem[8]

Accessing A Direct-Mapped Cache

- DM cache contains 4 1-word blocks. Find the # Misses for each cache given this sequence of memory block accesses: 0, 8, 0, 6, 8

DM Memory Access 3: Mapping: $0 \bmod 4 = 0$

Mem Block	DM Hit/Miss	Block 0	<div>Mem[0]</div>
0	miss		
8	miss	Block 1	
0	miss	Block 2	
		Block 3	

Set 0 contains Mem[8]. Overwrite with Mem[0]

Accessing A Direct-Mapped Cache

- DM cache contains 4 1-word blocks. Find the # Misses for each cache given this sequence of memory block accesses: 0, 8, 0, 6, 8

DM Memory Access 4: Mapping: $6 \bmod 4 = 2$

Mem Block	DM Hit/Miss	Block 0
0	miss	Block 1
8	miss	Block 2
0	miss	Block 3
6		

Mem[0]

Accessing A Direct-Mapped Cache

- DM cache contains 4 1-word blocks. Find the # Misses for each cache given this sequence of memory block accesses: 0, 8, 0, 6, 8

DM Memory Access 4: Mapping: $6 \bmod 4 = 2$

Mem Block	DM Hit/Miss
0	miss
8	miss
0	miss
6	miss

Block 0

Block 1

Block 2

Block 3

Mem[0]
Mem[6]

Set 2 empty. Write Mem[6]

Accessing A Direct-Mapped Cache

- DM cache contains 4 1-word blocks. Find the # Misses for each cache given this sequence of memory block accesses: 0, 8, 0, 6, 8

DM Memory Access 5: Mapping: $8 \bmod 4 = 0$

Mem Block	DM Hit/Miss
0	miss
8	miss
0	miss
6	miss
8	

Block 0

Block 1

Block 2

Block 3

Mem[0]
Mem[6]

Accessing A Direct-Mapped Cache

DM cache contains 4 1-word blocks. Find the # Misses for each cache given this sequence of memory block accesses: 0, 8, 0, 6, 8

DM Memory Access 5: Mapping: $8 \bmod 4 = 0$

Mem Block	DM Hit/Miss	Block 0
0	miss	
8	miss	Block 1
0	miss	Block 2
6	miss	
8	miss	Block 3

Mem[8]
Mem[6]

Set 0 contains Mem[0]. Overwrite with Mem[8]

Ảnh xạ trực tiếp

□ Ưu điểm:

- Thiết kế đơn giản
- Nhanh vì ánh xạ cố định: khi biết địa chỉ bộ nhớ có thể tìm nó trong cache rất nhanh

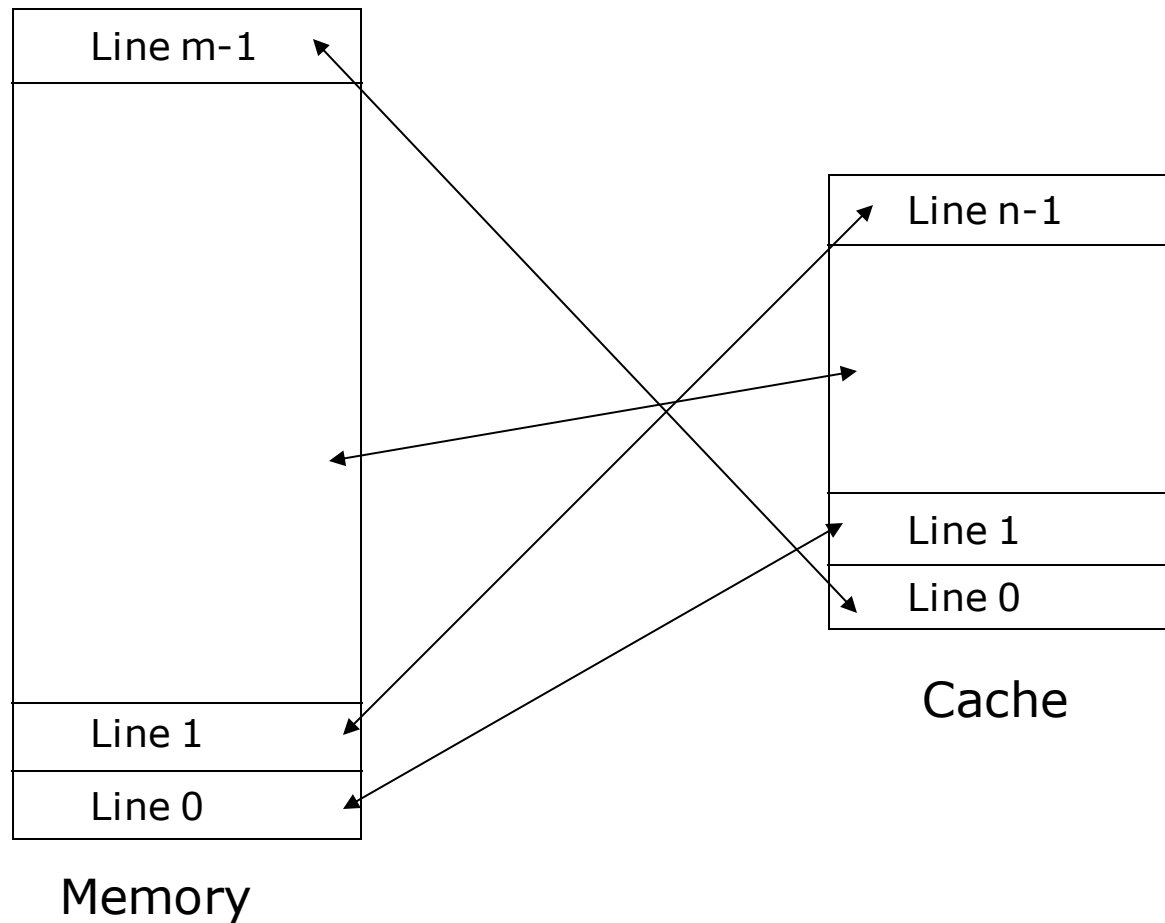
□ Nhược điểm:

- Vì ánh xạ cố định nên khả năng xảy ra xung đột cao
- Tỷ lệ hit thấp

Ánh xạ kết hợp (liên kết) đầy đủ

- Cache: Được chia thành n khối hoặc dòng (block or line), từ Line_0 tới Line_{n-1}
- Bộ nhớ:
 - Được chia thành m khối hay dòng, từ Line_0 tới Line_{m-1}
 - Kích thước mỗi dòng cache bằng kích thước một dòng bộ nhớ
 - Số lượng dòng trong bộ nhớ có thể lớn hơn nhiều số lượng dòng của cache ($m \gg n$)
- Ánh xạ:
 - Một dòng của bộ nhớ có thể ánh xạ tới dòng bất kì của cache
 - Line_i của bộ nhớ có thể ánh xạ tới line_j của cache

Ảnh xạ kết hợp đầy đủ



Địa chỉ ánh xạ kết hợp đầy đủ

Tag	Word
-----	------

- *Tag* (bit) là địa chỉ của line trong bộ nhớ (page =1)
- *Word* (bit) là địa chỉ của từ trong line

Địa chỉ ánh xạ kết hợp đầy đủ

□ Ví dụ: Tìm kích thước các trường địa chỉ (32 bit)

■ Input:

- Kích thước cache = 1KB
- Kích thước line = 32 byte

■ Output:

- Kích thước Line = 32 byte = 2^5 -> Word = 5 bit
- Tag = 32 bit – word = $32 - 5 = 27$ bit

Ánh xạ kết hợp đầy đủ

□ Ưu:

- Ít xung đột vì ánh xạ linh hoạt
- Tỷ lệ hit cao hơn

□ Nhược:

- Chậm vì phải tìm kiếm địa chỉ bộ nhớ trong cache
- Phức tạp vì có thêm n bộ so sánh địa chỉ trong cache

□ Thường sử dụng cho cache có kích thước nhỏ

Example: Accessing A Fully-Associative Cache

Fully-Associative cache contains 4 1-word blocks. Find the # Misses for each cache given this sequence of memory block accesses: 0, 8, 0, 6, 8

FA Memory Access 1:

Mem Block	DM Hit/Miss
0	

S
e
t
0

--	--	--	--

FA Block Replacement Rule: replace least recently used block in set

Example: Accessing A Fully-Associative Cache

Fully-Associative cache contains 4 1-word blocks. Find the # Misses for each cache given this sequence of memory block accesses: 0, 8, 0, 6, 8

FA Memory Access 1:

Mem Block	DM Hit/Miss
0	miss

S
e
t
0

Mem [0]			
------------	--	--	--

Set 0 is empty: write Mem[0] to Block 0

Example: Accessing A Fully-Associative Cache

Fully-Associative cache contains 4 1-word blocks. Find the # Misses for each cache given this sequence of memory block accesses: 0, 8, 0, 6, 8

FA Memory Access 2:

Mem Block	DM Hit/Miss
0	miss
8	

S
e
t
0

Mem [0]			
------------	--	--	--

Example: Accessing A Fully-Associative Cache

Fully-Associative cache contains 4 1-word blocks. Find the # Misses for each cache given this sequence of memory block accesses: 0, 8, 0, 6, 8

FA Memory Access 2:

Mem Block	DM Hit/Miss
0	miss
8	miss

S
e
t
0

Mem [0]	Mem [8]		
------------	------------	--	--

Blocks 1-3 are LRU: write Mem[8] to Block 1

Example: Accessing A Fully-Associative Cache

Fully-Associative cache contains 4 1-word blocks. Find the # Misses for each cache given this sequence of memory block accesses: 0, 8, 0, 6, 8

FA Memory Access 3:

Mem Block	DM Hit/Miss
0	miss
8	miss
0	

S
e
t
0

Mem [0]	Mem [8]		

Example: Accessing A Fully-Associative Cache

Fully-Associative cache contains 4 1-word blocks. Find the # Misses for each cache given this sequence of memory block accesses: 0, 8, 0, 6, 8

FA Memory Access 3:

Mem Block	DM Hit/Miss
0	miss
8	miss
0	hit

S
e
t
0

Mem [0]	Mem [8]		
------------	------------	--	--

Block 0 contains Mem[0]

Example: Accessing A Fully-Associative Cache

Fully-Associative cache contains 4 1-word blocks. Find the # Misses for each cache given this sequence of memory block accesses: 0, 8, 0, 6, 8

FA Memory Access 4:

Mem Block	DM Hit/Miss
0	miss
8	miss
0	hit
6	

S
e
t
0

Mem [0]	Mem [8]		

Example: Accessing A Fully-Associative Cache

Fully-Associative cache contains 4 1-word blocks. Find the # Misses for each cache given this sequence of memory block accesses: 0, 8, 0, 6, 8

FA Memory Access 4:

Mem Block	DM Hit/Miss
0	miss
8	miss
0	hit
6	miss

S
e
t
0

Mem [0]	Mem [8]	Mem [6]	

Blocks 2-3 are LRU : write Mem[6] to Block 2

Example: Accessing A Fully-Associative Cache

Fully-Associative cache contains 4 1-word blocks. Find the # Misses for each cache given this sequence of memory block accesses: 0, 8, 0, 6, 8

FA Memory Access 5:

Mem Block	DM Hit/Miss
0	miss
8	miss
0	hit
6	miss
8	

S
e
t
0

Mem [0]	Mem [8]	Mem [6]	

Example: Accessing A Fully-Associative Cache

Fully-Associative cache contains 4 1-word blocks. Find the # Misses for each cache given this sequence of memory block accesses: 0, 8, 0, 6, 8

FA Memory Access 5:

Mem Block	DM Hit/Miss
0	miss
8	miss
0	hit
6	miss
8	hit

Set 0

Mem [0]	Mem [8]	Mem [6]	
---------	---------	---------	--

Block 1 contains Mem[8]

Ánh xạ tập kết hợp (liên kết theo nhóm)

□ Cache:

- Được chia thành k đường (ways) có kích thước bằng nhau
- Mỗi đường được chia thành n dòng (block or line), từ Line_0 tới Line_{n-1}

□ Bộ nhớ:

- Được chia thành m trang, từ page_0 tới page_{m-1}
- Kích thước trang page bằng kích thước way của cache
- Mỗi trang có n line, từ Line_0 tới Line_{n-1}

Ánh xạ tập kết hợp

□ Ánh xạ:

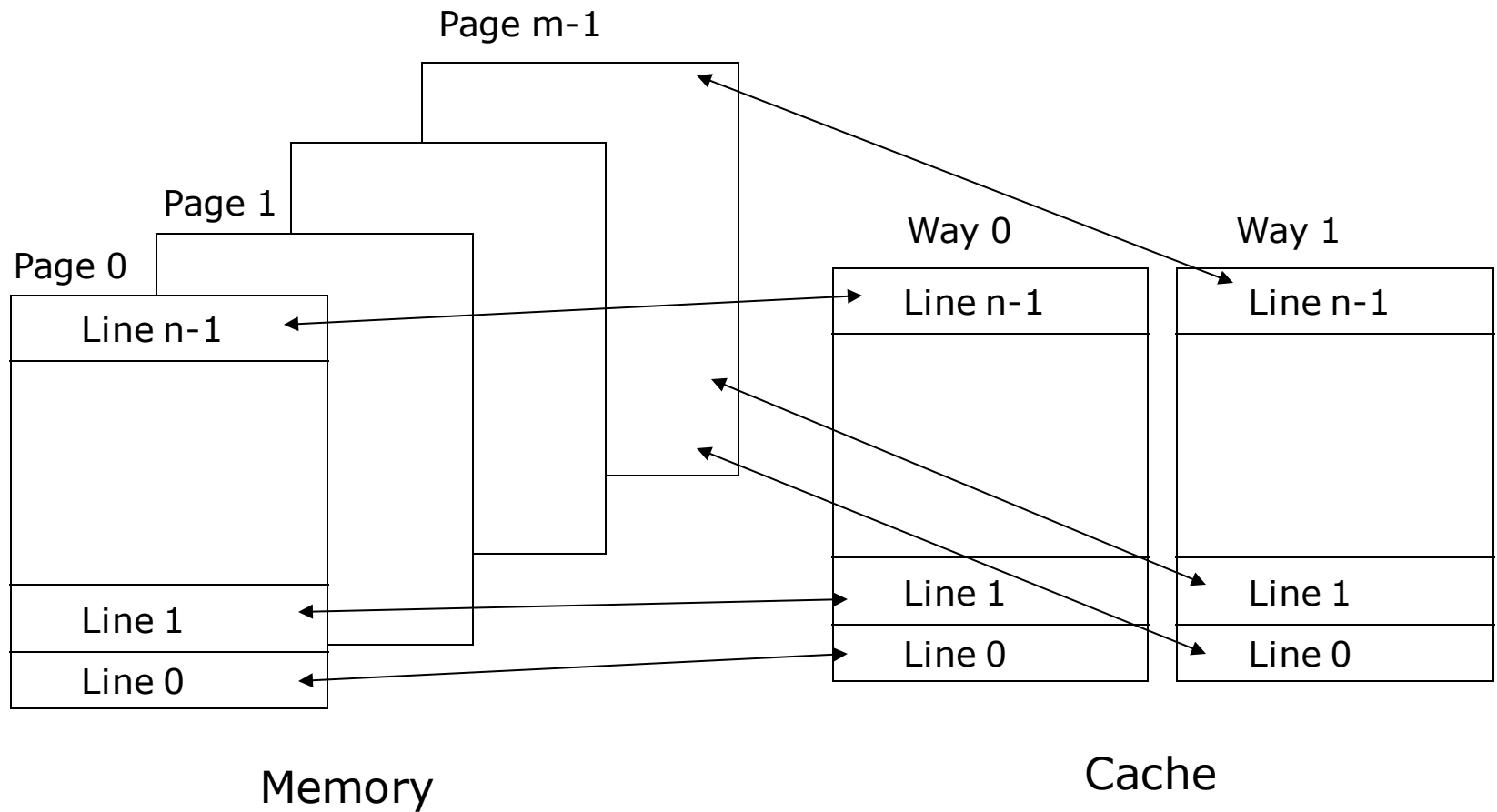
■ Page được ánh xạ tới Way (ánh xạ linh hoạt):

- Một page bộ nhớ có thể được ánh xạ tới way bất kì của cache

■ Line của page ánh xạ tới line của way (ánh xạ cố định):

- Line_0 của page_i được ánh xạ tới Line_0 của way_j ;
- Line_1 của page_i được ánh xạ tới Line_1 của way_j ;
- ...
- Line_{n-1} của page_i được ánh xạ tới Line_{n-1} của way_j ;

Ảnh xạ tập kết hợp



Địa chỉ ánh xạ tập kết hợp

Tag	Set	Word
-----	-----	------

- *Tag* (bit): là địa chỉ của trang trong bộ nhớ
- *Set* (bit): là địa chỉ của line trong way của cache
- *Word* (bit): là địa chỉ của word trong line

Địa chỉ ánh xạ tập kết hợp

□ Ví dụ:

■ Input:

- Kích thước bộ nhớ: 4GB
- Kích thước cache: 1KB, 2 ways
- Kích thước Line: 32 byte

■ Output:

- Line size = 32 byte = $2^5 \rightarrow$ Word = 5 bit
- Cache size = 1KB = $2^{10} \rightarrow$ có $2^{10} / 2^5 / 2 = 2^4$ lines trong 1 way \rightarrow Set = 4 bit
- Tag = 32bit địa chỉ – Set – Word = $32 - 4 - 5 = 23$ bit.

Ánh xạ tập kết hợp

□ Ưu:

- Nhanh vì ánh xạ trực tiếp được sử dụng cho ánh xạ dòng
- Ít xung đột vì ánh xạ từ trang nhớ tới đường của cache là linh hoạt
- Tỷ lệ tìm thấy (hit) cao

□ Nhược:

- Thiết kế và điều khiển phức tạp vì cache được chia thành các way

Accessing A Set-Associative Cache

2-way Set-Associative cache contains 2 sets, 2 one-word blocks each. Find the # Misses for each cache given this sequence of memory block accesses: 0, 8, 0, 6, 8

SA Memory Access 1: Mapping: $0 \bmod 2 = 0$

Mem Block	DM Hit/Miss
0	

Set 0

Set 1

SA Block Replacement Rule: replace least recently used block in set

Accessing A Set-Associative Cache

2-way Set-Associative cache contains 2 sets, 2 one-word blocks each. Find the # Misses for each cache given this sequence of memory block accesses: 0, 8, 0, 6, 8

SA Memory Access 1: Mapping: $0 \bmod 2 = 0$

Mem Block	DM Hit/Miss
0	miss

Set 0

Set 1

Mem[0]	

Set 0 is empty: write Mem[0] to Block 0

Accessing A Set-Associative Cache

2-way Set-Associative cache contains 2 sets, 2 one-word blocks each. Find the # Misses for each cache given this sequence of memory block accesses: 0, 8, 0, 6, 8

SA Memory Access 2: Mapping: $8 \bmod 2 = 0$

Mem Block	DM Hit/Miss
0	miss
8	

Set 0

Set 1

Mem[0]	

Accessing A Set-Associative Cache

2-way Set-Associative cache contains 2 sets, 2 one-word blocks each. Find the # Misses for each cache given this sequence of memory block accesses: 0, 8, 0, 6, 8

SA Memory Access 2: Mapping: $8 \bmod 2 = 0$

Mem Block	DM Hit/Miss
0	miss
8	miss

Set 0

Set 1

Mem[0]	Mem[8]

Set 0, Block 1 is LRU: write Mem[8]

Accessing A Set-Associative Cache

2-way Set-Associative cache contains 2 sets, 2 one-word blocks each. Find the # Misses for each cache given this sequence of memory block accesses: 0, 8, 0, 6, 8

SA Memory Access 3: Mapping: $0 \bmod 2 = 0$

Mem Block	DM Hit/Miss
0	miss
8	miss
0	

Set 0

Set 1

Mem[0]	Mem[8]

Accessing A Set-Associative Cache

2-way Set-Associative cache contains 2 sets, 2 one-word blocks each. Find the # Misses for each cache given this sequence of memory block accesses: 0, 8, 0, 6, 8

SA Memory Access 3: Mapping: $0 \bmod 2 = 0$

Mem Block	DM Hit/Miss
0	miss
8	miss
0	hit

Set 0

Set 1

Mem[0]	Mem[8]

Set 0, Block 0 contains Mem[0]

Accessing A Set-Associative Cache

2-way Set-Associative cache contains 2 sets, 2 one-word blocks each. Find the # Misses for each cache given this sequence of memory block accesses: 0, 8, 0, 6, 8

SA Memory Access 4: Mapping: $6 \bmod 2 = 0$

Mem Block	DM Hit/Miss
0	miss
8	miss
0	hit
6	

Set 0

Set 1

Mem[0]	Mem[8]

Accessing A Set-Associative Cache

2-way Set-Associative cache contains 2 sets, 2 one-word blocks each. Find the # Misses for each cache given this sequence of memory block accesses: 0, 8, 0, 6, 8

SA Memory Access 4: Mapping: $6 \bmod 2 = 0$

Mem Block	DM Hit/Miss
0	miss
8	miss
0	hit
6	miss

Set 0

Set 1

Mem[0]	Mem[6]

Set 0, Block 1 is LRU: overwrite with Mem[6]

Accessing A Set-Associative Cache

2-way Set-Associative cache contains 2 sets, 2 one-word blocks each. Find the # Misses for each cache given this sequence of memory block accesses: 0, 8, 0, 6, 8

SA Memory Access 5: Mapping: $8 \bmod 2 = 0$

Mem Block	DM Hit/Miss
0	miss
8	miss
0	hit
6	miss
8	

Set 0

Set 1

Mem[0]	Mem[6]

Accessing A Set-Associative Cache

2-way Set-Associative cache contains 2 sets, 2 one-word blocks each. Find the # Misses for each cache given this sequence of memory block accesses: 0, 8, 0, 6, 8

SA Memory Access 5: Mapping: $8 \bmod 2 = 0$

Mem Block	DM Hit/Miss
0	miss
8	miss
0	hit
6	miss
8	miss

Set 0

Set 1

Mem[8]	Mem[6]

Set 0, Block 0 is LRU: overwrite with Mem[8]

Đọc/ ghi thông tin trong cache

□ Thao tác đọc:

- Trường hợp tìm thấy (hit case): mục dữ liệu yêu cầu tìm thấy trong cache
 - Mục dữ liệu được đọc từ cache vào CPU
 - Bộ nhớ chính không tham gia
- Trường hợp không tìm thấy (miss case):
 - Mục dữ liệu được đọc từ bộ nhớ vào cache
 - Sau đó dữ liệu được chuyển từ cache vào CPU
 - ⇒ Miss penalty: thời gian truy cập mục dữ liệu bằng tổng thời gian truy cập cache và bộ nhớ chính

Ghi thông tin trong cache

- Trường hợp hit (bản tin thuộc 1 khối trong cache)
 - Write through (ghi thẳng): mục dữ liệu được ghi vào cache và ghi vào bộ nhớ đồng thời
 - Write back (ghi trễ): mục dữ liệu trước tiên được ghi vào cache và cả dòng (block) chứa nó ở trong cache sẽ được ghi lại vào bộ nhớ sau đó, khi mà dòng đó bị thay thế
- Trường hợp miss (bản tin ko có trong cache)
 - Write allocate (ghi có đọc lại): mục dữ liệu trước hết được ghi vào bộ nhớ sau đó cả dòng chứa nó sẽ được đọc vào cache
 - Write non-allocate (ghi không đọc lại): mục dữ liệu chỉ được ghi vào bộ nhớ

Các chính sách thay thế dòng cache

- Tại sao phải thay thế dòng cache?
 - Ánh xạ dòng (bộ nhớ) -> dòng (cache) thường là ánh xạ nhiều -> một
 - Nhiều dòng bộ nhớ chia sẻ một dòng cache -> các dòng bộ nhớ được nạp vào cache sử dụng một thời gian và được thay thế bởi dòng khác theo yêu cầu thông tin phục vụ CPU
- Chính sách thay thế (replacement policies): xác định cách thức lựa chọn các dòng trong cache để thay thế khi có dòng mới từ bộ nhớ cần chuyển vào
- 3 chiến lược chính:
 - Thay thế ngẫu nhiên (random replacement)
 - Vào trước ra trước FIFO (First In First Out)
 - Thay thế các dòng ít được sử dụng gần đây nhất LRU (Least Recently Used)

Các chính sách thay thế cache

□ Thay thế ngẫu nhiên:

- Các dòng trong cache được chọn ngẫu nhiên để thay
- Đơn giản
- Tỷ lệ miss cao vì phương pháp này không xét tới dòng cache nào đang thực sự được sử dụng
 - Nếu một dòng cache đang được sử dụng mà bị thay thế thì sự kiện miss xảy ra và cần phải đọc lại vào cache

Các chính sách thay thế cache

- Thay thế kiểu vào trước ra trước FIFO
 - Dựa trên nguyên lý FIFO
 - Các dòng cache được đọc vào cache trước sẽ được chọn để thay trước
 - Tỷ lệ miss thấp hơn so với phương pháp ngẫu nhiên
 - Tỷ lệ miss vẫn cao vì phương pháp này vẫn chưa thực sự xem xét tới block nào đang thực sự được sử dụng
 - Một dòng cache “cũ” có thể vẫn đang được sử dụng
 - Cài đặt phức tạp vì cần thêm mạch để giám sát thứ tự nạp các dòng bộ nhớ vào cache

Các chính sách thay thế cache

- Thay thế các dòng ít được sử dụng gần đây nhất LRU
 - Các dòng cache ít được sử dụng nhất trong thời gian gần đây sẽ được chọn để thay
 - Xét tới các dòng đang được sử dụng
 - Tỷ lệ miss thấp nhất so với phương pháp ngẫu nhiên và FIFO
 - Cài đặt phức tạp vì cần thêm mạch để giám sát tần suất sử dụng các dòng cache

Hiệu năng cache

- Thời gian truy cập trung bình của hệ thống nhớ có cache:

$$\begin{aligned} T_{\text{access}} &= (\text{hit cost}) + (\text{miss cost}) \\ &= (\text{Hit cost}) + (\text{miss rate}) * (\text{miss penalty}) \\ &= t_{\text{cache}} + (1 - H) * (t_{\text{memory}}) \end{aligned}$$

Trong đó H là hệ số hit

Nếu $t_{\text{cache}} = 5\text{ns}$, $t_{\text{memory}} = 60\text{ns}$ và $H=80\%$, ta có:

$$t_{\text{access}} = 5 + (1 - 0.8) * (60) = 5 + 12 = 17\text{ns}$$

Nếu $t_{\text{cache}} = 5\text{ns}$, $t_{\text{memory}} = 60\text{ns}$ và $H=95\%$, ta có:

$$t_{\text{access}} = 5 + (1 - 0.95) * (60) = 5 + 3 = 8\text{ns}$$

Các yếu tố ảnh hưởng – kích thước cache

□ Kích thước cache lớn:

- Có thể lưu trữ nhiều khối dữ liệu trong bộ nhớ hơn
- Giảm tần suất trao đổi các khối dữ liệu của chương trình khác nhau giữa bộ nhớ và cache
- Cache lớn chậm hơn cache nhỏ:
 - Tìm kiếm vị trí bộ nhớ trong không gian lớn hơn
- Xu hướng: cache càng ngày càng lớn
 - Hỗ trợ đa nhiệm (multi-tasking) tốt hơn
 - Hỗ trợ tốt hơn cho xử lý song song
 - Hỗ trợ tốt hơn cho các hệ thống CPU đa nhân

Các yếu tố ảnh hưởng - Cache nhiều mức

- Nâng cao hiệu năng hệ thống vì cache nhiều mức có thể dung hòa tốc độ của CPU và MEM tốt hơn cache một mức

CPU	L1	L2	L3	Main memory
1ns	5ns	15ns	30ns	60ns
1ns	5ns			60ns

- Thực tế, cache thường có 2 mức: L1 và L2. Một số cache có 3 mức: L1, L2 và L3
- Giảm giá thành hệ thống nhớ

Các yếu tố ảnh hưởng – phân chia cache

- Cache có thể được chia thành cache lệnh và cache dữ liệu để hiệu năng tốt hơn vì:
 - Dữ liệu và lệnh khác nhau về tính cục bộ
 - Dữ liệu có tính chất cục bộ về thời gian hơn
 - Lệnh có tính chất cục bộ về không gian hơn
 - Cache lệnh chỉ hỗ trợ thao tác đọc còn cache dữ liệu hỗ trợ cả 2 thao tác đọc ghi
- ⇒ Dễ tối ưu cache hơn
- Hỗ trợ nhiều thao tác đọc/ ghi cùng lúc => giảm xung đột tài nguyên
- Kết hợp các chức năng khác như giải mã trước lệnh vào trong cache lệnh => xử lý lệnh tốt hơn

Các yếu tố ảnh hưởng – phân chia cache

- Trong thực tế, hầu hết cache L1 được chia thành 2 phần:
 - I_cache (Instruction cache): cache lệnh
 - D_cache (data cache): cache dữ liệu.
- Các cache mức cao hơn không được chia
 - Chia cache L1 có lợi ích cao nhất vì nó gần CPU nhất. CPU đọc/ ghi trực tiếp lên cache L1. Cache L1 cần hỗ trợ nhiều lệnh truy nhập đồng thời và các biện pháp tối ưu hóa
 - Chia các cache mức cao hơn không đem lại hiệu quả cao và làm phức tạp trong hệ thống điều khiển cache

Các biện pháp giảm miss

□ Cache tốt:

- Hệ số hit cao
- Hệ số miss thấp
- Miss penalty không quá chậm

□ Các kiểu miss:

- Compulsory misses (không tìm thấy bắt buộc): thường xảy ra tại thời điểm chương trình được kích hoạt, khi mã chương trình đang được tải vào bộ nhớ và chưa được nạp vào cache
- Capacity misses (không tìm thấy do dung lượng): do kích thước của cache hạn chế, đặc biệt trong môi trường đa nhiệm. Khi cache nhỏ, mã chương trình thường xuyên bị chuyển đổi giữa cache và bộ nhớ
- Conflict misses (không tìm thấy do xung đột): do có xung đột khi có nhiều dòng nhớ cùng cạnh tranh 1 dòng cache

Các biện pháp giảm miss

□ Tăng kích thước dòng cache

- Giảm compulsory misses: dòng có kích thước lớn sẽ có khả năng bao phủ lân cận tốt hơn
- Tăng conflict miss:
 - Kích thước dòng lớn => giảm số lượng dòng trong cache => có nhiều dòng bộ nhớ cùng tham chiếu tới dòng cache hơn => tăng conflict miss
- Kích thước dòng lớn có thể gây lãng phí dung lượng của cache (một số phần của dòng có thể không bao giờ được sử dụng)
- Kích thước dòng thường dùng là 64 bytes

Các phương pháp giảm miss

- Tăng mức độ liên kết (tăng số lượng cache way):
 - Giảm conflict miss:
 - Tăng số lượng ways \Rightarrow ánh xạ bộ nhớ - cache linh hoạt hơn hay nhiều lựa chọn hơn \Rightarrow giảm conflict miss
 - Làm cache chậm hơn:
 - Tăng số lượng ways \Rightarrow không gian tìm kiếm lớn hơn \Rightarrow làm cache chậm đi

Calculating Bits in Cache

- How many total bits are needed for a direct- mapped cache with 64 KBytes of data and one word blocks, assuming a 32-bit address?
- How many total bits would be needed for a 4-way set associative cache to store the same amount of data
- How many total bits are needed for a direct- mapped cache with 64 KBytes of data and 8 word blocks, assuming a 32-bit address?

Calculating Bits in Cache

- How many total bits are needed for a direct-mapped cache with 64 KBytes of data and one word blocks, assuming a 32-bit address?
- $64 \text{ Kbytes} = 16 \text{ K words} = 2^{14} \text{ words} = 2^{14} \text{ blocks}$
 - $\text{block size} = 4 \text{ bytes} \Rightarrow \text{offset size} = 2 \text{ bits},$
 - $\# \text{sets} = \# \text{blocks} = 2^{14} \Rightarrow \text{index size} = 14 \text{ bits}$
 - $\text{tag size} = \text{address size} - \text{index size} - \text{offset size} = 32 - 14 - 2 = 16 \text{ bits}$
 - $\text{bits/block} = \text{data bits} + \text{tag bits} + \text{valid bit} = 32 + 16 + 1 = 49$
 - $\text{bits in cache} = \# \text{blocks} \times \text{bits/block} = 2^{14} \times 49 = 98 \text{ Kbytes}$

Calculating Bits in Cache

- How many total bits would be needed for a 4-way set associative cache to store the same amount of data
 - block size and #blocks does not change
 - $\#sets = \#blocks/4 = (2^{14})/4 = 2^{12} \Rightarrow$ index size = 12 bits
 - tag size = address size - index size - offset = $32 - 12 - 2 = 18$ bits
 - bits/block = data bits + tag bits + valid bit = $32 + 18 + 1 = 51$
 - bits in cache = $\#blocks \times bits/block = 2^{14} \times 51 = 102 \text{ Kbytes}$
- Increase associativity \Rightarrow increase bits in cache

Calculating Bits in Cache

- How many total bits are needed for a direct- mapped cache with 64 KBytes of data and 8 word blocks, assuming a 32-bit address?
 - $64 \text{ Kbytes} = 2^{14} \text{ words} = (2^{14})/8 = 2^{11} \text{ blocks}$
 - $\text{block size} = 32 \text{ bytes} \Rightarrow \text{offset size} = 5 \text{ bits},$
 - $\# \text{sets} = \# \text{blocks} = 2^{11} \Rightarrow \text{index size} = 11 \text{ bits}$
 - $\text{tag size} = \text{address size} - \text{index size} - \text{offset size} = 32 - 11 - 5 = 16 \text{ bits}$
 - $\text{bits/block} = \text{data bits} + \text{tag bits} + \text{valid bit} = 8 \times 32 + 16 + 1 = 273 \text{ bits}$
 - $\text{bits in cache} = \# \text{blocks} \times \text{bits/block} = 2^{11} \times 273 = 68.25 \text{ Kbytes}$

□ Increase block size \Rightarrow decrease bits in cache

Q1: Block Placement

- Where can block be placed in cache?
 - In one predetermined place - direct-mapped
 - Use part of address to calculate block location in cache
 - Compare cache block with tag to check if block present
 - Anywhere in cache - fully associative
 - Compare tag to every block in cache
 - In a limited set of places - set-associative
 - Use portion of address to calculate set (like direct-mapped)
 - Place in any block in the set
 - Compare tag to every block in set
 - Hybrid of direct mapped and fully associative

Q2: Block Replacement

- On a miss, data must be read from memory.
- So, where do we put the new data?
 - Direct-mapped cache: must place in fixed location
 - Set-associative, fully-associative - can pick within set

Replacement Algorithms

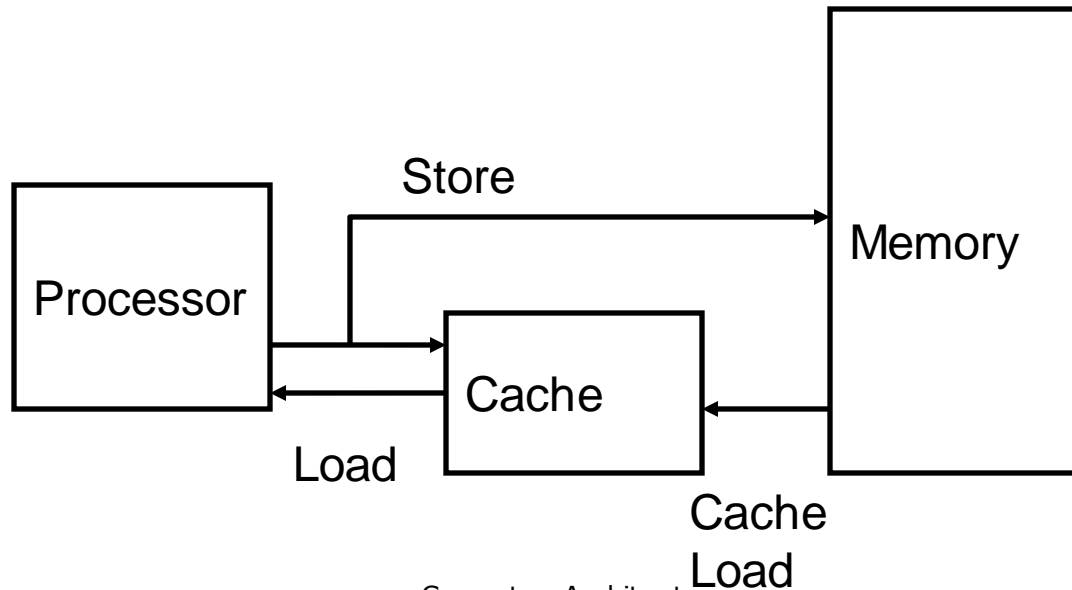
- *When a block is fetched, which block in the target set should be replaced?*
- Optimal algorithm:
 - replace the block that will not be used for the longest time (must know the future)
- Usage based algorithms:
 - Least recently used (LRU)
 - replace the block that has been referenced least recently
 - hard to implement
- Non-usage based algorithms:
 - First-in First-out (FIFO)
 - treat the set as a circular queue, replace head of queue.
 - easy to implement
 - Random (RAND)
 - replace a random block in the set
 - even easier to implement

Q4: Write Strategy

- What happens on a write?
 - **Write through** - write to memory, stall processor until done
 - **Write buffer** - place in buffer (allows pipeline to continue*)
 - **Write back** - delay write to memory until block is replaced in cache

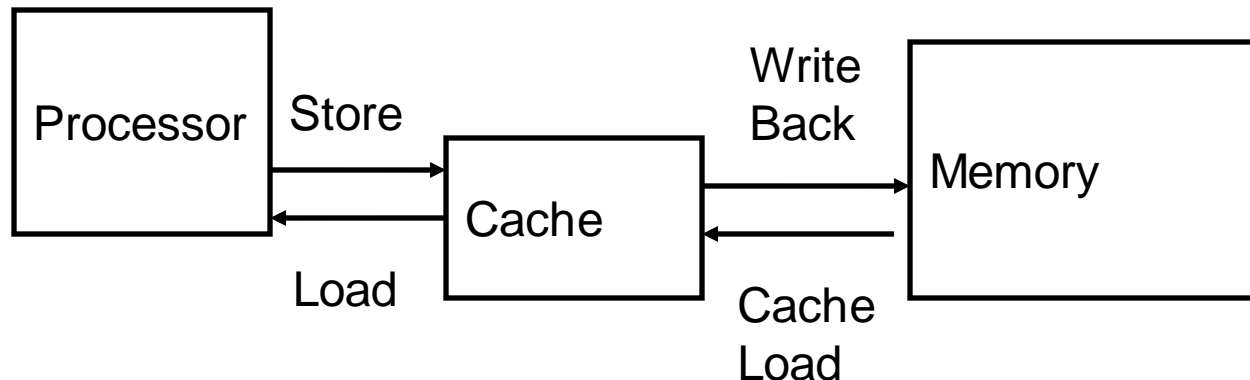
Write Through

- ❑ Store by processor updates cache *and* memory
- ❑ Memory always consistent with cache
- ❑ ~2X more loads than stores
- ❑ WT always combined with write buffers so that don't wait for lower level memory

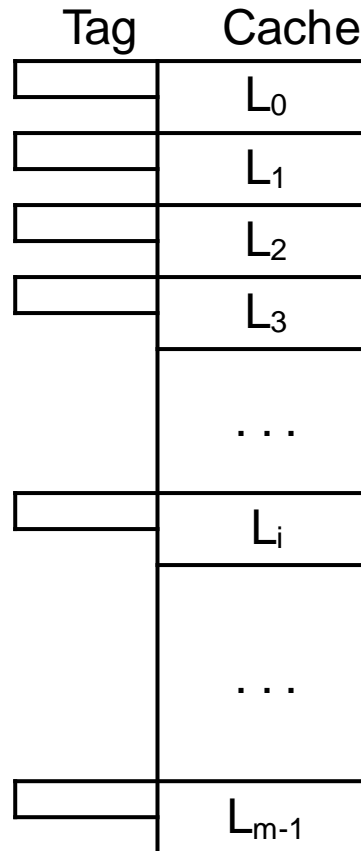
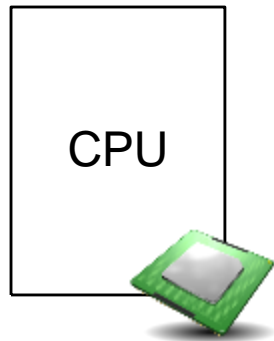


Write Back

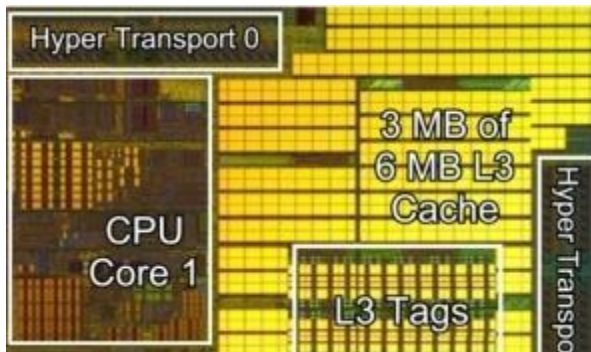
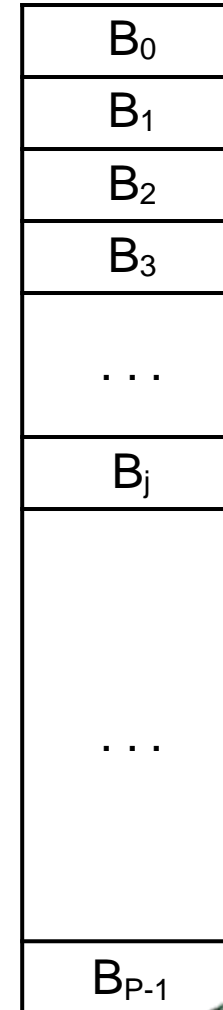
- Store by processor only updates cache line
- Modified line written to memory only when it is evicted
 - Requires “dirty bit” for each line
 - Set when line in cache is modified
 - Indicates that line in memory is not updated
- Memory not always consistent with cache
- No writes of repeated writes



Cấu trúc chung của cache



Bộ nhớ chính



Cấu trúc chung của cache

- Bộ nhớ chính có 2^N byte nhớ
- Bộ nhớ chính được chia thành các khối, gọi là **Block**, có kích thước bằng nhau.
 - Bộ nhớ chính: $\mathbf{B}_0, \mathbf{B}_1, \mathbf{B}_2, \dots, \mathbf{B}_{p-1}$ (p Blocks)
 - Kích thước của Block = 8,16,32,64,128 byte
- Bộ nhớ cache được chia thành các khối, gọi là **Line**, có kích thước bằng nhau.
 - Bộ nhớ cache: $\mathbf{L}_0, \mathbf{L}_1, \mathbf{L}_2, \dots, \mathbf{L}_{m-1}$ (m Lines)
 - Mỗi Line trong cache có một thẻ nhớ (Tag) được gắn vào
- Kích thước của **Line** (trong cache) và **Block** (trong bộ nhớ chính) bằng nhau

Cấu trúc chung của cache (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

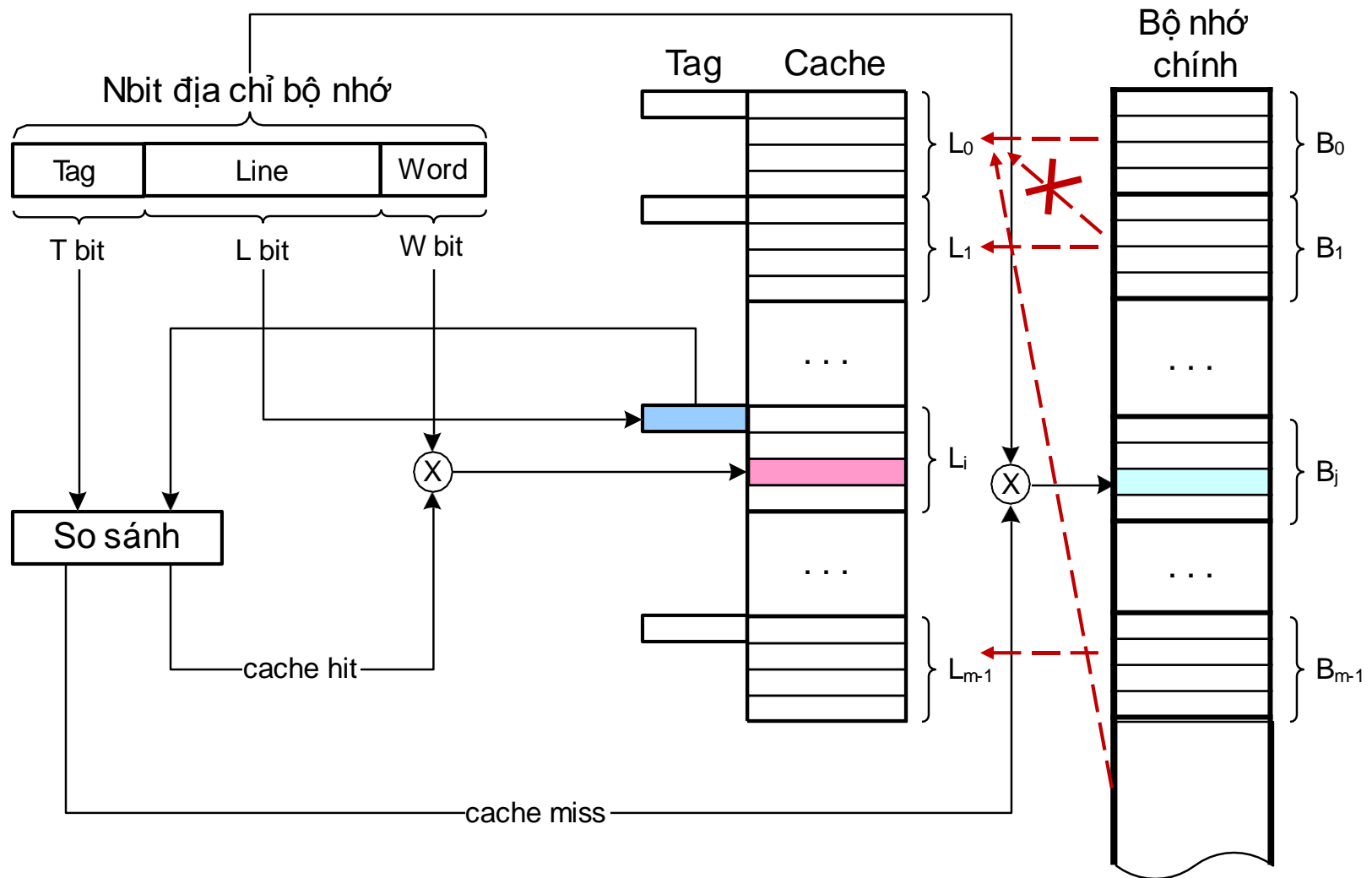
(Chính là các phương pháp tổ chức bộ nhớ cache)

- 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*.

Minh họa ánh xạ trực tiếp



Đặc điểm của ánh xạ trực tiếp

- Mỗi một địa chỉ N bit của bộ nhớ chính gồm ba trường:
 - Trường **Word** gồm W bit xác định một từ nhớ trong *Block* hay *Line*:
 $2^W = \text{kích thước của Block hay Line}$
 - Trường **Line** gồm L bit xác định một trong số các *Line* trong *cache*:

$2^L = \text{số Line trong cache}$

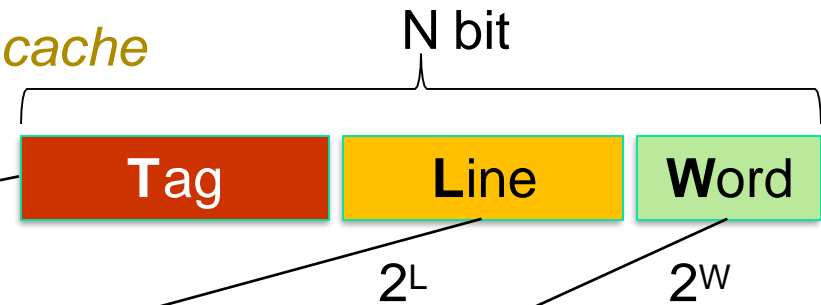
- Trường **Tag** gồm T bit:

$$T = N - (W + L)$$

- Mỗi quan hệ

$$\text{<Kích thước Cache>} = \text{<Số line>} * \text{<Kích thước line>}$$

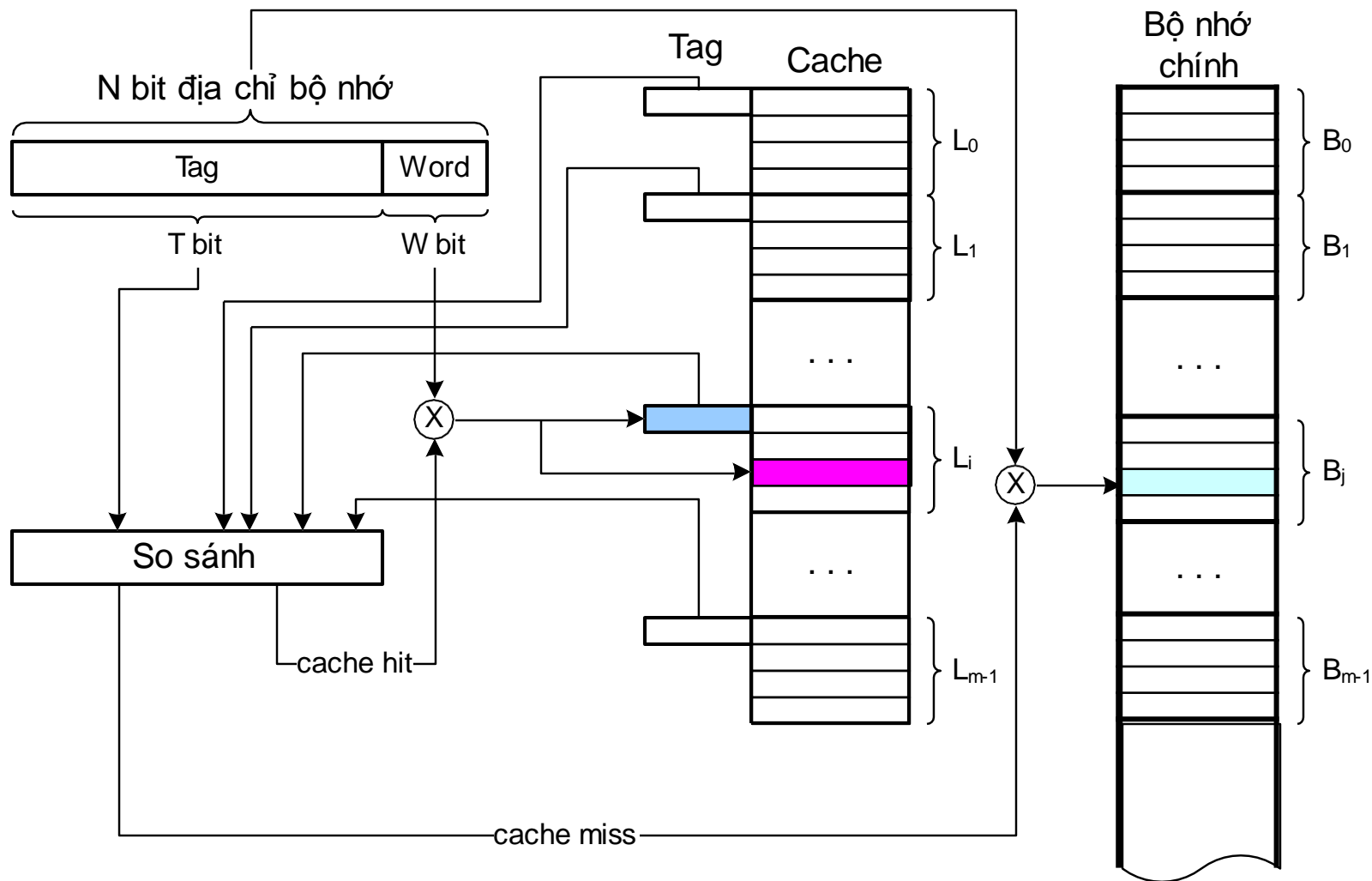
- Bộ so sánh đơn giản
- Xác suất *cache hit* thấp



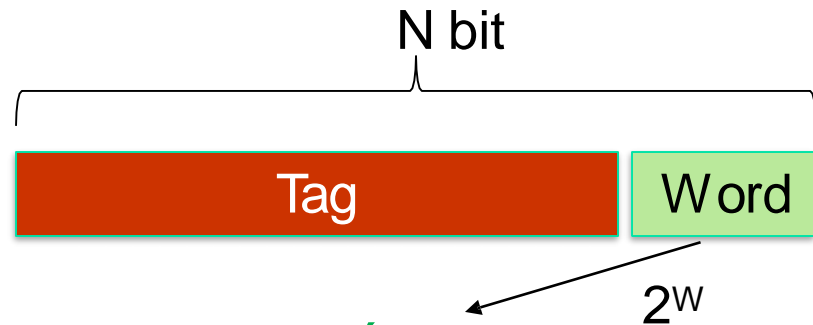
Ảnh xạ liên kết toàn phần

- Mỗi *Block* có thể nạp vào bất kỳ *Line* nào của *cache*.
- Địa chỉ của bộ nhớ chính bao gồm hai trường:
 - Trường **Word** giống như trường hợp ở trên.
 - Trường **Tag** dùng để xác định *Block* của bộ nhớ chính.
- Tag xác định *Block* đang nằm ở *Line* đó

Minh hoạ ánh xạ liên kết toàn phần



Đặc điểm của ánh xạ liên kết toàn phần

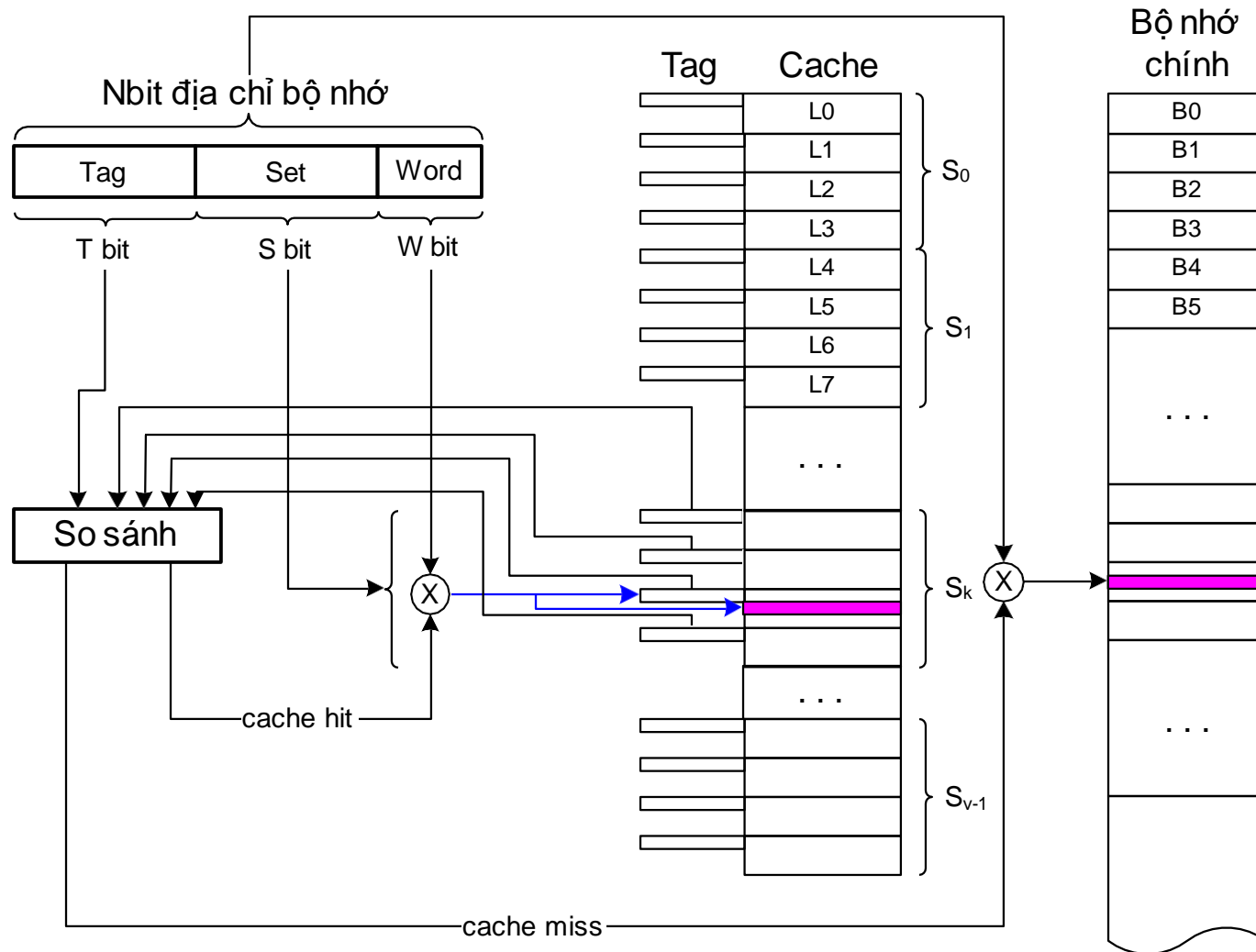


- Mỗi quan hệ:
 $\text{<Kích thước Line>} = 2^{\text{<Số bit Word>}}$
- So sánh đồng thời với tất cả các Tag → mất nhiều thời gian
- Xác suất *cache hit* cao.
- Bộ so sánh phức tạp.

Ánh xạ liên kết tập hợp

- 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:
 - $B_0 \rightarrow S_0 \rightarrow$ line bất kì thuộc S_0
 - $B_1 \rightarrow S_1 \rightarrow$ line bất kì thuộc S_1
 - $B_2 \rightarrow S_2 \rightarrow$ line bất kì thuộc S_2
 - $B_3 \rightarrow S_3 \rightarrow$ line bất kì thuộc S_3
 - $B_4 \rightarrow S_0 \rightarrow$ line bất kì thuộc S_0


Minh hoạ ánh xạ liên kết tập hợp



Đặc điểm của ánh xạ liên kết tập hợp

- Kích thước Block = 2^W Word
- Trường Set có S bit dùng để xác định một trong số $V = 2^S$ Set
- Trường Tag có T bit: $T = N - (W+S)$

- Mỗi quan hệ



The diagram shows three adjacent rectangular boxes. The first box is orange and labeled 'Tag'. The second box is pink and labeled 'Set'. The third box is light green and labeled 'Word'. An arrow points from the 'Tag' box to the text '' in the formula below. Another arrow points from the 'Set' box to the text '<Số line/set>' in the formula. A third arrow points from the 'Word' box to the text '<Kích thước line>' in the formula. Below the boxes, the text '<Kích thước cache> =' is followed by the formula: '<Số set> * <Số line/set> * <Kích thước line>'. The 'Số set' is in pink, '<Số line/set>' is in blue, and '<Kích thước line>' is in green. Above the 'Set' box, the text ' 2^S ' is written. Above the 'Word' box, the text ' 2^W ' is written.

$$\text{<Kích thước cache>} = \text{<Số set>} * \text{<Số line/set>} * \text{<Kích thước line>}$$

- Tổng quát cho cả hai phương pháp trên
- Thông thường 2,4,8,16 Lines/Set

Ví dụ về ánh xạ địa chỉ

- 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 $\rightarrow N = 32$ bit
- $Line = 32$ byte = 2^5 byte $\rightarrow W = 5$ bit
- $Cache = 256$ KiB = 2^{18} byte.
- $\langle \text{Kích thước Cache} \rangle = \langle \text{Số line} \rangle * \langle \text{Kích thước line} \rangle$
 $256 \text{ KiB} = 2^{18} = \langle \text{Số line} \rangle * 32B$
- Số Line trong cache = $2^{18} / 2^5 = 2^{13}$ Line
 $\rightarrow L = 13$ bit
- $T = N - (W + L) = 32 - (13 + 5) = 14$ bit

Tag	Line	Word
14 bit	13 bit	5 bit

Với ánh xạ liên kết toàn phần

- Bộ nhớ chính = 4GiB = 2^{32} byte \rightarrow N = 32 bit
- $Line = 32$ byte = 2^5 byte \rightarrow W = 5 bit
- Số bit của trường *Tag* sẽ là: T = 32 - 5 = 27 bit

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

Với ánh xạ liên kết tập hợp 4 đường

- Bộ nhớ chính = 4GiB = 2^{32} byte \rightarrow N = 32 bit
 - Line = 32 byte = 2^5 byte \rightarrow W = 5 bit
 - $\langle \text{Kích thước cache} \rangle = \langle \text{Số set} \rangle * \langle \text{Số line/set} \rangle * \langle \text{Kích thước line} \rangle$
 $256 \text{ KiB} = 2^{18} = \langle \text{Số set} \rangle * 4 * 32\text{B}$
- \rightarrow số Set trong cache = $2^{18} / 2^2 / 2^5 = 2^{11}$ Set \rightarrow
S = 11 bit
- Số bit của trường Tag sẽ là: $T = 32 - (11 + 5)$
= 16 bit

Tag	Set	Word
16 bit	11 bit	5 bit

Bài tập

Giả thiết rằng máy tính có **128KiB cache** tổ chức theo kiểu ánh xạ **liên kết tập hợp 4-line**. Cache có tất cả là **1024 Set** từ S0 đến S1023. Địa chỉ bộ nhớ chính là **32-bit** và đánh địa chỉ cho từng byte.

a) Tính số bit cho các trường địa chỉ khi truy nhập cache ?

b) Xác định byte nhớ có địa chỉ $003D02AF_{(16)}$ được ánh xạ vào Set nào của cache ?

Đáp án

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