COMPUTER ARCHITECTURE
CSE Fall 2017

Faculty of Computer Science and Engineering

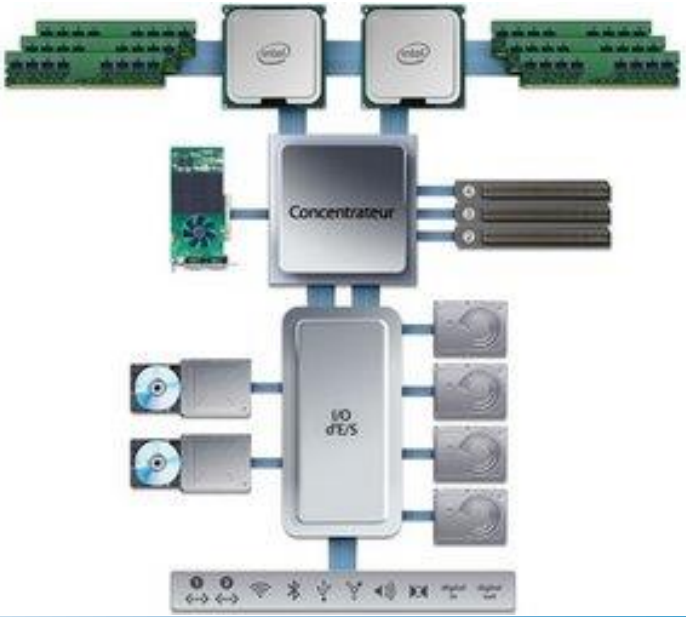
Department of Computer Engineering

Vo Tan Phuong

http://www.cse.hcmut.edu.vn/~vtphuong

Chapter 5

Bộ nhớ máy tính

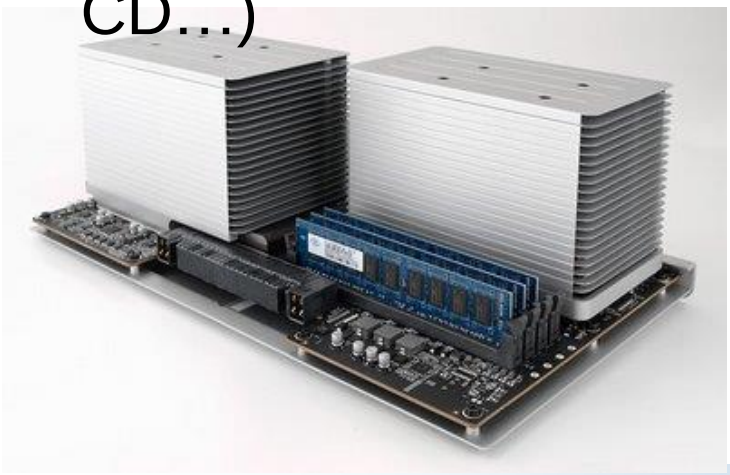


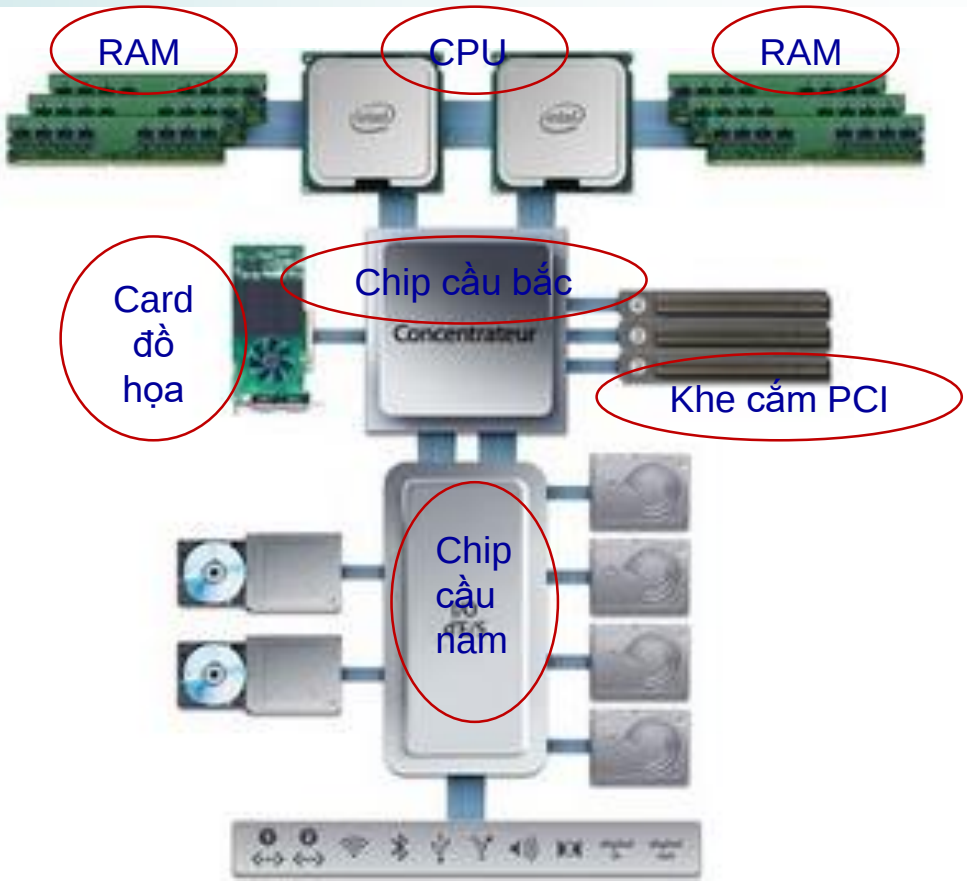
Nội dung trình bày

- Công nghệ và thuật ngữ liên quan đến bộ nhớ
- Tổ chức / thiết kế bộ nhớ
- Sự cần thiết phải có bộ nhớ đệm
- Phân loại bộ nhớ đệm
- Đánh giá hiệu năng của bộ nhớ đệm

Máy tính PC

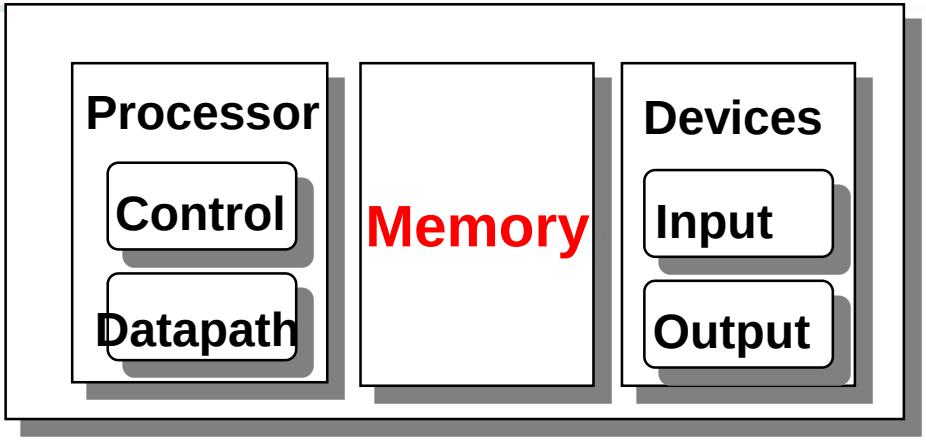
- Nhìn từ bên ngoài, bộ xử lý kết nối với RAM và chip cầu bắc (quản lý các kết nối vào bus tốc độ cao như card đồ họa, khe PCI), chip cầu bắc nối với chip cầu nam (quản lý USB bus, network, ổ cứng, ổ CD…)

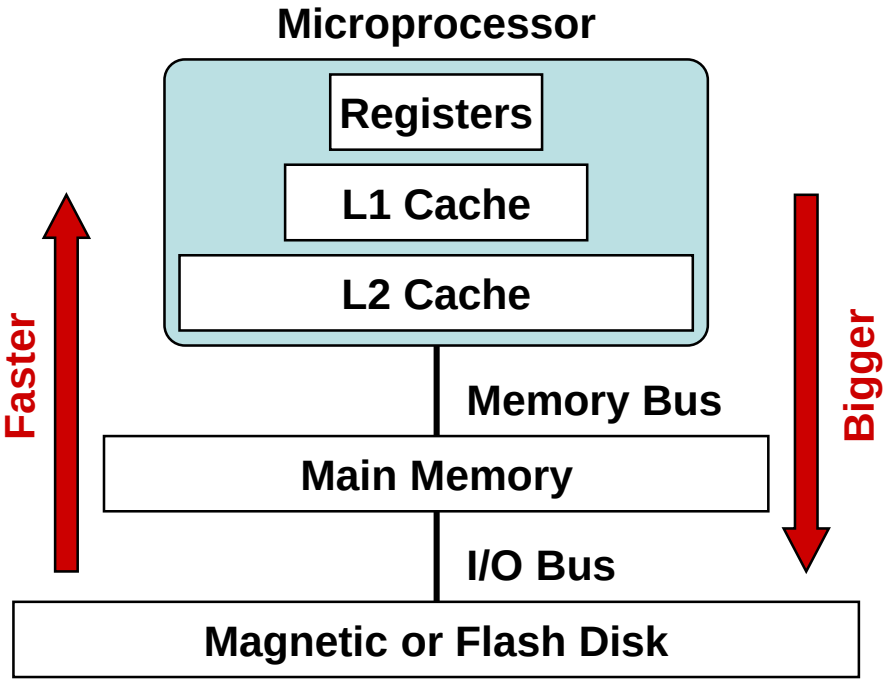




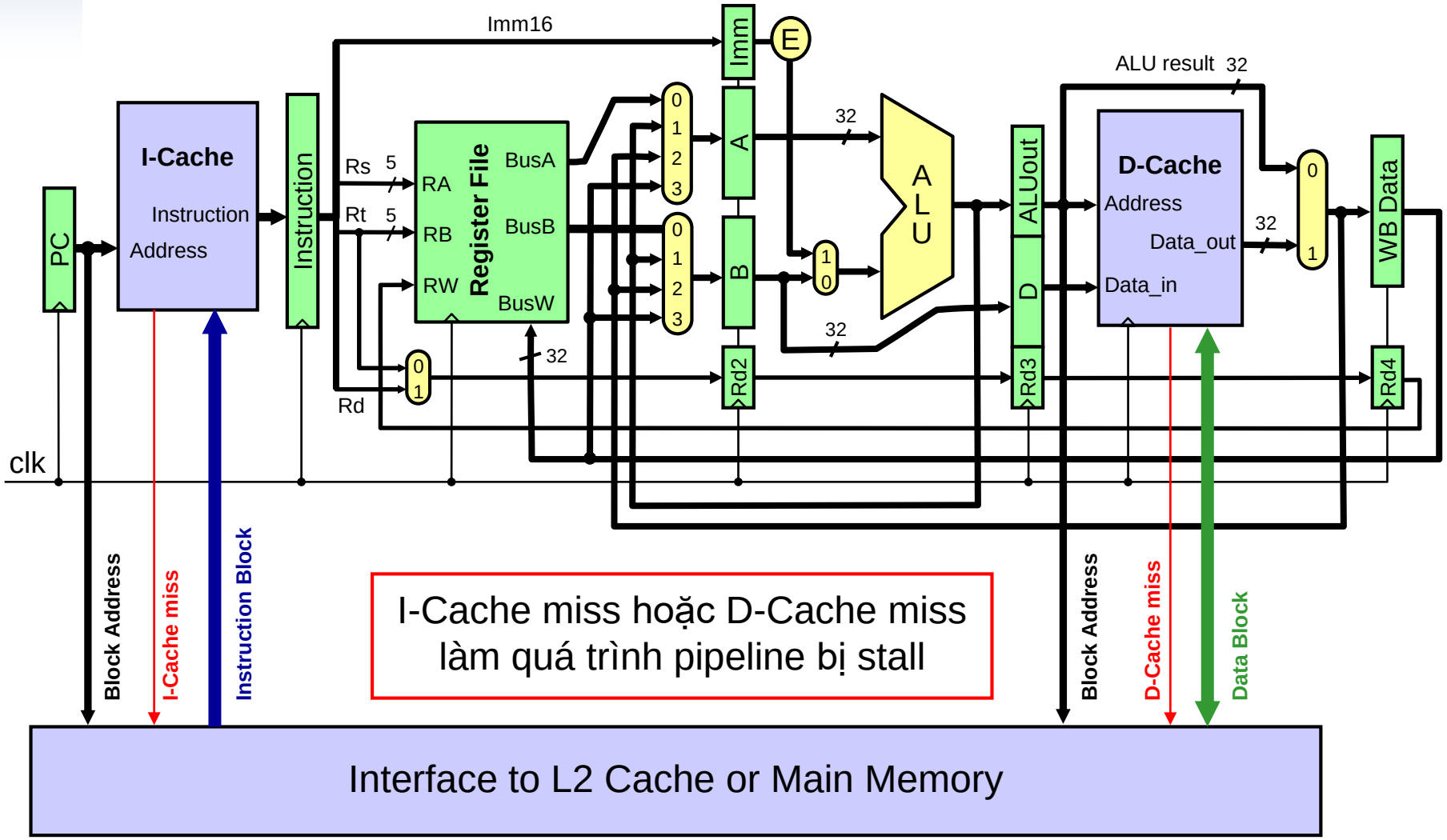
Hệ thống bộ nhớ máy tính

- Bộ nhớ máy tính hiểu theo nghĩa rộng là tất cả nơi chứa dữ liệu (thanh ghi, bộ nhớ đệm, bộ nhớ chính, bộ nhớ ngoài)
- Hệ thống bộ nhớ tổ chức theo mô hình phân cấp, trên cùng là bộ thanh ghi, đến bộ nhớ đệm L1, L2, bộ nhớ chính, bộ nhớ phụ



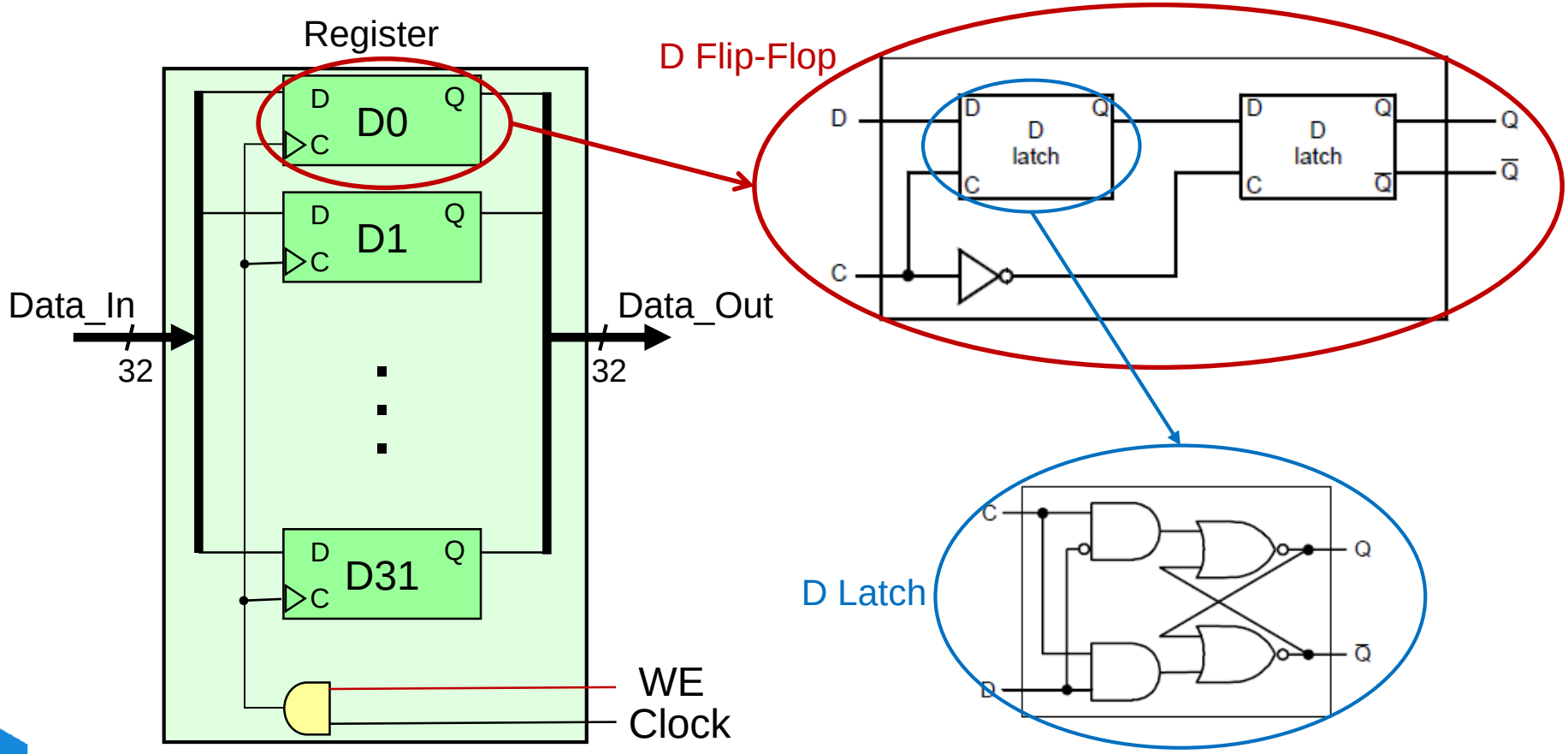


Hệ thống bộ nhớ phân cấp trong Pipline MIPS CPU



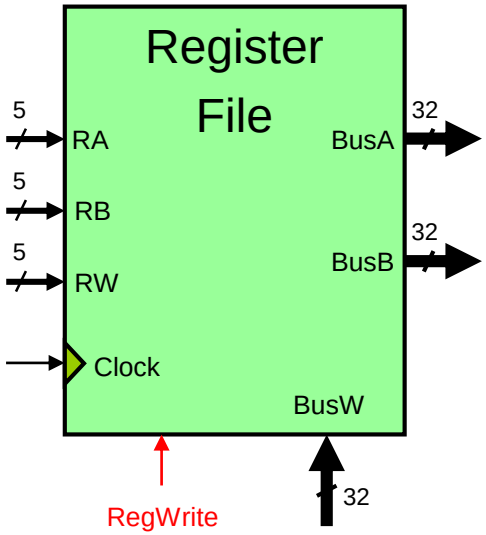
Cấu tạo thanh ghi

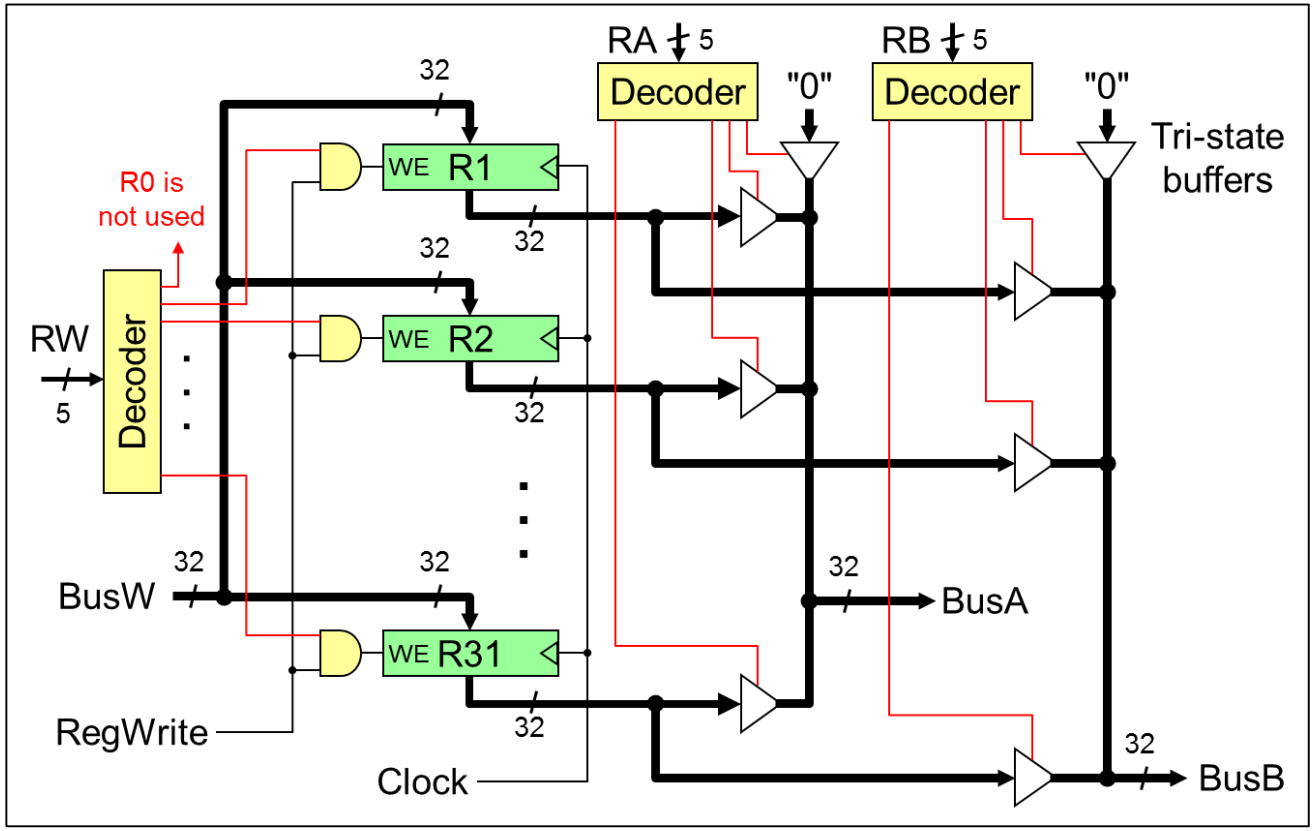
- Thanh ghi có thành phần chính là các D Flip-Flop, có n bit dữ liệu vào, n bit dữ liệu ra, tín hiệu WE (Write Enable) và tín hiệu xung nhịp Clock



Chi tiết bộ thanh ghi

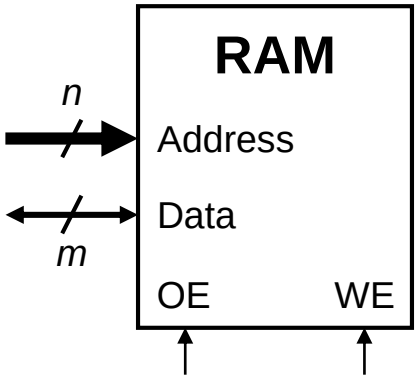
Bộ thanh ghi MIPS gồm 32 thanh ghi 32 bit; RA, RB là 2 đầu vào yêu cầu truy xuất dữ liệu, dữ liệu được đưa ra BusA, BusB; để ghi dữ liệu cần chỉ ra nơi lưu RW, giá trị lưu BusW, yêu cầu ghi RegWrite và thời điểm ghi được đồng bộ với xung nhịp Clock





Bộ nhớ truy xuất ngẫu nhiên RAM

- Là một mảng 2^n phần tử nhớ, mỗi phần tử nhớ lưu trữ m-bit dữ liệu
- Là bộ nhớ "bốc hơi"
  - Dữ liệu chỉ được lưu khi còn được cung cấp điện
- Truy xuất ngẫu nhiên
  - Thời gian truy xuất dữ liệu từ một phần tử nhớ bất kỳ là như nhau
- Tín hiệu điều khiển Output Enable (OE)
  - Yêu cầu xuất dữ liệu khi "đọc"
- Tín hiệu điều khiển Write Enable (WE)
  - Yêu cầu ghi dữ liệu

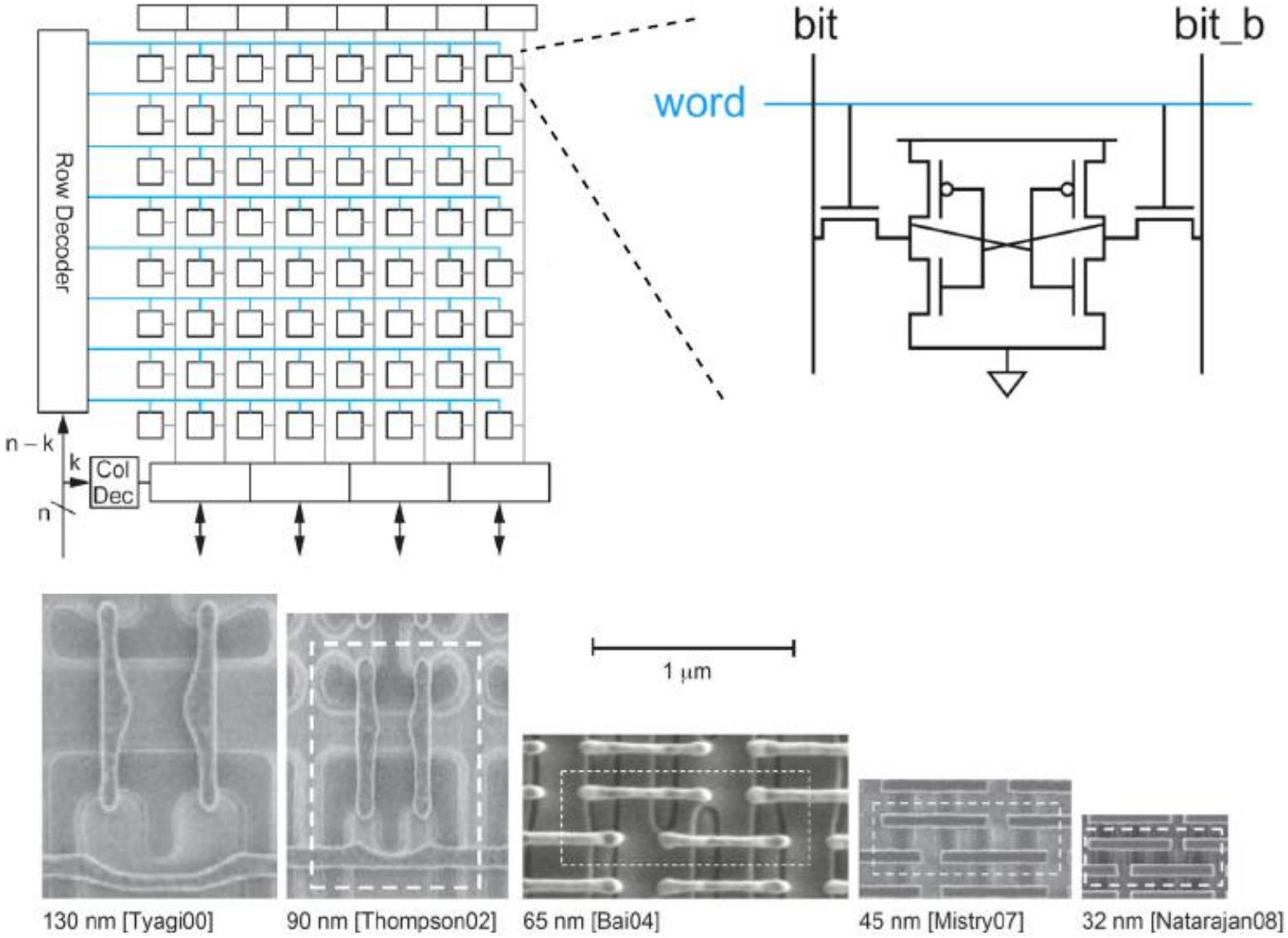


- $2^n \times m$ RAM : n-bit địa chỉ, m-bit dữ liệu

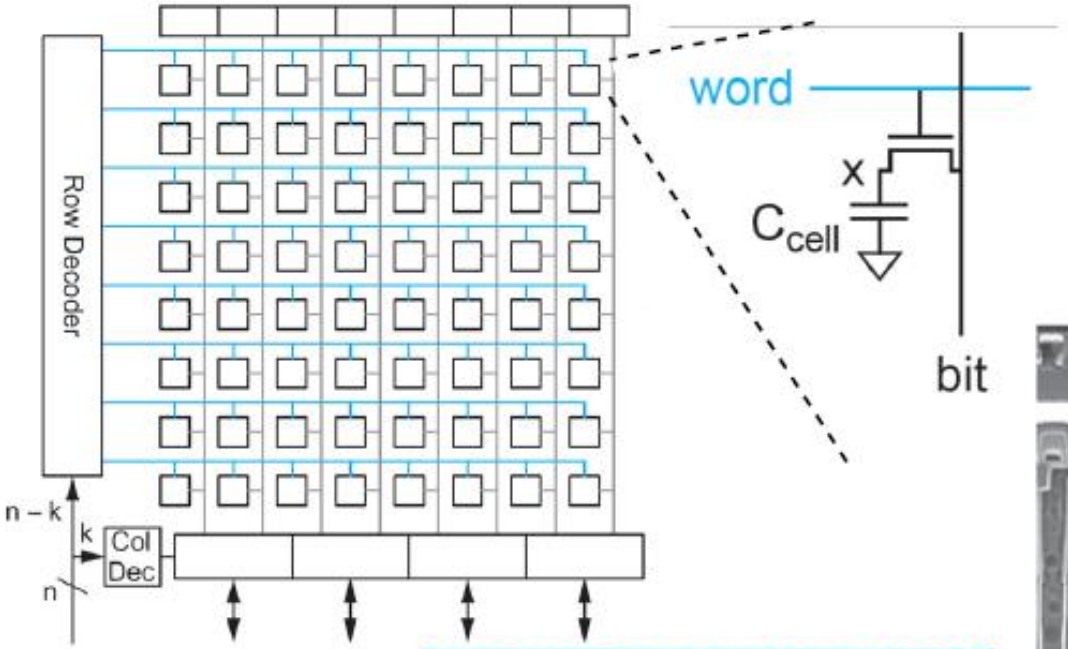
Công nghệ bộ nhớ

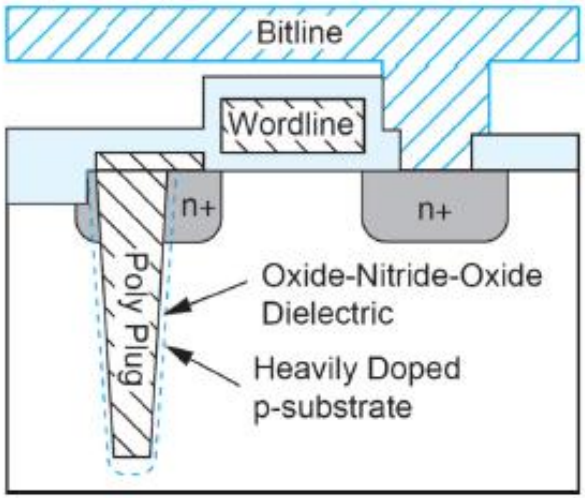
- Static RAM (SRAM) dùng cho bộ nhớ đệm
  - 1 bit nhớ cần 6 transistor
  - Tốc độ truy xuất dữ liệu cao (~ 1ns)
  - Cần ít năng lượng để duy trì giá trị lưu trữ
- Dynamic RAM (DRAM) dùng cho bộ nhớ chính
  - 1 bit nhớ cần 1 transistor + 1 capacitor
  - Tốc độ truy xuất dữ liệu thấp (~ 100ns)
  - Cần phải ghi lại giá trị vào ô nhớ sau khi đọc
  - Cần phải định kỳ "làm tươi"
    - Mỗi hàng có thể được làm tươi đồng thời

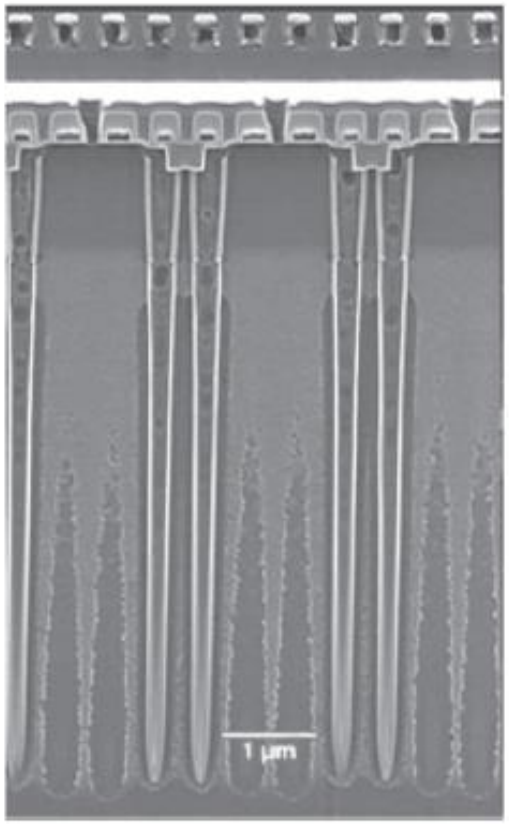
Cấu tạo SRAM


Row Decoder
n – k
k
Col Dec
n
bit
bit_b
word
1 μm
130 nm [Tyagi00]
90 nm [Thompson02]
65 nm [Bai04]
45 nm [Mistry07]
32 nm [Natarajan08]

Cấu tạo DRAM

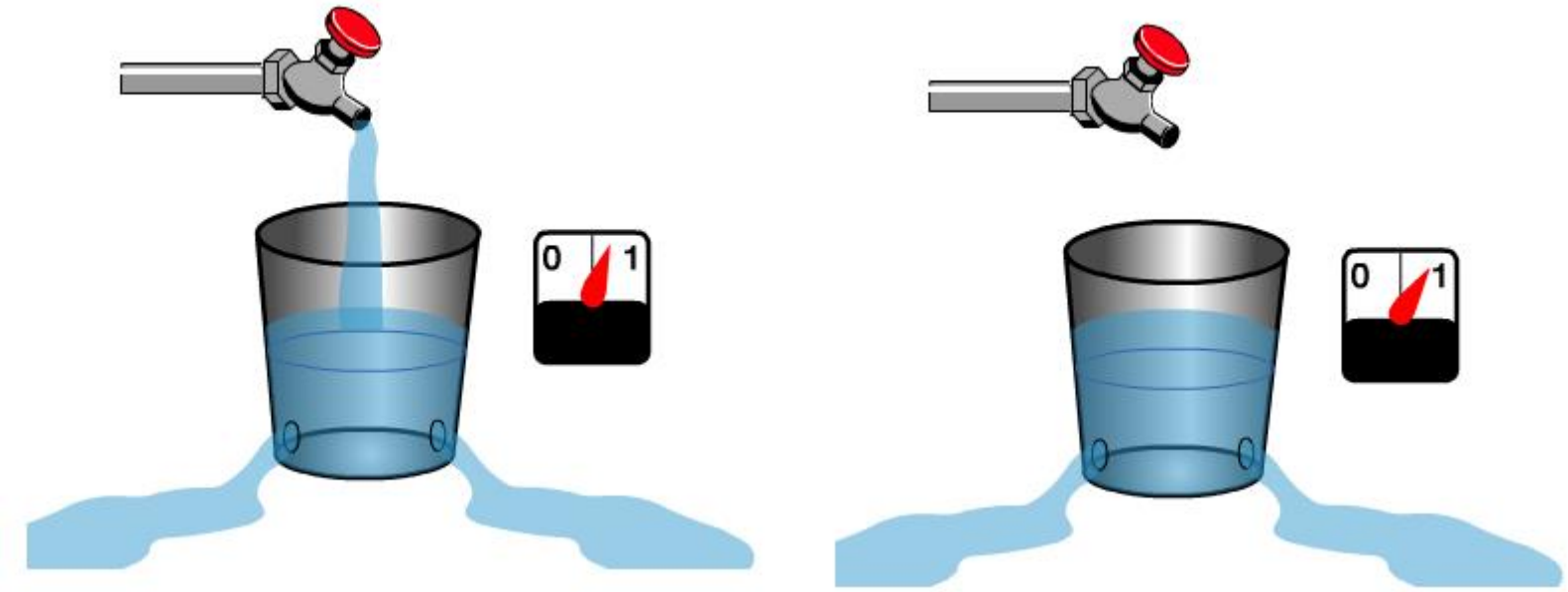

Row Decoder
word
x
C_{cell}
bit
n – k
k
Col Dec
n


Bitline
Wordline
n+
n+
Poly Plug
Oxide-Nitride-Oxide Dielectric
Heavily Doped p-substrate


1 μm

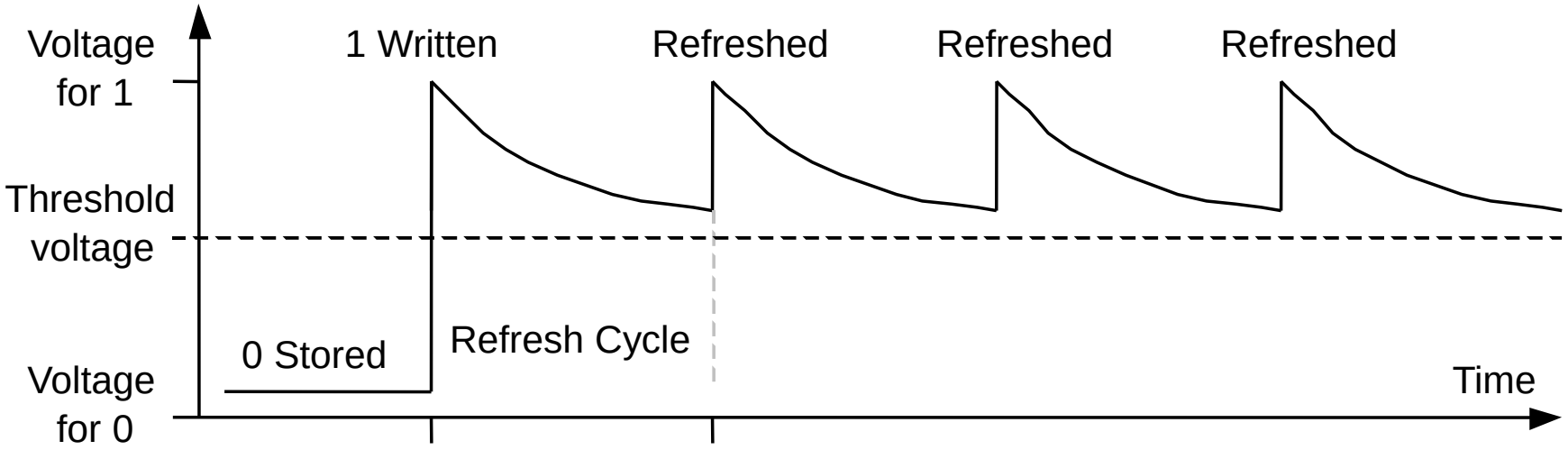
Mô hình bit nhớ của DRAM

- 1 bit nhớ sử dụng tụ làm phần tử lưu trữ
- Tụ có đặt tính "rò điện tích" theo thời gian
- Cần "làm tươi" để giữ mức điện thế tương ứng mức 1



Chu kỳ làm tươi DRAM

- Chu kỳ làm tươi (refresh cycle) vào khoảng 10ms
- Việc làm tươi được thực hiện cho toàn bộ nhớ
- Mỗi hàng sẽ được đọc và ghi trở lại để phục hồi điện tích
- Băng thông bộ nhớ bị giảm bởi việc làm tươi



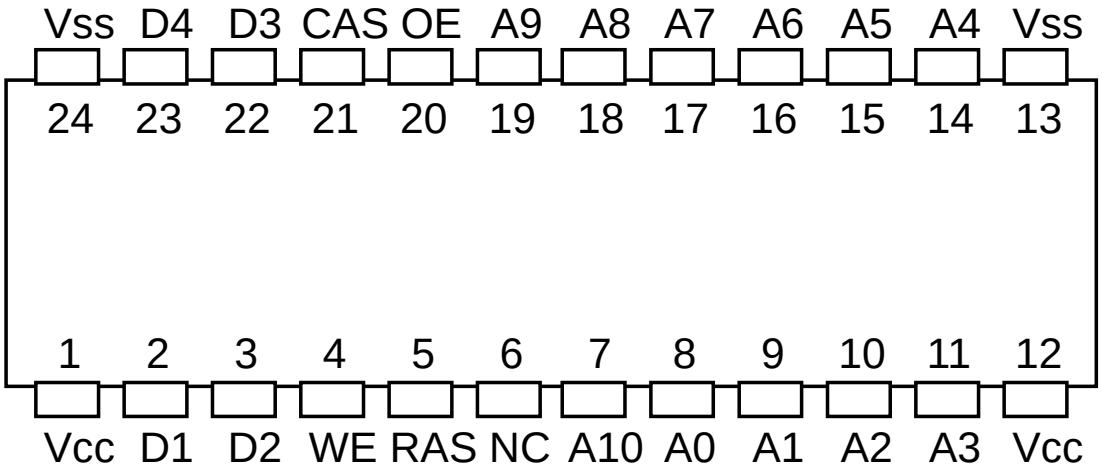
Ví dụ một IC DRAM

- 24 chân, dạng dual in-line cho bộ nhớ 16Mbit = $2^{22} \times 4$
- 22-bit địa chỉ bao gồm
  - 11-bit row address
  - 11-bit column address
  - Địa chỉ hàng và cột dùng chung

11 đường địa chỉ A0-A10

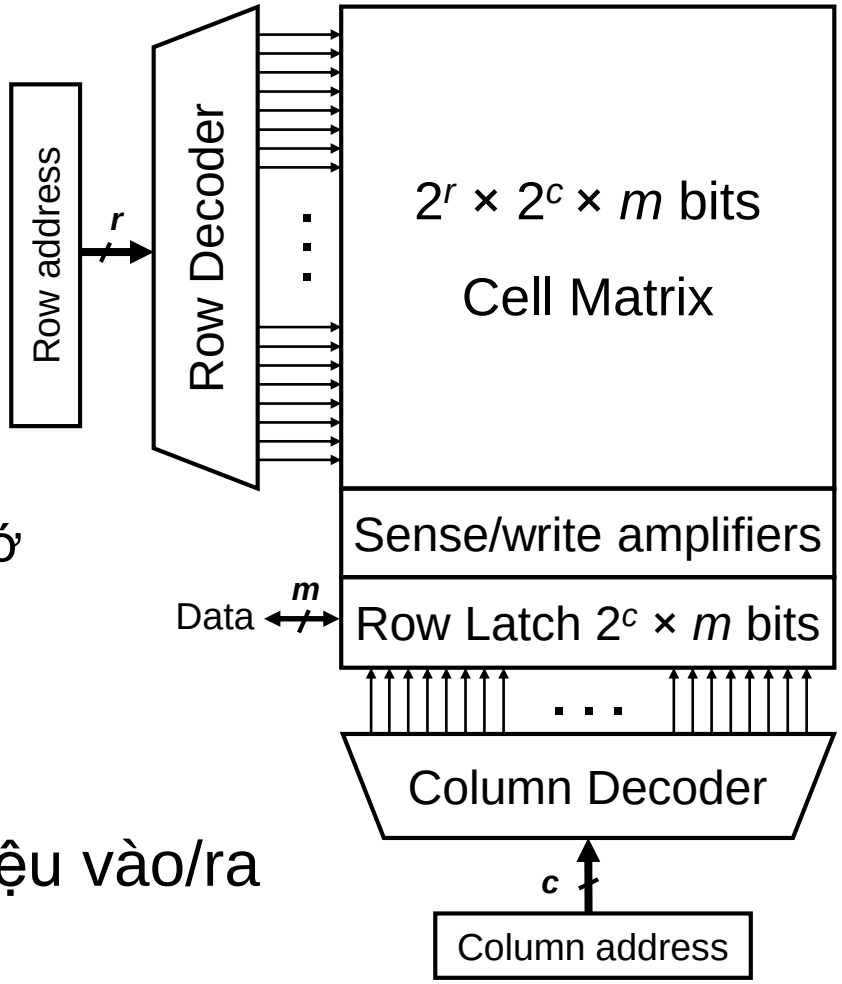
Chú thích:

A*i*	Address bit *i*
CAS	Column address strobe
D*j*	Data bit *j*
NC	No connection
OE	Output enable
RAS	Row address strobe
WE	Write enable



Ví dụ cấu trúc của DRAM

- Row decoder
  - Lựa chọn hàng để đọc/ghi
- Column decoder
  - Lựa chọn cột để đọc/ghi
- Cell Matrix
  - Mảng 2 chiều các phần tử nhớ
- Sense/Write amplifiers
  - Làm rõ mức 0/1 khi đọc/ghi
- Sử dụng chung đường dữ liệu vào/ra



Thông số DRAM

Year Produced	Chip size	Type	Row access	Column access	Cycle Time New Request
1980	64 Kbit	DRAM	170 ns	75 ns	250 ns
1983	256 Kbit	DRAM	150 ns	50 ns	220 ns
1986	1 Mbit	DRAM	120 ns	25 ns	190 ns
1989	4 Mbit	DRAM	100 ns	20 ns	165 ns
1992	16 Mbit	DRAM	80 ns	15 ns	120 ns
1996	64 Mbit	SDRAM	70 ns	12 ns	110 ns
1998	128 Mbit	SDRAM	70 ns	10 ns	100 ns
2000	256 Mbit	DDR1	65 ns	7 ns	90 ns
2002	512 Mbit	DDR1	60 ns	5 ns	80 ns
2004	1 Gbit	DDR2	55 ns	5 ns	70 ns
2006	2 Gbit	DDR2	50 ns	3 ns	60 ns
2010	4 Gbit	DDR3	35 ns	1 ns	37 ns
2012	8 Gbit	DDR3	30 ns	0.5 ns	31 ns

SDRAM và DDR SDRAM

- SDRAM: Synchronous Dynamic RAM
  - Thêm tín hiệu clock vào DRAM
- SDRAM được đồng bộ với xung nhịp hệ thống
  - DRAM với công nghệ cũ là loại bất đồng bộ
  - Khi xung nhịp hệ thống tăng, SDRAM có hiệu năng cao hơn DRAM bất đồng bộ
- DDR: Double Data Rate SDRAM
  - Giống với SDRAM, DDR đồng bộ với xung nhịp hệ thống, nhưng khác ở chỗ DDR độc dữ liệu tại cạnh lên và cạnh xuống của tín hiệu xung nhịp

Transfer Rates & Peak Bandwidth

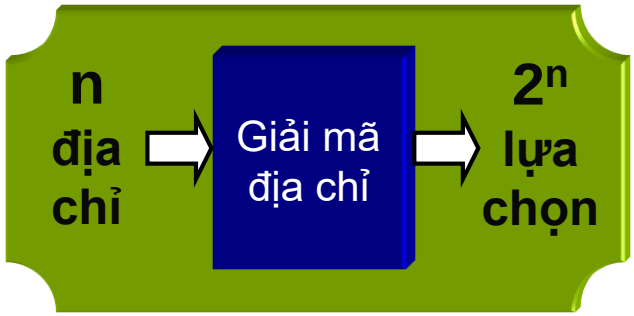
Standard Name	Memory Bus Clock	Millions Transfers per second	Module Name	Peak Bandwidth
DDR-200	100 MHz	200 MT/s	PC-1600	1600 MB/s
DDR-333	167 MHz	333 MT/s	PC-2700	2667 MB/s
DDR-400	200 MHz	400 MT/s	PC-3200	3200 MB/s
DDR2-667	333 MHz	667 MT/s	PC-5300	5333 MB/s
DDR2-800	400 MHz	800 MT/s	PC-6400	6400 MB/s
DDR2-1066	533 MHz	1066 MT/s	PC-8500	8533 MB/s
DDR3-1066	533 MHz	1066 MT/s	PC-8500	8533 MB/s
DDR3-1333	667 MHz	1333 MT/s	PC-10600	10667 MB/s
DDR3-1600	800 MHz	1600 MT/s	PC-12800	12800 MB/s
DDR4-3200	1600 MHz	3200 MT/s	PC-25600	25600 MB/s

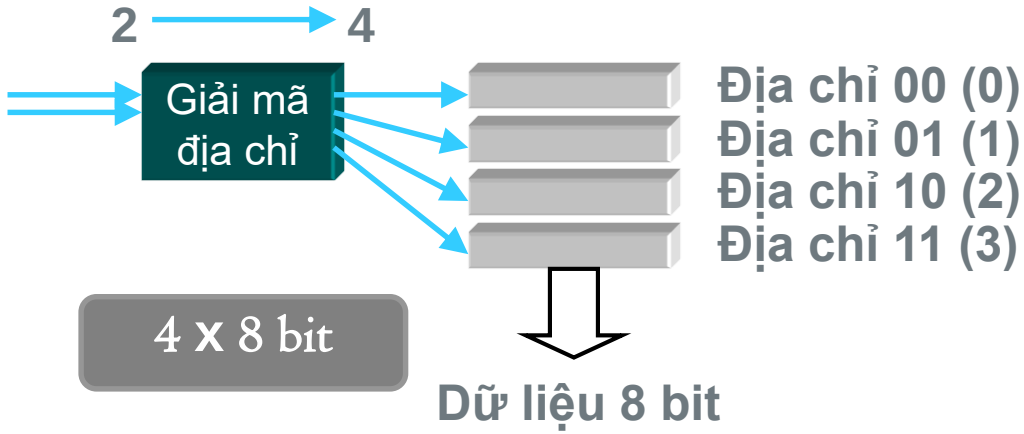
- 1 Transfer = 64 bits = 8 bytes of data

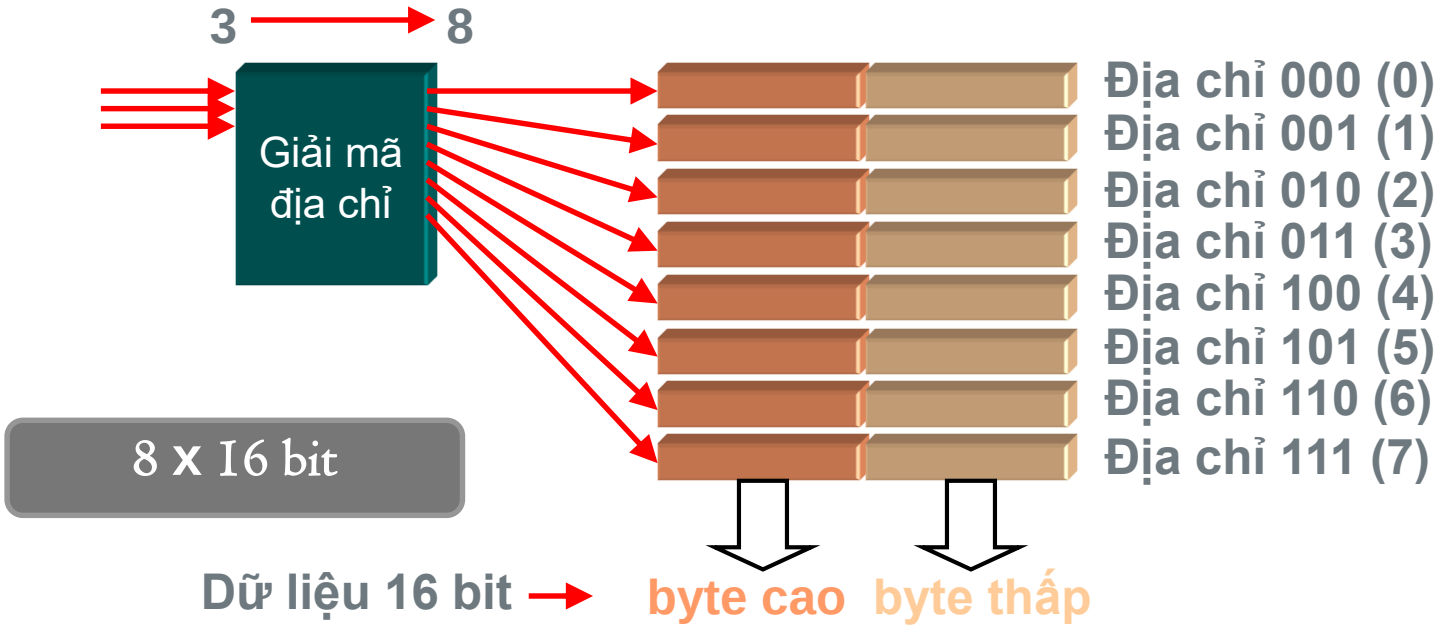
Nội dung trình bày

- Công nghệ và thuật ngữ liên quan đến bộ nhớ
- Tổ chức / thiết kế bộ nhớ
- Sự cần thiết phải có bộ nhớ đệm
- Phân loại bộ nhớ đệm
- Đánh giá hiệu năng của bộ nhớ đệm

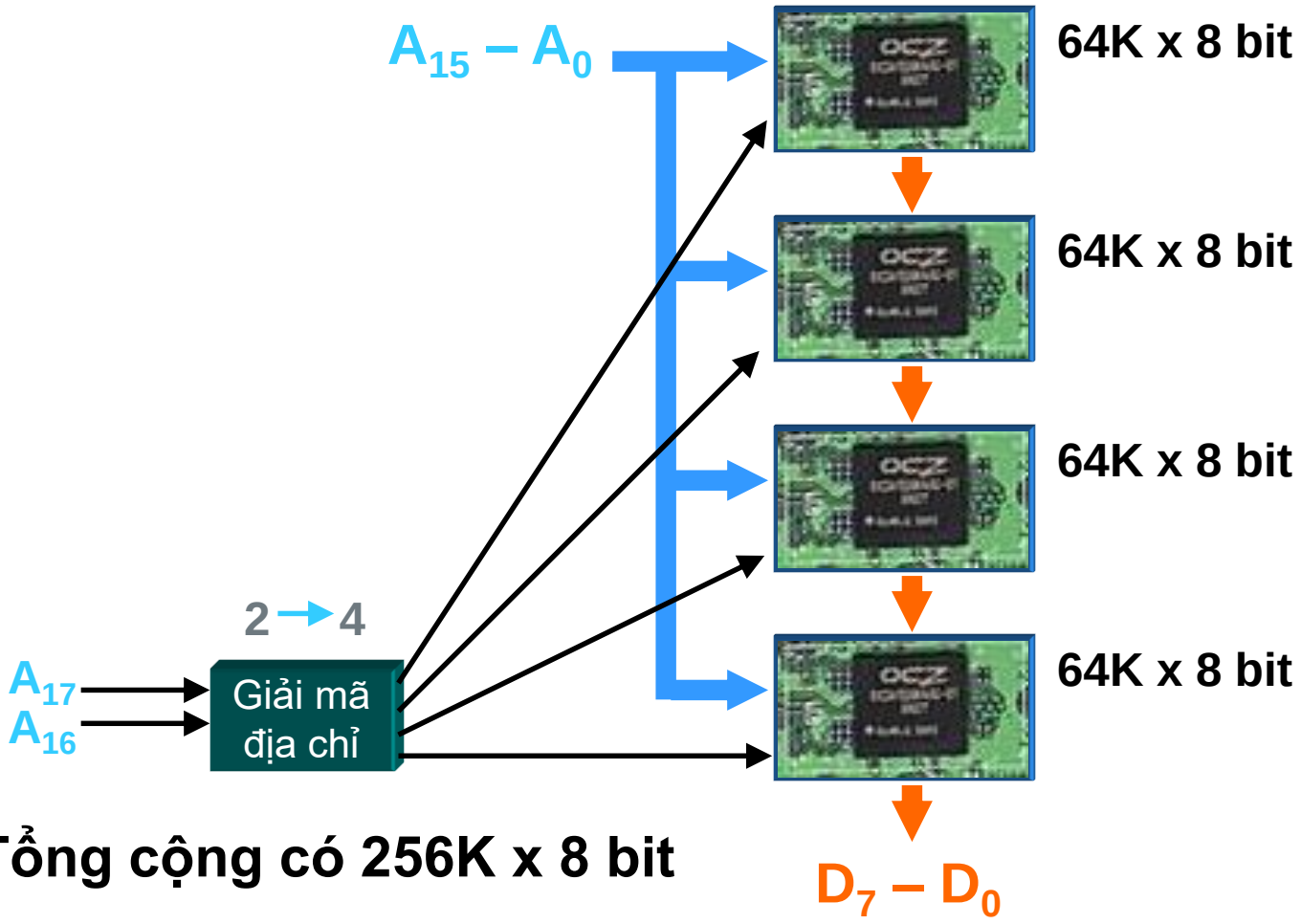
Tổ chức bộ nhớ





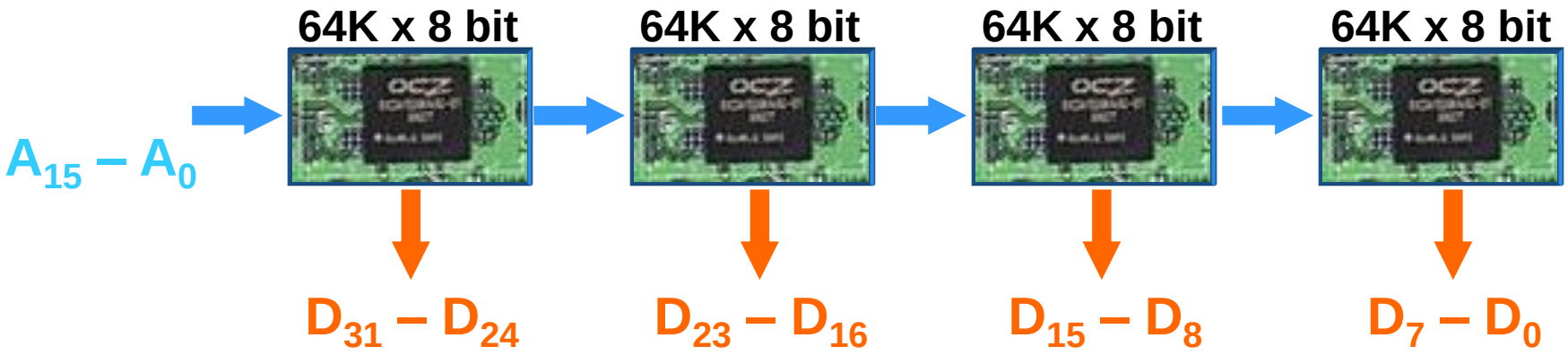


Tổ chức theo dung lượng



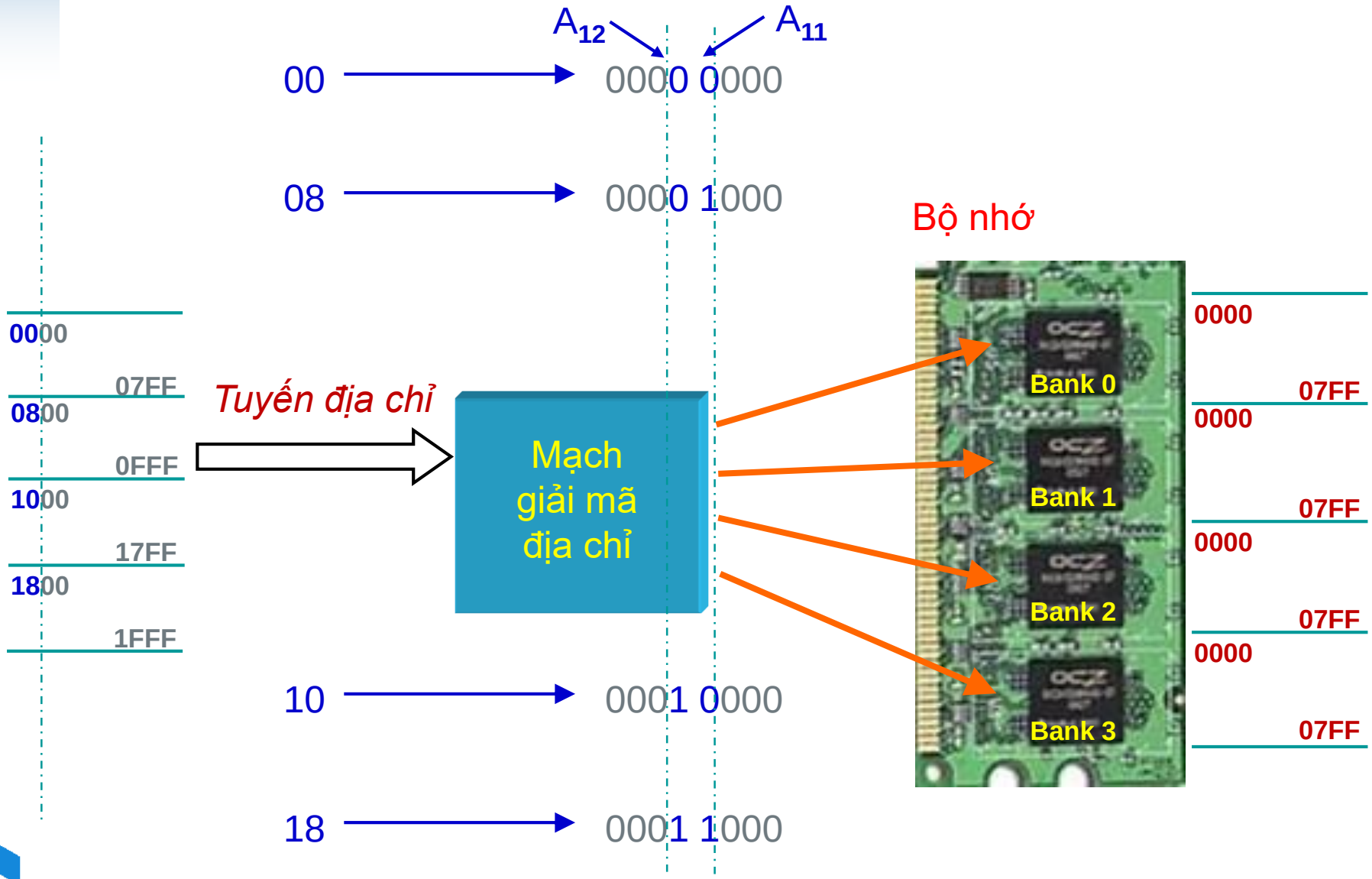
Tổng cộng có 256K x 8 bit

Tổ chức theo kích thước



Tổng cộng có 64K x 32 bit

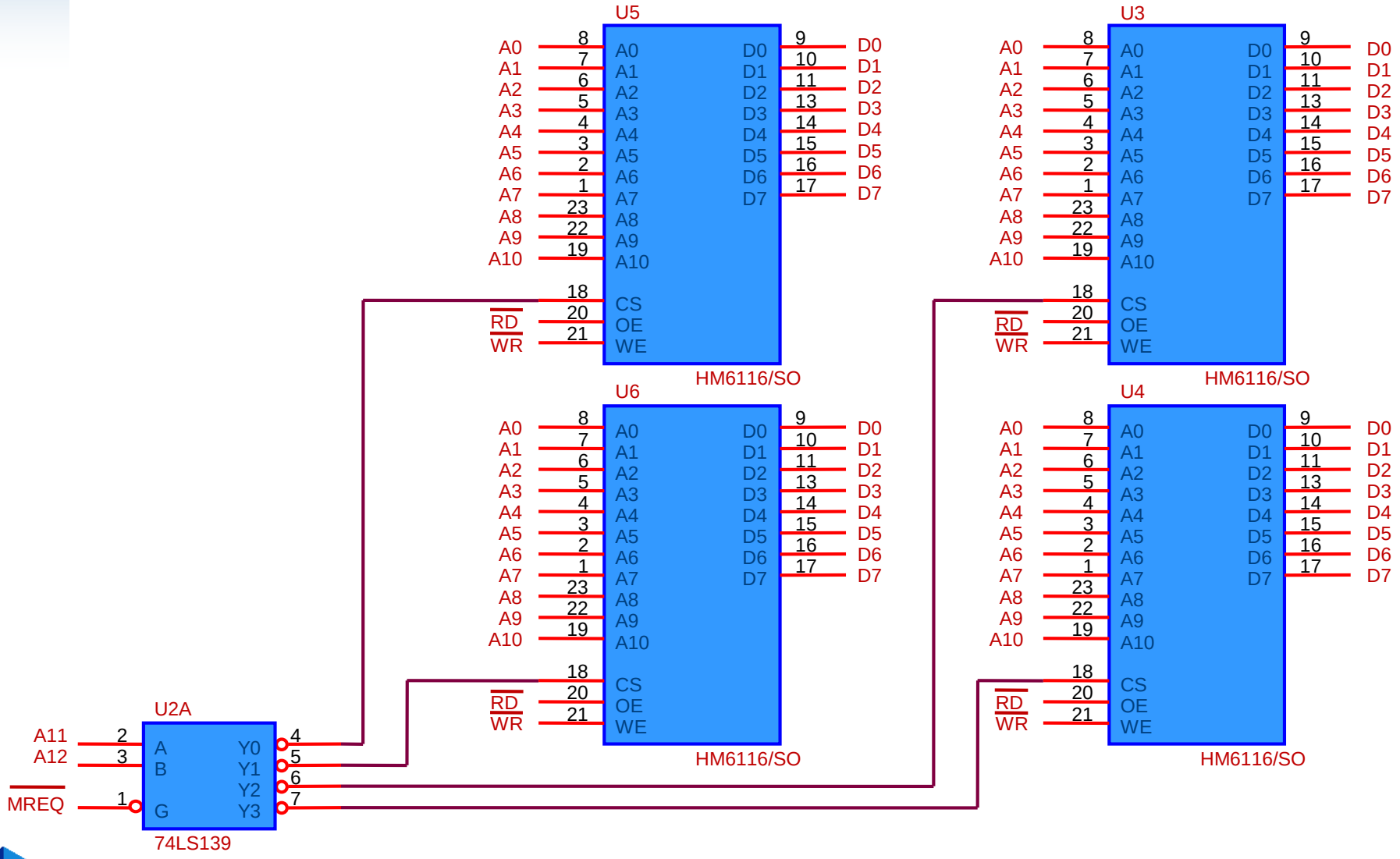
Giải mã địa chỉ bộ nhớ



Giải mã địa chỉ bộ nhớ

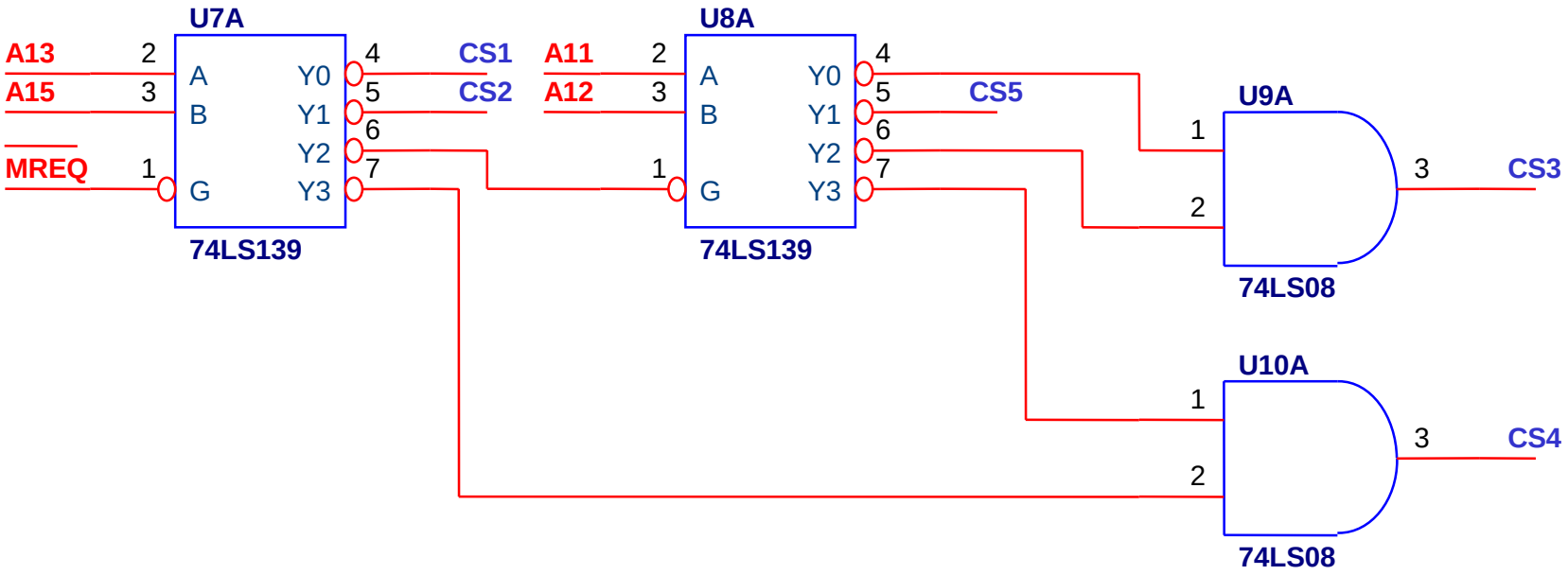
Địa chỉ	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
0000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
01	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
. . .																
07FF	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
0800	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0801	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
. . .																
0FFF	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
1000	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1001	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
. . .																
17FF	0	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1
1800	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
1801	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1
. . .																
1FFF	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1

Mạch chi tiết



Bài tập (1)

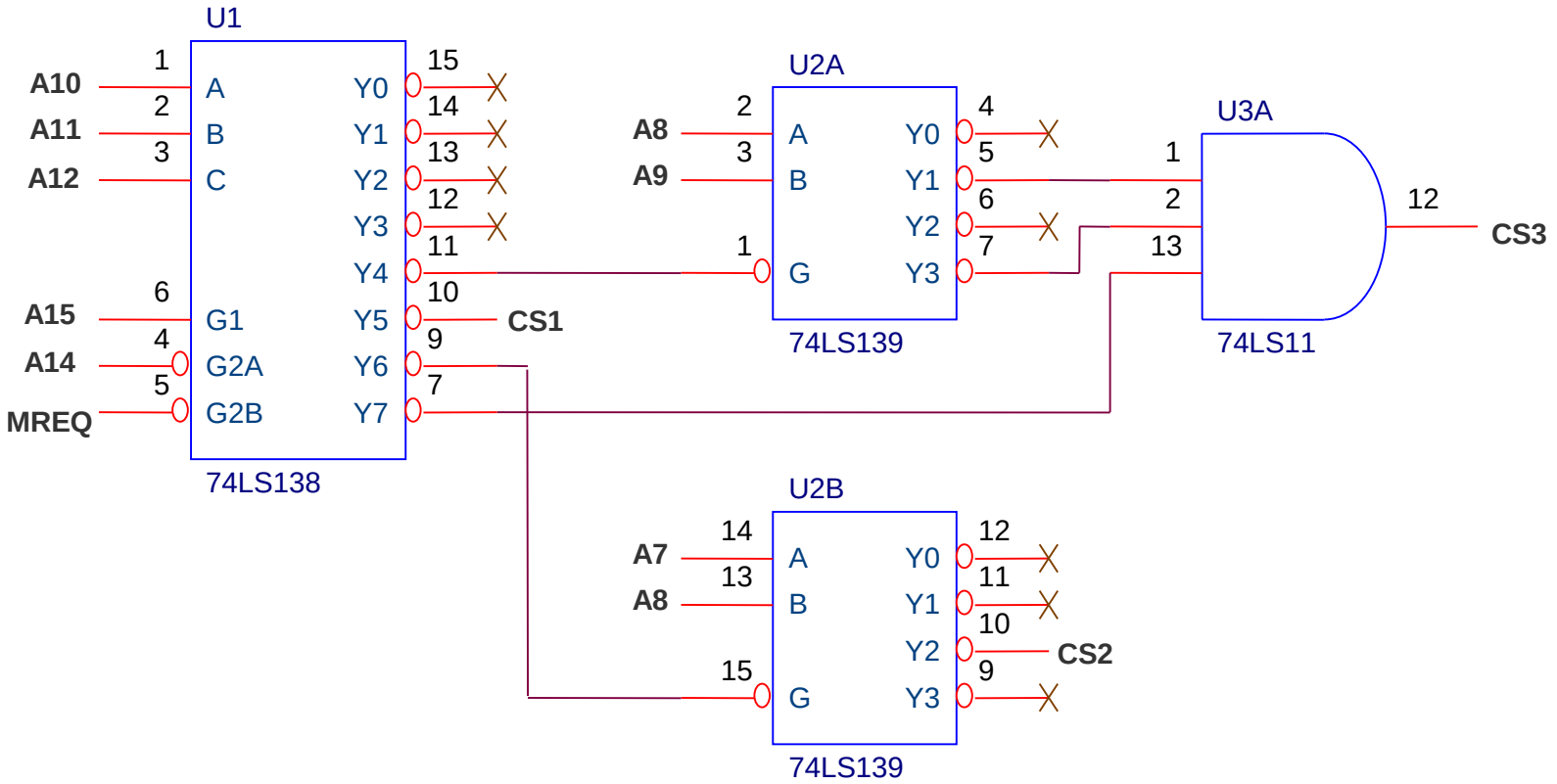
1) Tính các địa chỉ CS1, CS2, CS3, CS4



2) Vẽ mạch giải mã địa chỉ cho bộ nhớ cho các trường hợp sau :
   a. 14KB = 2 x 4KB + 2 x 2KB + 2 x 1KB
   b. 32KB= 2 x 8KB + 4 x 4KB
3) Thiết kế mạch giải mã địa chỉ bộ nhớ cho hệ thống Z80-CPU: 1ROM 4K, 1RAM 4K và 2RAM 2K. Yêu cầu địa chỉ RAM liên tục từ 1800H trở đi.

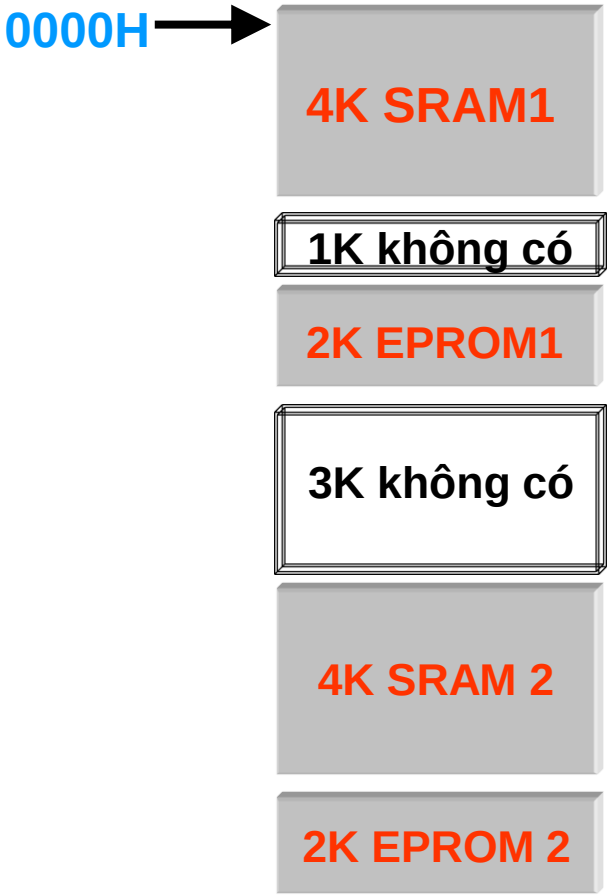
Bài tập (2)

4) Hãy xác định tầm địa chỉ làm cho các tín hiệu CS1, CS2, CS3 trong mạch giải mã địa chỉ sau đây tích cực.



Bài tập (3)

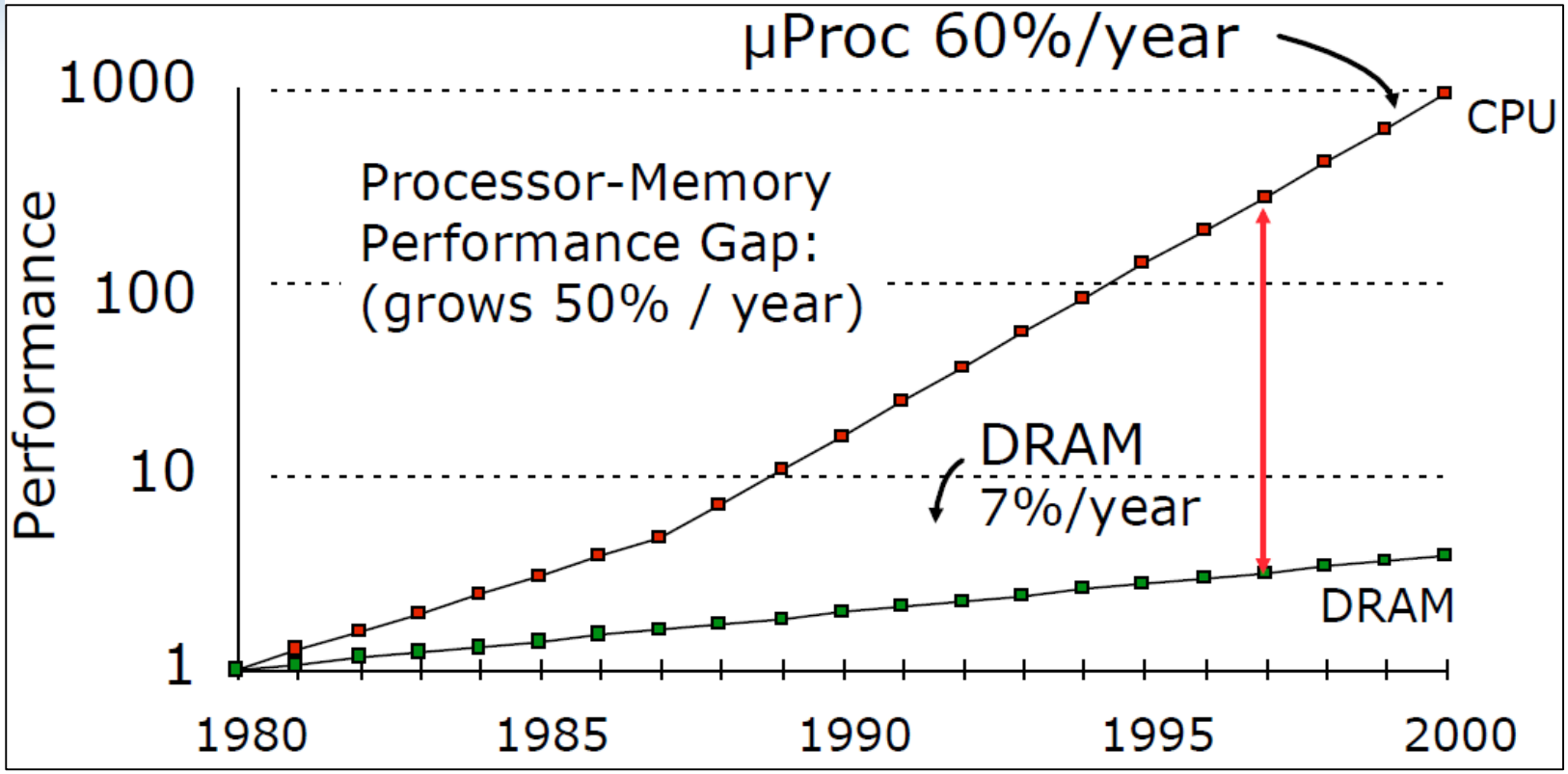
5. Một hệ thống máy tính dùng CPU có tuyến địa chỉ 16 đường $A_{15} – A_0$ và có bản đồ sử dụng bộ nhớ như sau:
   a) Hãy xác định tầm địa chỉ sử dụng của từng chip bộ nhớ trong hệ thống.
   b) Thiết kế mạch giải mã địa chỉ bộ nhớ đầy đủ cho CPU trên. Tín hiệu chọn bộ nhớ có tên là MREQ.



Nội dung trình bày

- Công nghệ và thuật ngữ liên quan đến bộ nhớ
- Tổ chức / thiết kế bộ nhớ
- Sự cần thiết phải có bộ nhớ đệm
- Phân loại bộ nhớ đệm
- Đánh giá hiệu năng của bộ nhớ đệm

Khoảng cách về hiệu năng của CPU-DRAM



- Bộ xử lý MIPS chạy với tốc độ 2GHz có thể thực thi 100 lệnh trong thời gian truy xuất dữ liệu ở bộ nhớ chính DRAM có thời gian truy xuất 50ns

Quy luật cục bộ của tham khảo

- Chương trình có xu hướng sử dụng lại dữ liệu và lệnh đã sử dụng hoặc đã tham khảo gần đây
- Cục bộ về mặt thời gian
  - Nếu một phần tử (biến, hàm, đối tượng…) được tham khảo, sẽ được tham khảo lại ngay sau đó
  - VD: các lệnh trong vòng lặp được nạp lại sau mỗi lần lặp; hàm `printf` có thể được gọi và thực thi nhiều lần
- Cục bộ về mặt không gian
  - Một lệnh đang được thực thi, lệnh kế tiếp nhiều khả năng được thực thi
  - Một phần tử được tham khảo, các phần tử gần đó có khả truy xuất sau đó
  - VD: Các lệnh được thực thi tuần tự; duyệt các phần tử mảng liên tiếp nhau

VD tính cục bộ của tham khảo

❖ Xét đoạn chương trình:

```
sum  = 0;
for (i = 0; i<n; i++)
   sum += a[i];
return sum
```

❖ Cục bộ về mặt thời gian

✧ Biến `sum` được tham khảo ở mỗi lần lặp

✧ Lệnh `sum += a[i]` được lặp lại

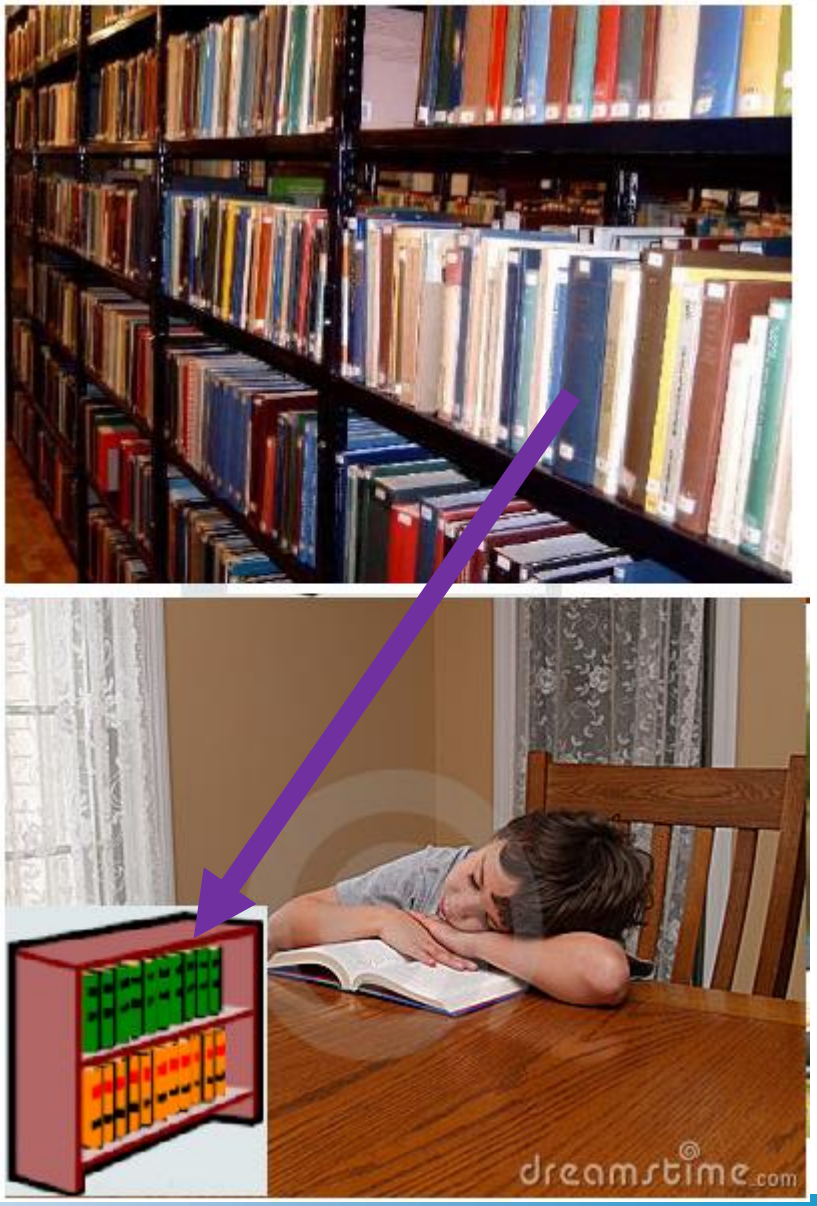
❖ Cục bộ về mặt không gian

✧ Các phần tử của mảng a[] lần lượt được tham khảo qua mỗi lần lặp

✧ Các lệnh liên tiếp nhau (dạng hợp ngữ của đoạn chương trình) sẽ được thực thi

VD tính cục bộ của tham khảo

- Sinh viên mượn sách tham khảo từ thư viện
- Cục bộ về mặt thời gian
  - Sách "Computer Architecture" được tham khảo lại hàng tuần trong học kỳ này
- Cục bộ về mặt không gian
  - Sách của những môn khác (vd: Cấu trúc dữ liệu – giải thuật) được giảng dạy trong kỳ này cũng được tham khảo



Sự cần thiết của bộ nhớ đệm

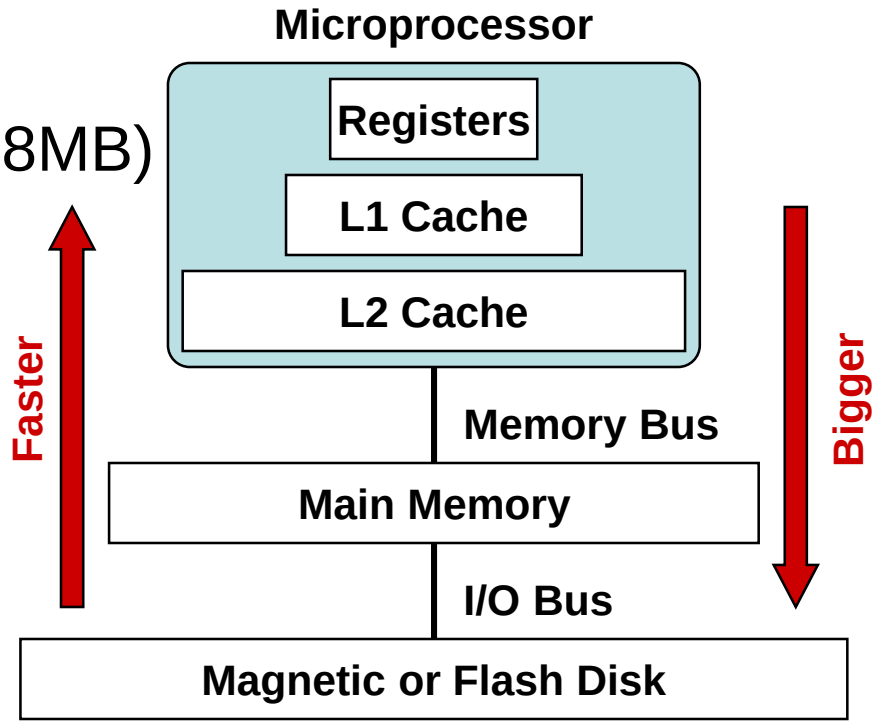
- Khoản cách tốc độ ngày càng xa giữa CPU - DRAM
- Mỗi lệnh cần ít nhất một truy xuất đến bộ nhớ
  - Một truy xuất để nạp lệnh
  - Truy xuất thứ hai có thể có cho lệnh load/store
- Băng thông bộ nhớ giới hạn tốc độ thực hiện lệnh
- Chi phí trên một phần tử nhớ của bộ nhớ tốc độ cao (SRAM, D Flip-Flop) lớn -> giá SRAM 1GHz ??
- Cần giải quyết bài toán bộ nhớ "**vừa nhanh, vừa rẻ**"
- Bài toán này được giải quyết bởi việc đệm dữ liệu (bộ nhớ đệm, tổ chức bộ nhớ phân cấp) dựa vào đặc tính "**cục bộ của tham khảo**" của một chương trình

Bộ nhớ đệm là gì ?

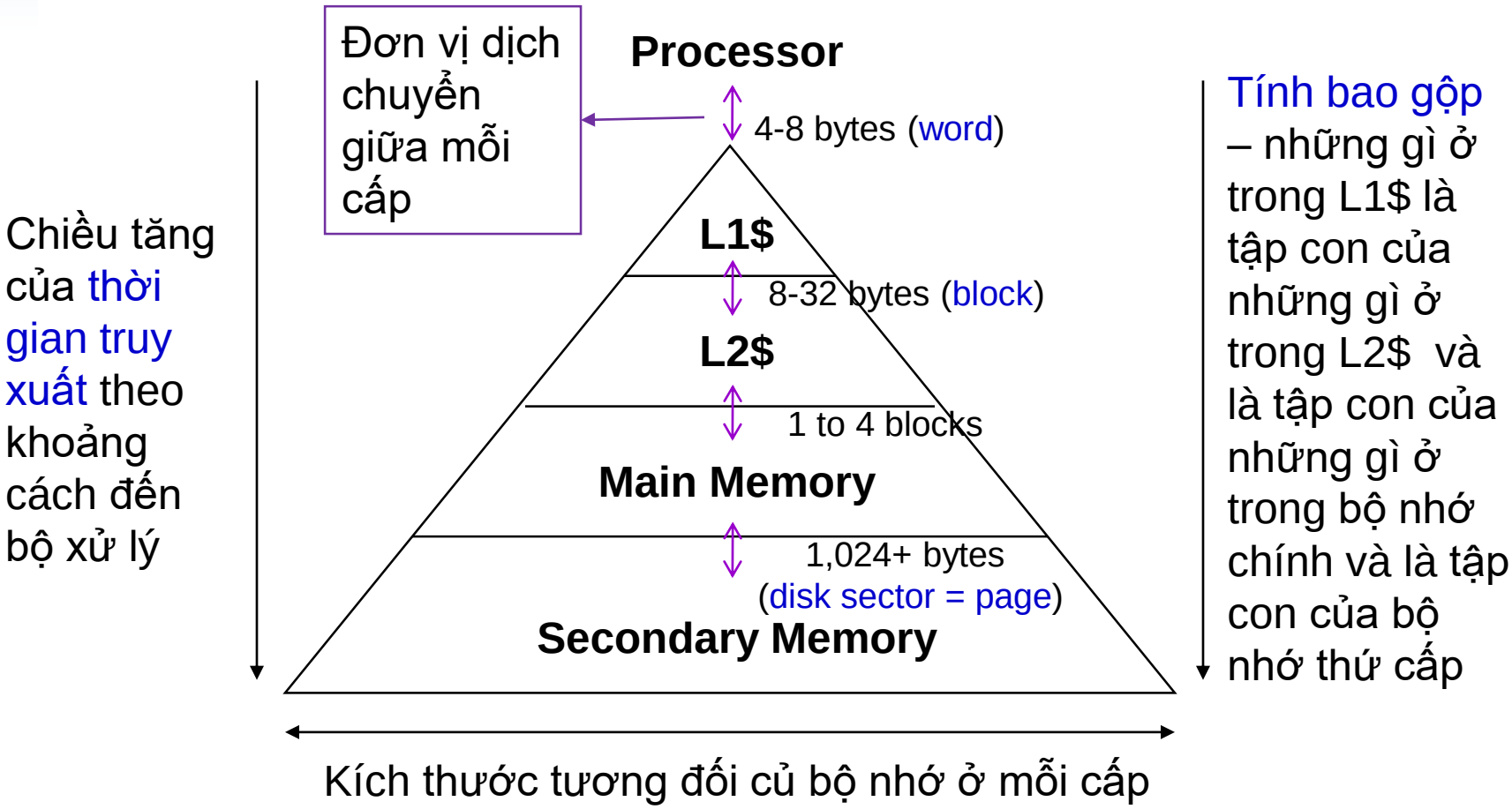
- Bộ nhớ nhanh (SRAM), dung lượng ít
  - Đệm một phần dữ liệu và lệnh của chương trình đang thực thi
- Được sử dụng để giảm thời gian truy xuất trung bình đến bộ nhớ chính
- Bộ nhớ đệm tận dụng tính cục bộ về thời gian bởi …
  - Giữ dữ liệu truy xuất gần đây gần với bộ xử lý
- Bộ nhớ đệm tận dụng tính cục bộ về không gian bởi …
  - Đưa cả những dữ liệu gần dữ liệu đang truy xuất vào bộ nhớ đệm
- Mục đích để đạt được
  - Tốc độ truy xuất nhanh của SRAM dùng trong bộ nhớ đệm
  - Cân bằng chi phí của hệ thống bộ nhớ

Tổ chức bộ nhớ phân cấp

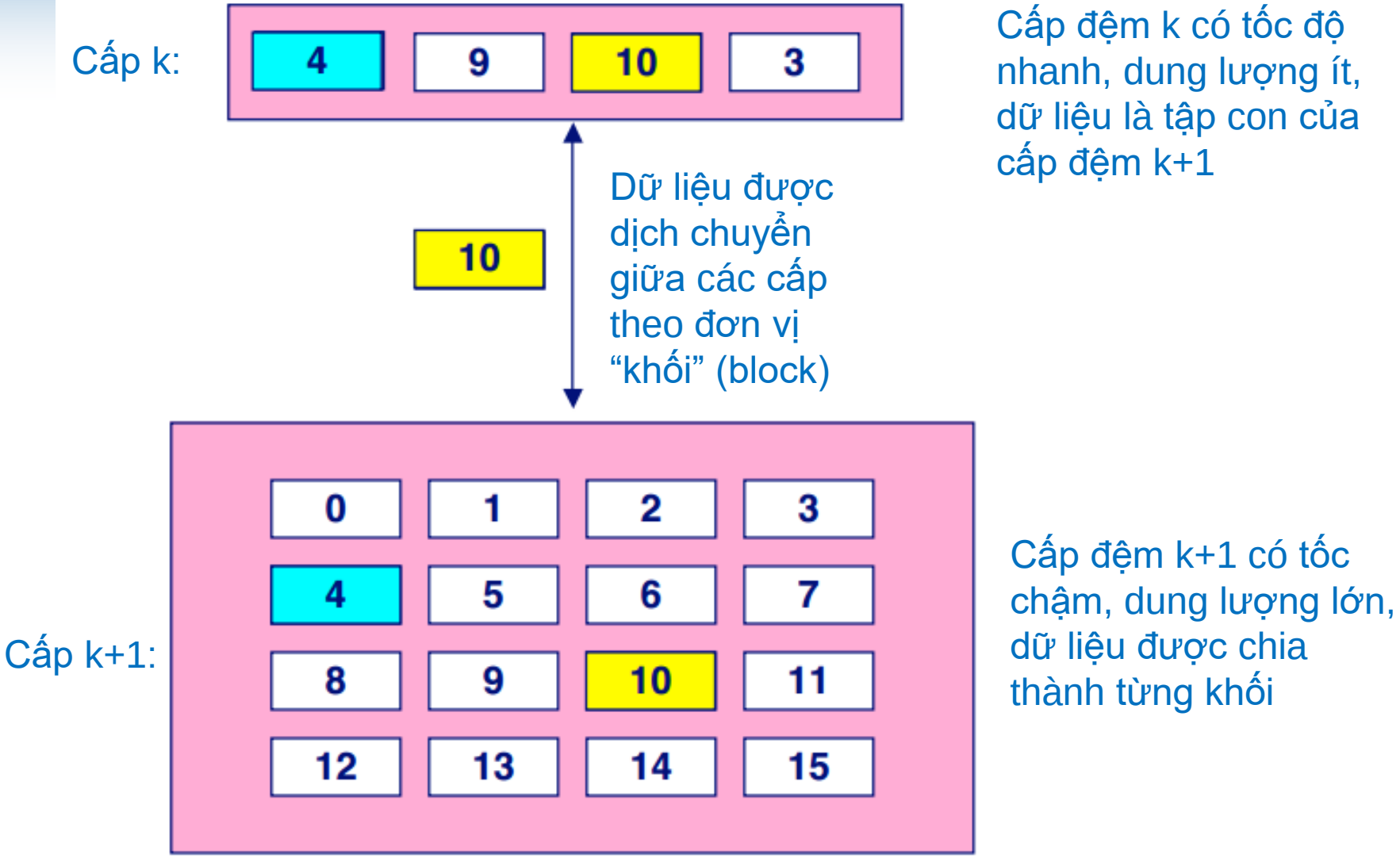
- Các thanh ghi ở trên cùng
  - Dung lượng thông thường < 1 KB
  - Thời gian truy xuất (access time) < 0.5 ns
- Bộ nhớ đệm cấp 1 L1 Cache (8 – 64 KB)
  - Access time: 1 ns
- Bộ nhớ đệm L2 (512KB – 8MB)
  - Access time: 3 – 10 ns
- Bộ nhớ chính (4 – 16 GB)
  - Access time: 50 – 100 ns
- Disk Storage (> 200 GB)
  - Access time: 5 – 10 ms



Đặc tính của bộ nhớ phân cấp



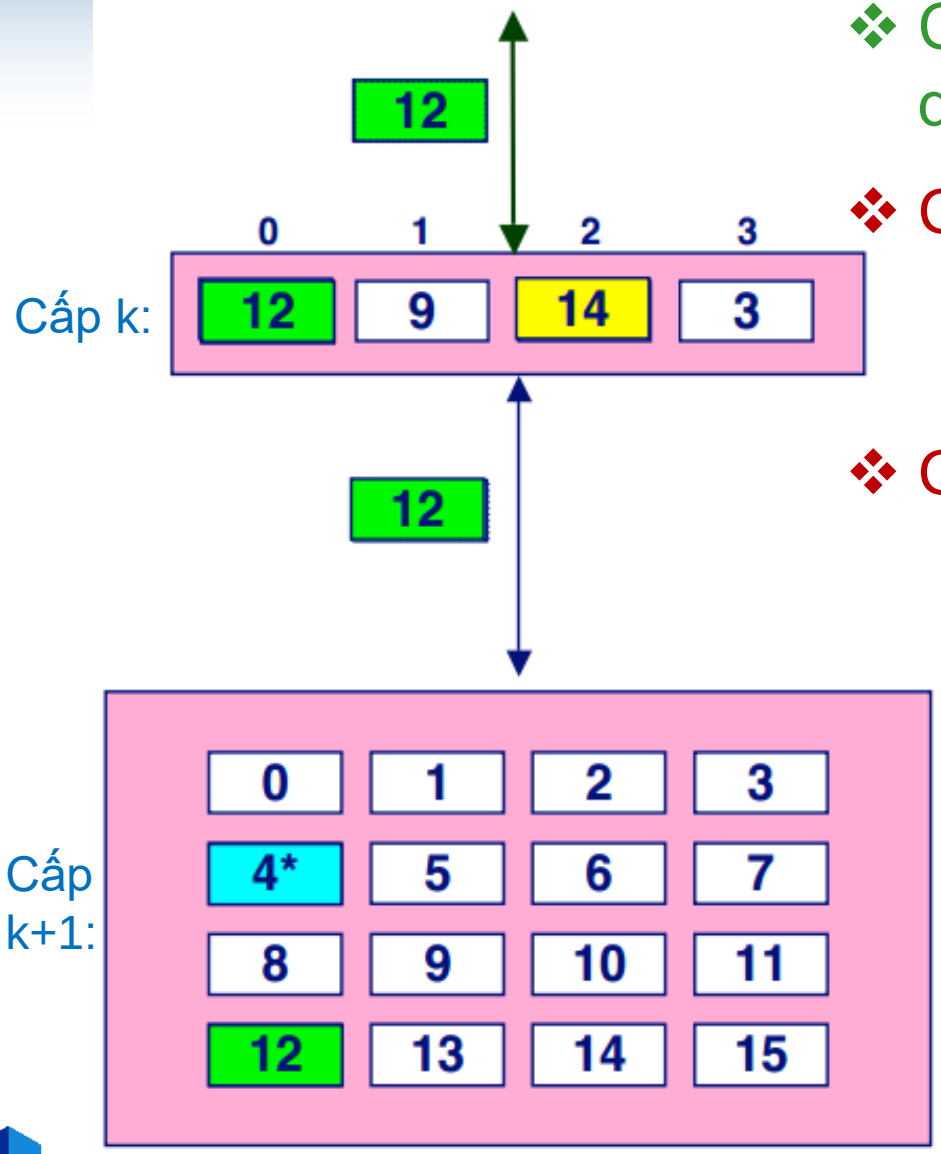
Đệm dữ liệu của bộ nhớ đệm



Cấp đệm k có tốc độ nhanh, dung lượng ít, dữ liệu là tập con của cấp đệm k+1

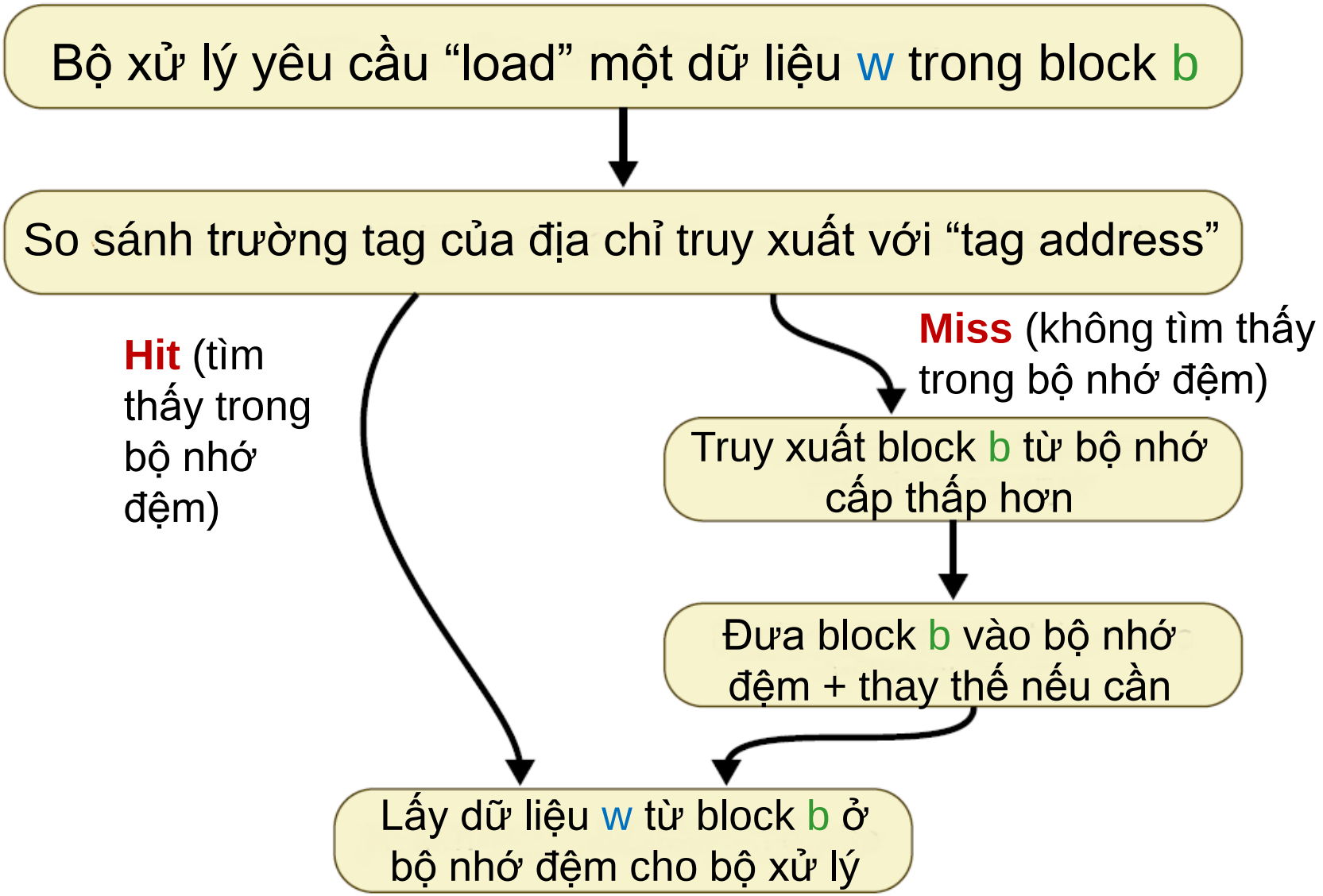
Cấp đệm k+1 có tốc chậm, dung lượng lớn, dữ liệu được chia thành từng khối

Các khác niệm về bộ nhớ đệm



- Chương trình đang tham khảo dữ liệu **w** ở trong block **b**
- Cache hit:
  - Chương trình tìm thấy b ở bộ nhớ đệm cấp k (vd: block 14)
- Cache miss:
  - b (vd: 12) không có ở cấp k, phải lấy từ cấp k+1, đồng thời được đưa vào cấp k
    - Cách ánh xạ? b sẽ được đưa vào vị trí nào (vd: 12 mod 4 = 0)
  - Nếu bộ nhớ đệm cấp k đầy, một block nào đó sẽ được thay thế:
    - Chiến lược thay thế? vd: thay thế bock đã được đưa vào sớm nhất

Hoạt động của bộ nhớ đệm



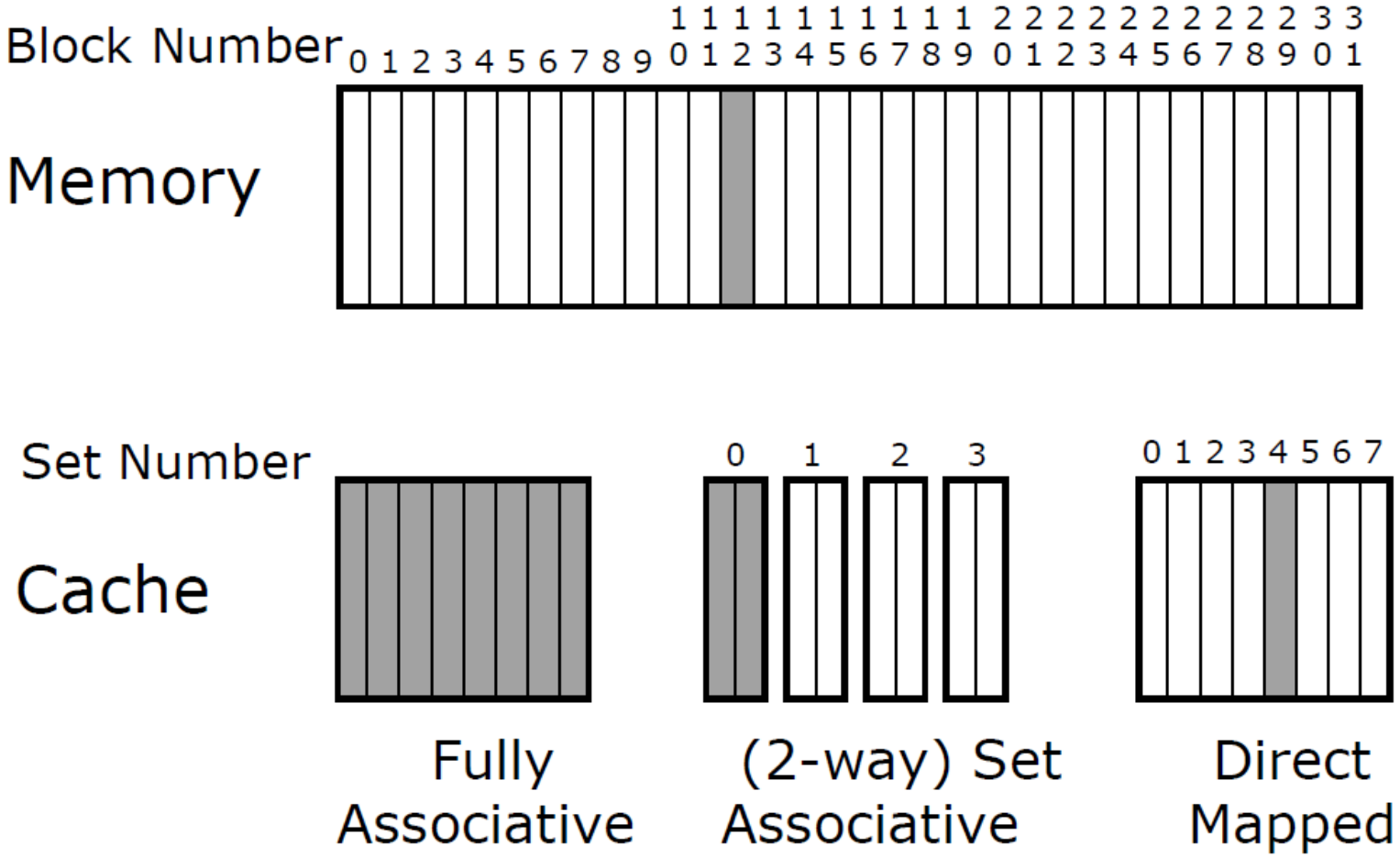
Nội dung trình bày

- Công nghệ và thuật ngữ liên quan đến bộ nhớ
- Tổ chức / thiết kế bộ nhớ
- Sự cần thiết phải có bộ nhớ đệm
- Phân loại bộ nhớ đệm
- Đánh giá hiệu năng của bộ nhớ đệm

Bốn câu hỏi liên quan tới bộ nhớ đệm

- Q1: Một khối sẽ được đưa vào đâu trong bộ nhớ đệm?
  - Direct Mapped, Set Associative, Fully Associative
- Q2: Làm sao để tìm một khối (xác định hit/miss)?
  - Block address, tag, index
- Q3: Khối nào sẽ được thay thế khi miss và bộ nhớ đệm đầy?
  - FIFO, Random, LRU
- Q4: Việc ghi như thế nào?
  - Write Back/Write Through (with Write Buffer)

Q1: Đưa một block vào bộ nhớ đệm



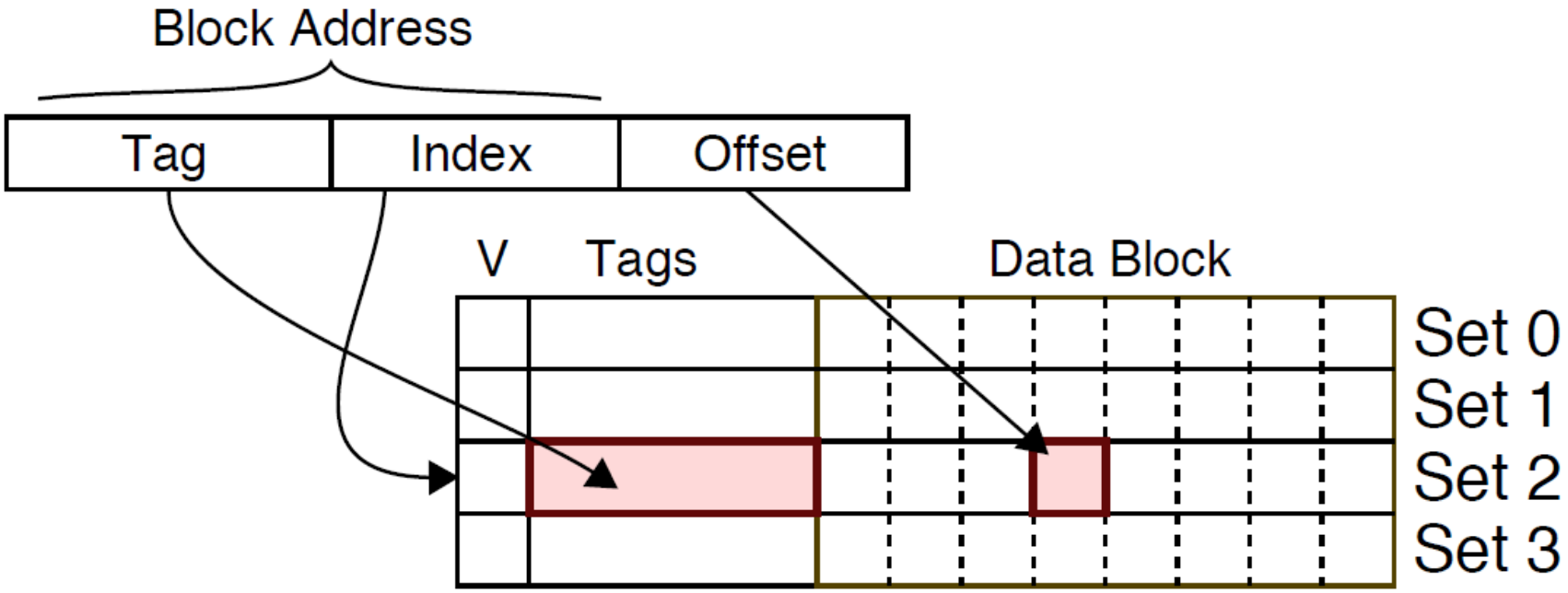
Block 12 được đưa vào vị trí nào?

Bất kỳ vị trí nào

Bất kỳ vị trí nào trong set 0 (12 mod 4)

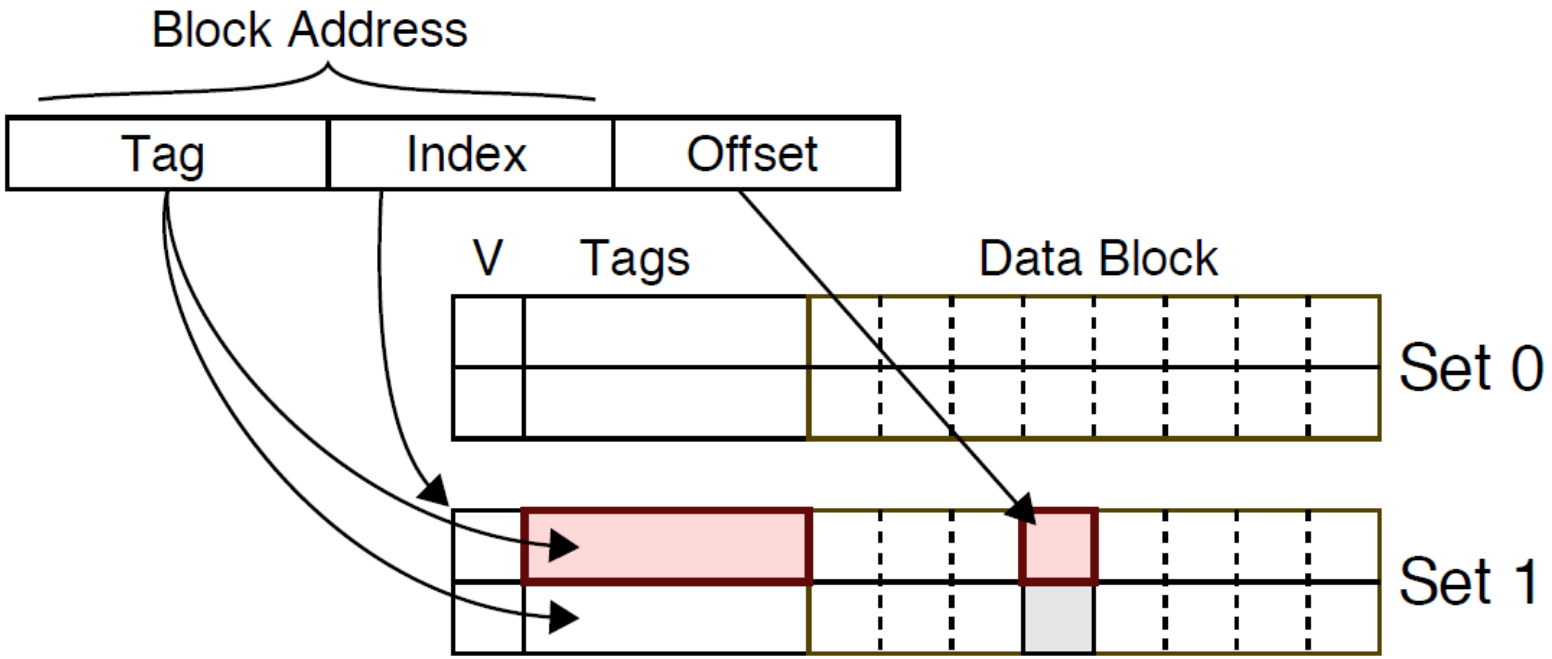
Tại vị trí 4 (12 mod 8)

Q2: Tìm một block trong bộ nhớ đệm



- Index: để tìm đến line (set) nào; so sánh Tag của địa chỉ truy xuất với giá trị ở trường tags (với Valid bit V là hợp lệ)
- Offset: để xác định dữ liệu w cần lấy trong block b
- Hình trên: Direct mapped, 8 Byte/block, 4 line cache
  - Offset: 3 bit; Index: 2 bit; Tag: 32 – (3 + 2) = 27 bit

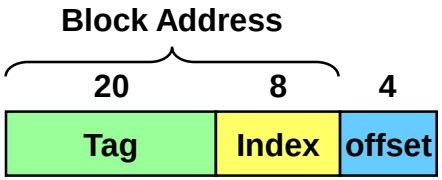
Q2: Tìm một block trong bộ nhớ đệm



- Index: để tìm đến set nào; so sánh Tag ở địa chỉ truy xuất với tất cả giá trị ở trường tags trong set đã tìm đến (với Valid bit V là hợp lệ)
- Hình trên: 2-way associative, 8 Byte/block, 4 line cache
  - Offset: 3 bit; Index: 1 bit; Tag: 32 – (3 + 1) = 28 bit

Ánh xạ một địa chỉ vào một block ở bộ nhớ đệm

- Ví dụ:
  - Xét một bộ nhớ đệm direct-mapped có 256 block (line)
  - Block size = 16 bytes
  - Tính giá trị tag, index, byte offset của địa chỉ: 0x01FFF8AC
- Lời giải
  - 32-bit địa chỉ được chia:

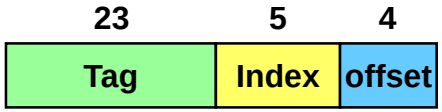


    - 4-bit byte offset, vì block size = 2^4 = 16 bytes
    - 8-bit cache index, vì bộ nhớ đệm có 2^8 = 256 sets (1 set gồm 1 line/block)
    - 20-bit tag
  - Byte offset = 0xC = 12 (4 bit thấp giá trị địa chỉ)
  - Cache index = 0x8A = 138 (8 bit thấp tiếp theo của giá trị địa chỉ)
  - Tag = 0x01FFF (20 bit cao của giá trị địa chỉ)

Ví dụ Cache Hits & Misses

- Xét một bộ nhớ đệm direct-mapped có 32 block
  - Bộ nhớ đệm ban đầu rỗng, Block size = 16 bytes
  - Các địa chỉ sau (dạng thập phân) được tham khảo:

    1000, 1004, 1008, 2548, 2552, 2556.
  - Ánh xạ những địa chỉ đó vào block trong bộ nhớ đệm và xác định hit/miss



- Lời giải:
  - 1000 = 0x3E8 cache index = 0x1E Miss (truy xuất lần đầu)
  - 1004 = 0x3EC cache index = 0x1E Hit
  - 1008 = 0x3F0 cache index = 0x1F Miss (V = không hợp lệ)
  - 2548 = 0x9F4 cache index = 0x1F Miss (khác tag)
  - 2552 = 0x9F8 cache index = 0x1F Hit
  - 2556 = 0x9FC cache index = 0x1F Hit

Ví dụ Cache Hits & Misses

❖ Direct-mapped cache, 2 line, block size = 2 word

Start with an empty cache - all blocks initially marked as not valid

0 1 2 3 4 3 4 15

0 miss

00	Mem(1)	Mem(0)

1 hit

00	Mem(1)	Mem(0)

2 miss

00	Mem(1)	Mem(0)
00	Mem(3)	Mem(2)

3 hit

00	Mem(1)	Mem(0)
00	Mem(3)	Mem(2)

4 miss

~~00~~ 01	~~Mem(1)~~ 5	~~Mem(0)~~ 4
00	Mem(3)	Mem(2)

3 hit

01	Mem(5)	Mem(4)
00	Mem(3)	Mem(2)

4 hit

01	Mem(5)	Mem(4)
00	Mem(3)	Mem(2)

15 miss

~~01~~ 11	~~Mem(5)~~ 15	~~Mem(4)~~ 14
00	Mem(3)	Mem(2)

Q3: Thay thế một block khi đầy + miss

- Không có lựa chọn trong bộ nhớ đệm "direct-mapped"
- Vớ bộ nhớ đệm "associative", block nào được thay thế khi đầy và miss?
- Ngẫu nhiên
- Least Recently Used (LRU)
  - Thay thế block không được tham khảo lâu nhất
  - Cập nhật thứ tự của block mỗi khi "hit"
- First In First Out (FIFO)
  - Thay thế block đưa vào trước ở một set
  - Một bộ đếm theo dõi được tăng mỗi khi "miss"

Q4: Chiến lược ghi

- Trường hợp "Cache Hit":
  - **Write Through** – ghi dữ liệu vào cả bộ nhớ đệm và bộ nhớ chính
  - **Write Back** – chỉ ghi dữ liệu vào bộ nhớ đệm, ghi dữ liệu vào bộ nhớ chính khi "miss"
- Trường hợp "Cache Miss":
  - **Write Allocate** – nạp block chứa dữ liệu cần ghi vào bộ nhớ đệm, sau đó thực hiện ghi
  - **No Write Allocate** – không thay đổi bộ nhớ đệm, chỉ ghi xuống bộ nhớ chính
- Các kết hợp thông dụng:
  - Write Through & No Write Allocate
  - Write Back & Write Allocate

Nội dung trình bày

- Công nghệ và thuật ngữ liên quan đến bộ nhớ
- Tổ chức / thiết kế bộ nhớ
- Sự cần thiết phải có bộ nhớ đệm
- Phân loại bộ nhớ đệm
- Đánh giá hiệu năng của bộ nhớ đệm

Hit Rate & Miss Rate

- Hit Rate = Hits / (Hits + Misses)
- Miss Rate = Misses / (Hits + Misses)
- I-Cache Miss Rate = Miss rate tại bộ nhớ đệm lệnh
- D-Cache Miss Rate = Miss rate tại bộ nhớ đệm dữ liệu
- Ví dụ:
  - 1000 lệnh được nạp, 150 lần miss ở I-Cache
  - 25% là lệnh load-store, 50 lần miss ở D-Cache
  - Tính I-cache và D-cache miss rate?
- I-Cache Miss Rate = 150 / 1000 = 15%
- D-Cache Miss Rate = 50 / (25% × 1000) = 50 / 250 = 20%

Memory Stall Cycles

- Bộ xử lý stall trong trường hợp Cache miss
  - Khi nạp lệnh từ Instruction Cache (I-cache)
  - Khi load/store dữ liệu tại Data Cache (D-cache)

Memory stall cycles = Combined Misses × Miss Penalty

- Miss Penalty: số chu kỳ xung nhịp cần để xử lý "miss"

Combined Misses = I-Cache Misses + D-Cache Misses

I-Cache Misses = I-Count × I-Cache Miss Rate

D-Cache Misses = LS-Count × D-Cache Miss Rate

LS-Count (Load & Store) = I-Count × LS Frequency

Memory Stall Cycles Per Instruction

- Memory Stall Cycles Per Instruction =

  Combined Misses Per Instruction × Miss Penalty

- Miss Penalty được giả sử bằng nhau cho I-cache & D-cache
- Miss Penalty được giả sử bằng nhau cho Load & Store
- Combined Misses Per Instruction =

  I-Cache Miss Rate + LS Frequency × D-Cache Miss Rate

- Do đó, Memory Stall Cycles Per Instruction =

  I-Cache Miss Rate × Miss Penalty +

  LS Frequency × D-Cache Miss Rate × Miss Penalty

Ví dụ về Memory Stall Cycles

- Xét một chương trình có các thông số sau:
  - Instruction count (I-Count) = 10^6 lệnh
  - 30% là lệnh load & store
  - D-cache miss rate là 5%; I-cache miss rate là 1%
  - Miss penalty: 100 clock cycles (chu kỳ xung nhịp
  - Tính combined misses per instruction và memory stall cycles
- Combined misses per instruction in I-Cache and D-Cache
  - 1% + 30% × 5% = 0.025 combined misses per instruction
  - 25 lần miss trên 1000 lệnh
- Memory stall cycles
  - 0.025 × 100 (miss penalty) = 2.5 stall cycles per instruction
  - Total memory stall cycles = 10^6 × 2.5 = 2,500,000

CPU Time với Memory Stall Cycles

$$\text{CPU Time} = \text{I-Count} \times \text{CPI}_{\text{MemoryStalls}} \times \text{Clock Cycle}$$

$$\text{CPI}_{\text{MemoryStalls}} = \text{CPI}_{\text{PerfectCache}} + \text{Mem Stalls per Instruction}$$

- $\text{CPI}_{\text{PerfectCache}}$ = CPI khi bộ nhớ đệm là lý tưởng (không bị miss)
- $\text{CPI}_{\text{MemoryStalls}}$ = CPI khi tính đến memory stalls
- Memory stall cycles làm tăng CPI tổng thể

Ví dụ CPI với Memory Stalls

- Một bộ xử lý có CPI là 1.5 cho trường hợp không miss
  - Cache miss rate là 2% cho I-cache và 5% cho D-cache (load & store)
  - 20% lệnh là load & store
  - Cache miss penalty: 100 clock cycles cho cả I-cache & D-cache
- Tính CPI khi tính tới cache miss?
- Lời giải:

Mem Stalls per Instruction = $\overbrace{0.02\times100}^{\text{Instruction}} + \overbrace{0.2\times0.05\times100}^{\text{data}} = 3$

$CPI_{MemoryStalls} = 1.5 + 3 = 4.5$ cycles per instruction

$CPI_{MemoryStalls} / CPI_{PerfectCache} = 4.5 / 1.5 = 3$

Bộ xử lý chạy chậm đi 3 lần vì memory stall cycles

$CPI_{NoCache} = 1.5 + (1 + 0.2) \times 100 = 121.5$ (rất lớn)

Average Memory Access Time

- Average Memory Access Time (AMAT)

  AMAT = Hit time + Miss rate × Miss penalty

- Thời gian truy xuất trung bình tính cả cho trường hợp hit & miss

- Ví dụ: Tính AMAT cho bộ nhớ đệm:
  - Thời gian truy xuất bộ nhớ đệm (Hit time) = 1 clock cycle = 2 ns
  - Miss penalty: 20 clock cycles
  - Miss rate: 0.05

- Lời giải:

  AMAT = 1 + 0.05 × 20 = 2 cycles = 4 ns

  Khi không dùng bộ nhớ đệm, AMAT sẽ bằng Miss penalty = 20 cycle

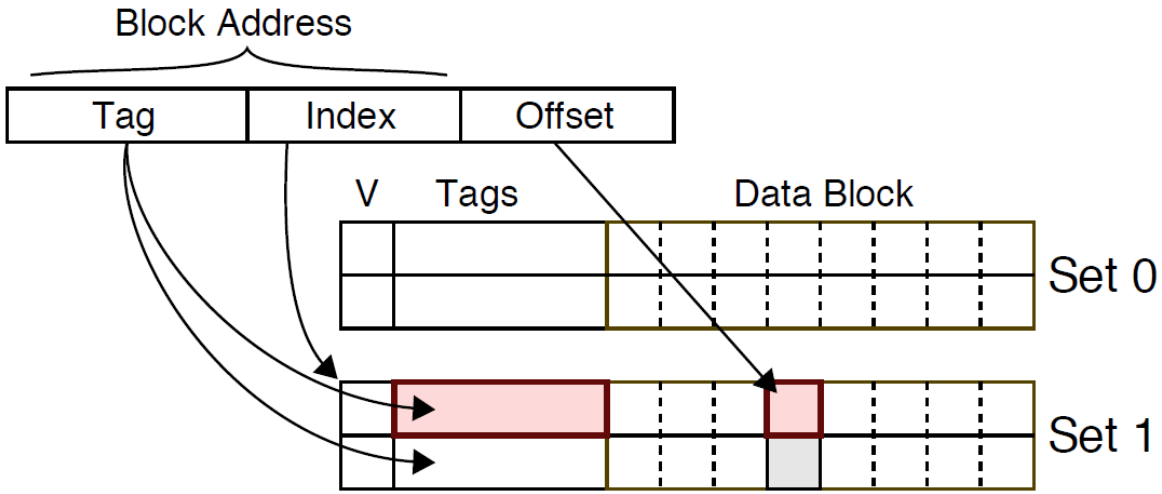
Cải tiến hiệu năng của bộ nhớ đệm

- Average Memory Access Time (AMAT)

  AMAT = Hit time + Miss rate * Miss penalty

- Công thức AMAT được sử dụng để thực hiện cải tiến
- Giảm Hit time
  - Bộ nhớ đệm phải nhỏ và đơn giản
- Giảm Miss Rate
  - Bộ nhớ đệm phải lớn, associativity nhiều, block size lớn
- Giảm Miss Penalty
  - Bộ nhớ đệm phải nhiều cấp

Phân loại Miss – 3C



- Compulsory (bắt buộc): Tham khảo đến một block chưa tồn tại trong bộ nhớ đệm (valid bit có giá trị không hợp lệ)
- Capacity (dung lượng): Dung lượng bộ nhớ đệm không đủ để chứa dữ liệu cần trong một chương trình (vd: chương trình cần 5 block để chứa dữ liệu > 4)
- Conflict (xung đột): Đụng độ do thiếu associativity (vd: chương trình lặp trên 3 block dữ liệu > 2)

Phân loại Miss

Compulsory misses độc lập với cache size

Rất nhỏ khi chạy chương trình dài

Capacity misses giảm khi tăng dung lượng bộ nhớ đệm

Conflict misses khi tăng tính associativity