Kiến trúc máy tính
(Computer Architecture)

Chương 4
Các cấp bộ nhớ

Mục đích, yêu cầu

- **Mục đích:** Giới thiệu chức năng, nguyên lý hoạt động của các loại bộ nhớ, tổ chức các cấp bộ nhớ và các kỹ thuật nâng cao hiệu quả hoạt động của bộ nhớ trong máy tính điện tử.
- **Yêu cầu:** Sinh viên hiểu được nguyên lý hoạt động của các loại bộ nhớ, tổ chức bộ nhớ phân cấp và các kỹ thuật nâng cao hiệu quả hoạt động của bộ nhớ trong máy tính điện tử: Bộ nhớ cache, bộ nhớ trong, bộ nhớ ảo…

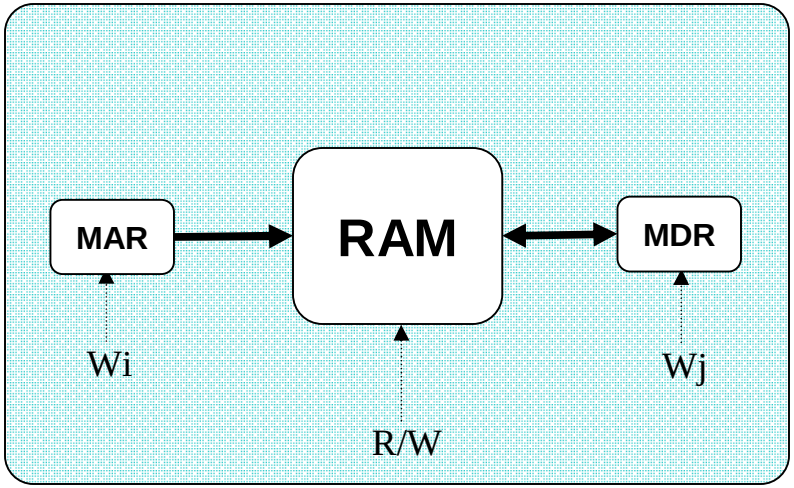
Chương 4

Các cấp bộ nhớ

1. Các loại bộ nhớ.
2. Các cấp bộ nhớ.
3. Xác xuất truy cập dữ liệu trong bộ nhớ.
4. Vận hành của cache.
5. Hiệu quả của cache.
6. Cache duy nhất hay cache riêng lẻ.
7. Các mức cache.
8. Bộ nhớ trong.
9. Bộ nhớ ảo.
10. Bảo vệ các tiến trình bằng bộ nhớ ảo.

1. Các loại bộ nhớ

- **Khái niệm:**
  - Bộ nhớ chứa chương trình (lệnh + dữ liệu):
  - Bộ nhớ là tập hợp các ô nhớ, mỗi ô nhớ có một địa chỉ, thông thường mỗi ô nhớ là 1 byte (8 bit) tuy nhiên có thể đọc hoặc viết một từ nhiều byte (2/4/8)
- **Các đặc trưng của bộ nhớ:**
  - Dung lượng (số ô nhớ)
  - Tổ chức (số bit cho 1 ô nhớ)
  - Thời gian thâm nhập (Latency: từ lúc đưa địa chỉ đến lúc đọc được nội dung)
  - Chu kỳ ô nhớ (Thời gian giữa hai lần liên tiếp thâm nhập bộ nhớ)

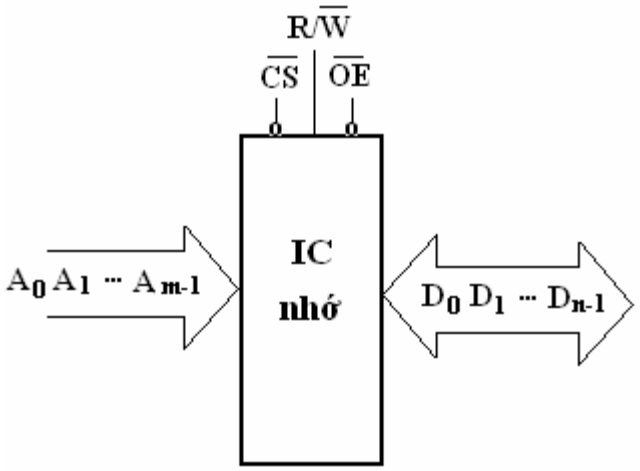


1. Các loại bộ nhớ

- **Các ngã vào ra của một chip nhớ**
  - **Ngã vào địa chỉ:** mỗi vị trí nhớ xác định một địa chỉ duy nhất. Một chip có n chân địa chỉ sẽ có 2^n vị trí nhớ. Ký hiệu các chân địa chỉ là A_0, A_1,...,A_n.
  - **Ngã vào/ra dữ liệu:** thường dữ liệu vào/ra chung một đường, nên các ngã này là ngã 3 trạng thái. Số chân địa chỉ và dữ liệu của một chip xác định dung lượng của chip đó. Ví dụ: IC có 10 chân địa chỉ và 8 chân dữ liệu có dung lượng nhớ là: 1024x8bit = 1KB.

1. Các loại bộ nhớ

- **Các ngã vào ra của một chip nhớ**
  - **Ngã vào điều khiển:** mỗi khi IC nhớ có yêu cầu xuất hoặc nhập dữ liệu thì các chân điều khiển tương ứng sẽ được tác động
  - **$\overline{\text{CS}}$:** Chip select, khi chân này xuống thấp chip nhớ được chọn.
  - **$\overline{\text{CE}}$:** Chip Enable, chức năng như chân **$\overline{\text{CS}}$**
  - **$\overline{\text{OE}}$:** Output Enable, cho phép xuất, dùng khi đọc dữ liệu.
  - **R/$\overline{\text{W}}$:** Read/Write, cho phép đọc dữ liệu khi ở mức cao và ghi dữ liệu khi ở mức thấp.
  - **$\overline{\text{CAS}}$, $\overline{\text{RAS}}$:** Column Address Strobe, Row address Strobe (chốt địa chỉ cột, chốt địa chỉ hàng). Chỉ những IC nhớ có điea chỉ hàng và cột mới có các chân này.



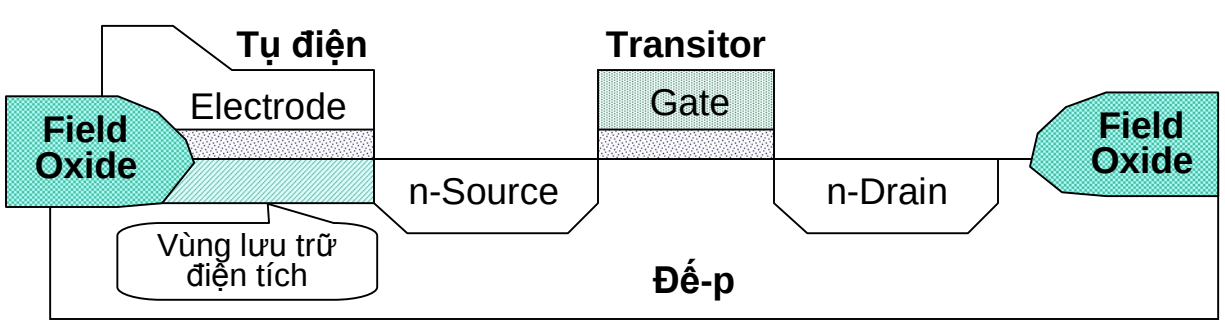
IC nhớ có m chân địa chỉ và n chân dữ liệu

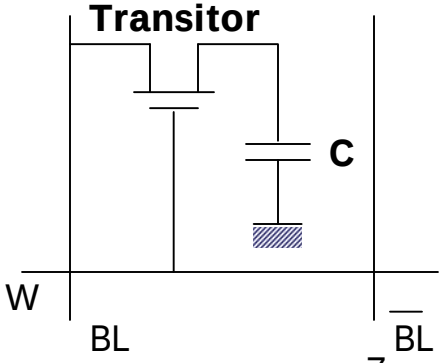
1. Các loại bộ nhớ

- **RAM (Random access Memory)**
  - **RAM động (DRAM: dynamic RAM)**
    - DRAM dùng kỹ thuật MOS.
    - Mỗi bít nhớ gồm có 1 transistor và 1 tụ điện.
    - Ghi nhớ dựa vào việc duy trì điện tích nạp vào tụ điện và như vậy việc đọc bít nhớ làm nội dung bít này bị hũy diệt.

    ⇨Chu kỳ bộ nhớ ít nhất là gấp đôi thời gian thâm nhập ô nhớ.

    ⇨Phải làm tươi bộ nhớ sau mỗi 2 μs (đọc ô nhớ và viết lại nội dung đó vào ô nhớ và làm như thế cho tất cả các ô nhớ).
    - Bộ nhớ DRAM chậm nhưng giá rẻ.

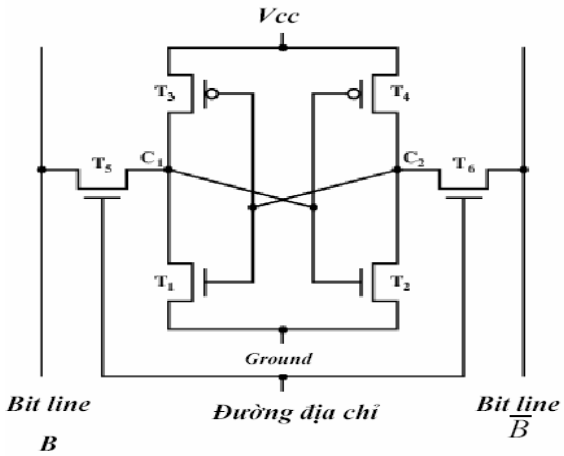


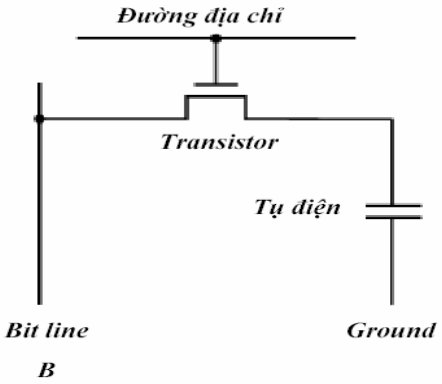


Các cấp bộ nhớ

1. Các loại bộ nhớ

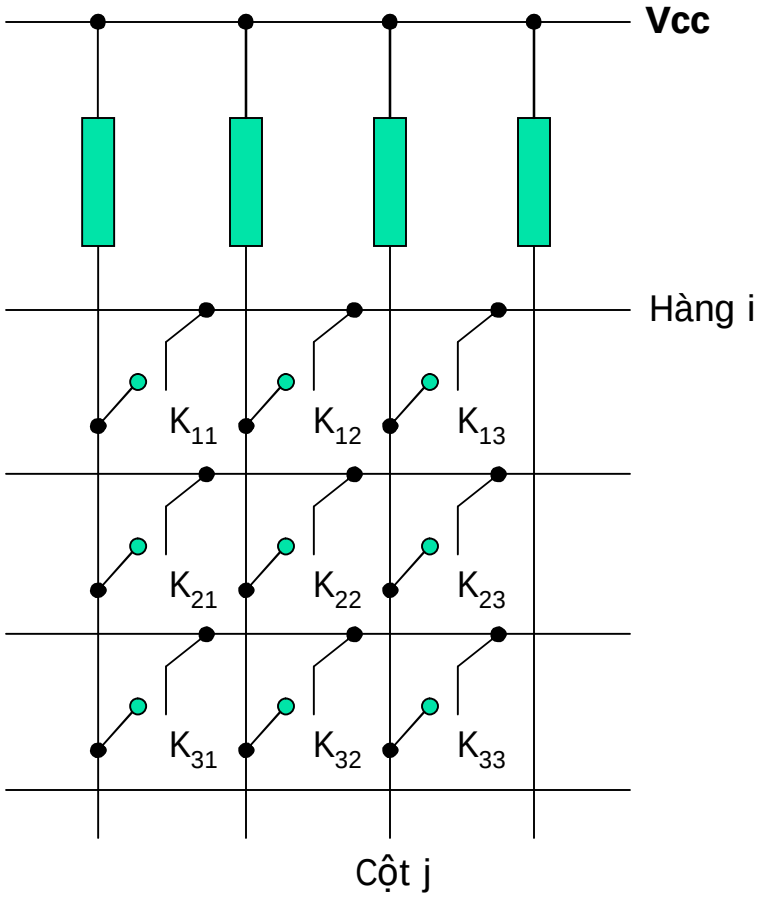
- **RAM (Random access Memory)**
  - **RAM tỉnh (SRAM: Static RAM)**
    - RAM tĩnh được chế tạo theo công nghệ CMOS và BiCMOS
    - Mỗi bít nhớ gồm có các cổng logic với độ 6 transistor MOS
    - Việc nhớ dữ liệu là không bị hủy diệt nếu được cung cấp điện
    - Chu kỳ bộ nhớ bằng thời gian thâm nhập.
    - SRAM là bộ nhớ nhanh nhưng giá thành cao.





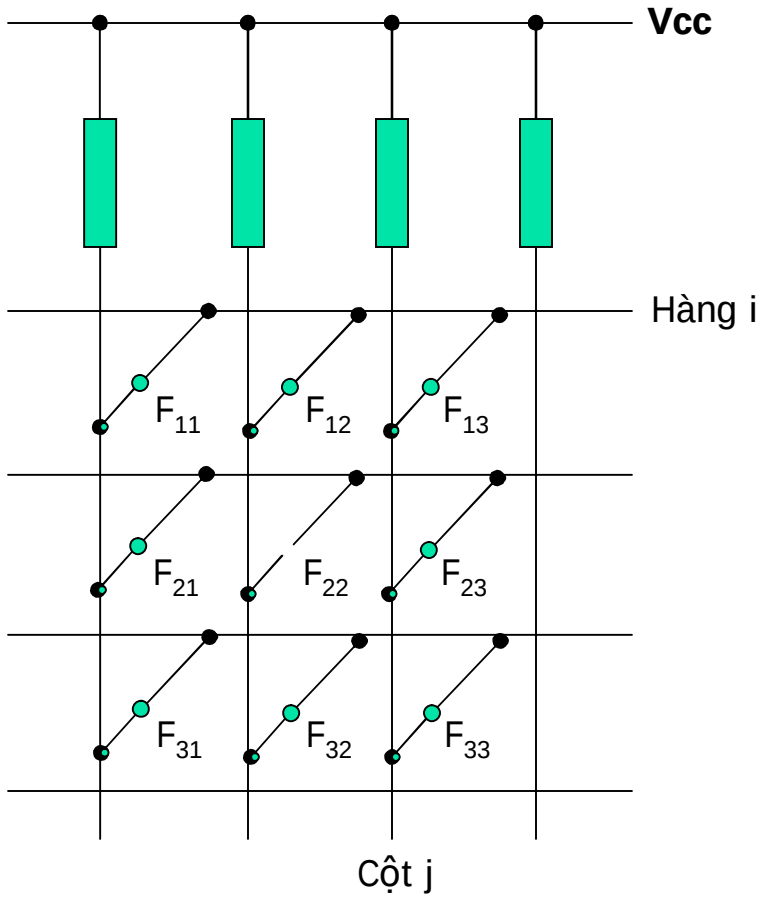
1. Các loại bộ nhớ

- **ROM (Read Only Memory):**
  Bộ nhớ ROM được chế tạo bằng công nghệ bán dẫn. Thông thường ROM chứa chương trình khởi động máy vi tính, chương trình điều khiển trong điều khiển tự động ...
  Chương trình / dữ liệu nằm trong ROM được viết lúc chế tạo.



1. Các loại bộ nhớ

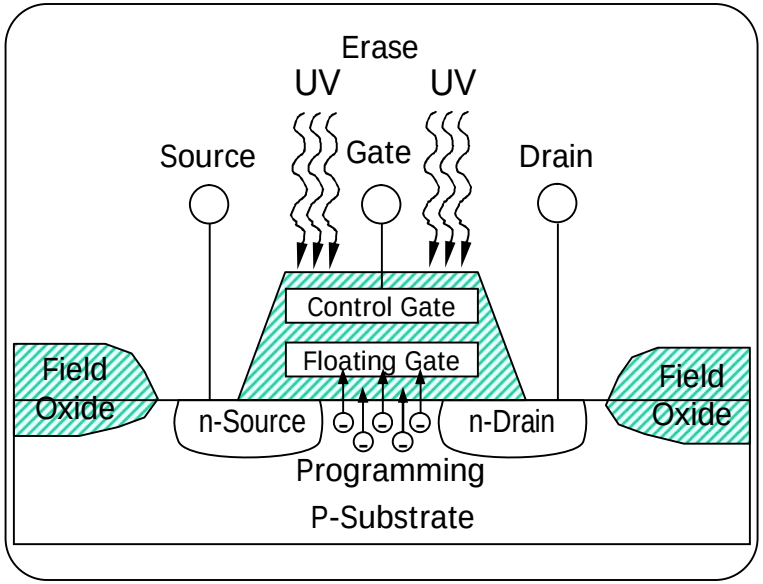
- **PROM** (Programable ROM):
  - Chế tao bằng các cầu chì (có thể làm đứt bằng điện).
  - Chương trình trong PROM có thể được viết vào bởi người sử dụng bằng thiết bị đặc biệt và không thể xóa được…

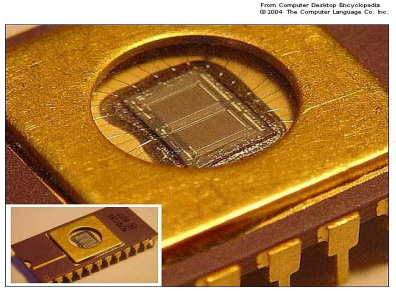


1. Các loại bộ nhớ

EPROM
(Erasable Programable ROM):

- Ngoài các thành phần cơ bản của một transitor FET (Field Effect Transitor) nó có thêm cực nổi (Floating Gate).
- Chương trình trong ROM có thể được viết vào bằng điện và có thể xóa bằng tia cực tím để viết lại bởi người sử dụng.

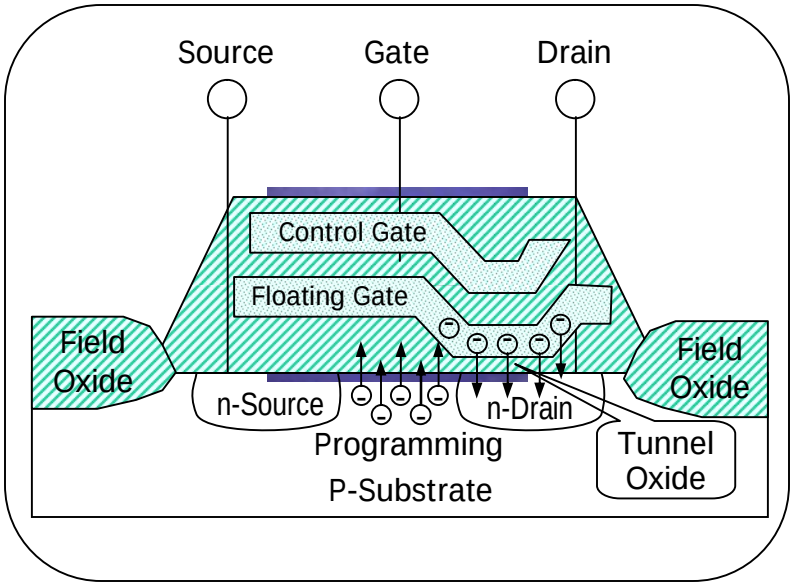




1. Các loại bộ nhớ

EEPROM (Eletrically Erasable Programable ROM)

- Chế tạo bằng chất bán dẫn.
- Nội dung nằm trong ROM có thể được viết và xóa bằng điện (để viết lại) bởi người sử dụng.

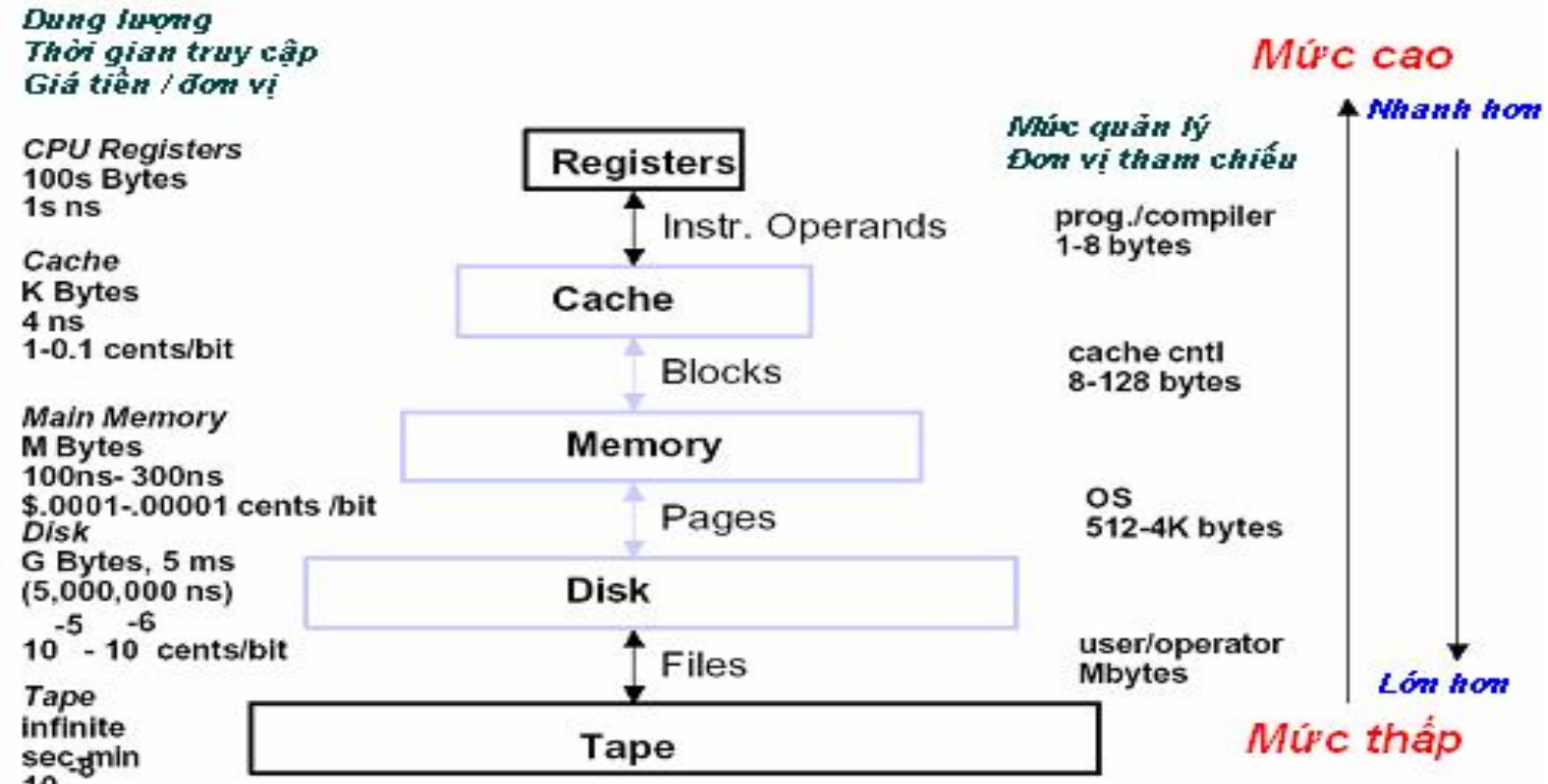


Các cấp bộ nhớ

1. Các loại bộ nhớ

- **Bộ nhớ ngoài:** Có thể lưu trữ dữ liệu lâu dài, bên ngoài máy tính.
  - **Đĩa từ**: Lưu trử thông tin bằng các vật liệu từ tính dưới dạng đĩa.
    - **Đĩa mềm** (Floppy Disk): Dùng để lưu trử tạm thời hoặc di chuyển số liệu giữa các máy tính.
    - **Đĩa ZIP** (Zip Disk): Như đĩa mềm nhưng dung lượng thông tin lớn nhờ mật độ lưu trữ cao và các kỹ thuật nén dữ liệu lưu trữ
    - **Đĩa cứng** (Hard Disk Drive): Gắn bên trong máy tính, dung lương lớn, tốc độ cao dùng để lưu trữ thông tin thường sử dụng và làm bộ nhớ ảo.
  - **Băng từ:** Lưu trữ thông tin dưới dạng các băng từ, dung lượng lớn, dùng để lưu trữ lâu dài và lưu trữ dự phòng cho đĩa cứng.
  - **Đĩa quang:** Lưu trữ thông tin dựa vào nguyên tắc phản xạ (land) và không phản xạ (pit) đối với ánh sáng (tia laser) trên bề mặt của đĩa (CDROM / DVDROM).
  - **Đĩa bán dẫn: DRAM / EEPROM**

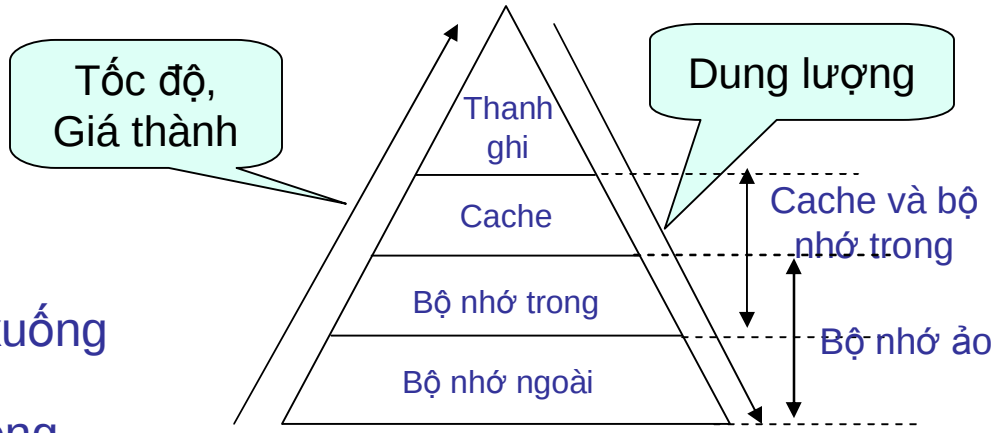
2. Các cấp bộ nhớ



Hình IV.2: Các cấp bộ nhớ

2. Các cấp bộ nhớ

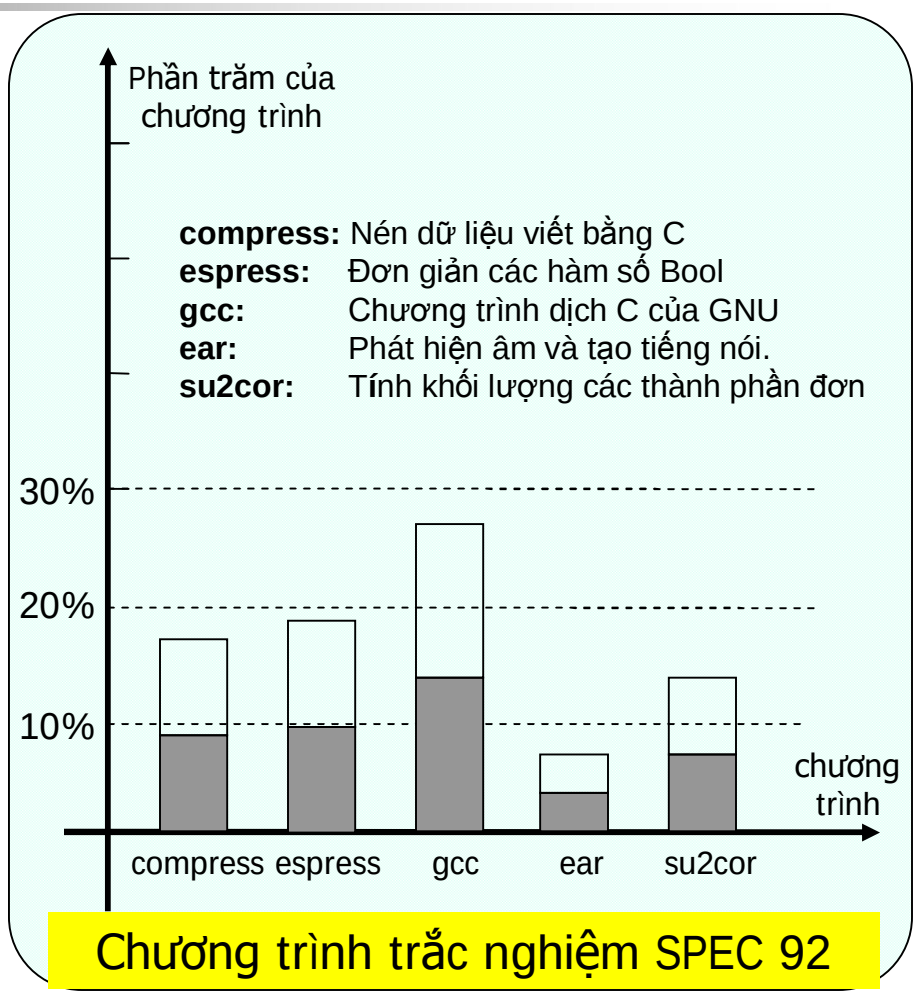
- **Mục đích:** Giúp người sử dụng có được bộ nhớ có dung lượng lớn, tốc độ nhanh, giá thành rẻ.
- **Các cấp bộ nhớ:**
  - Cấp thanh ghi
  - Cấp bộ nhớ cache
  - Cấp bộ nhớ trong
  - Cấp bộ nhớ ngoài
- **Cách phân cấp bộ nhớ:**
  - Dung lượng tăng từ trên xuống
  - Tốc độ giảm từ trên xuống
  - Giá thành giảm từ trên xuống



- **Nhận xét:**
  - Các cấp của bộ nhớ được lồng vào nhau. Dữ liệu trong một cấp được gặp lại trong cấp dưới và tiếp tục cho đến cấp thấp nhất.
  - Mỗi cấp có dung lượng lớn hơn cấp trên mình, ánh xạ một phần địa chỉ các ô nhớ của mình vào địa chỉ ô nhớ của cấp trên.
  - Các cấp bộ nhớ phải kiểm tra các địa chỉ. Sơ đồ bảo vệ với trách nhiệm kiểm tra địa chỉ là một phần của các cấp bộ nhớ.

3. Xác xuất truy cập dữ liệu bộ nhớ

Một chương trình mất 90% thời gian thi hành lệnh của nó để chỉ thi hành 10% số lệnh của chương trình.



3. Xác xuất truy cập dữ liệu bộ nhớ

- Cache là một bộ nhớ nhanh, dung lượng nhỏ thường là SRAM. Nó chứa lệnh và dữ liệu mà chương trình thường xuyên dùng đến.
- Tổ chức các cấp bộ nhớ sao cho các lệnh & các dữ liệu thường dùng nằm trong bộ nhớ cache, làm tăng hiệu quả của máy tính.
- **Ta có 2 nguyên tắc thâm nhập bộ nhớ của CPU:**
  - **Nguyên tắc về thời gian** cho biết các ô nhớ mà chúng ta vừa thâm nhập, có nhiều khả năng sẽ được thâm nhập trong tương lai gần.
    ⇨ Thâm nhập ô nhớ thì chuyển ô nhớ đó lên cache.
  - **Nguyên tắc về không gian** cho biết khi CPU thâm nhập một ô nhớ thì có nhiều khả năng nó thâm nhập ô nhớ có địa chỉ kế đó.
    ⇨ Chuyển một khối nhiều ô nhớ lân cận lên cache.

4. Vận hành của cache

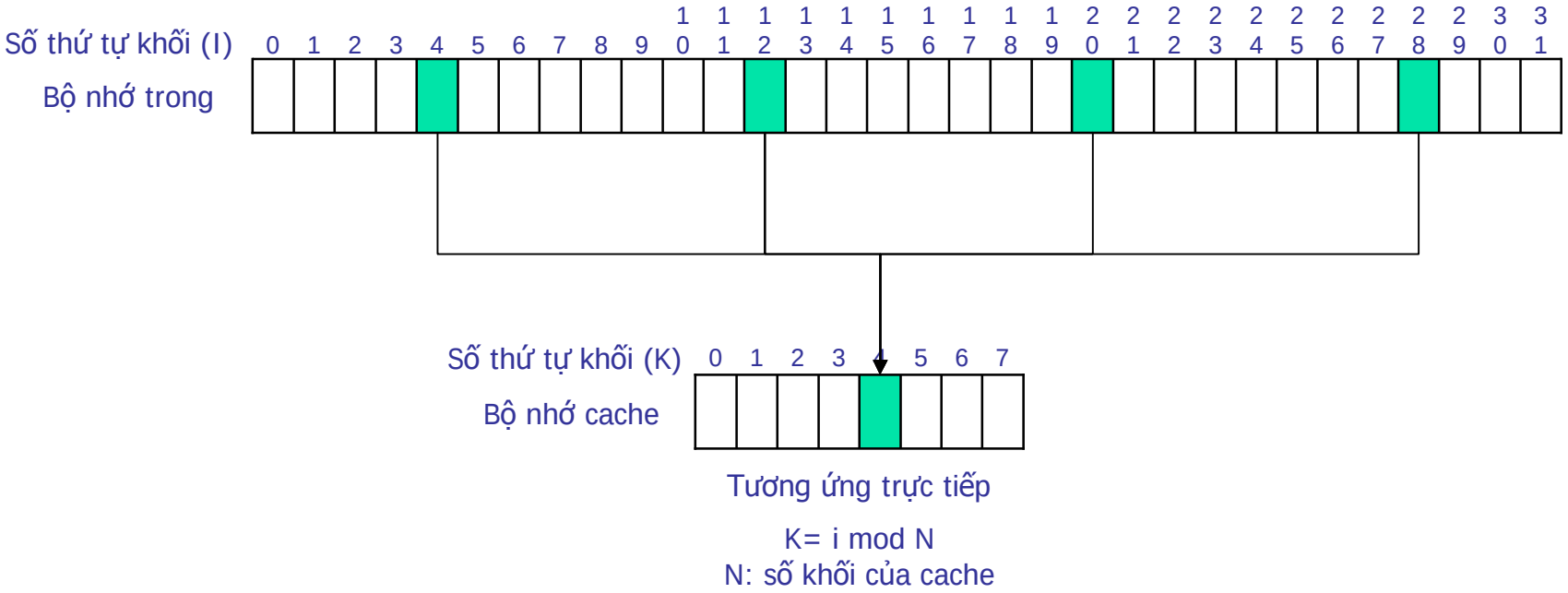
- **Đặc tính của cache:**
  - Cache vận hành trong suốt đối với bộ xử lý.
  - Cache chứa một phần của bộ nhớ trong.
  - Cache và bộ nhớ trong phải có cùng cấu trúc.
    - Cache và bộ nhớ được chia thành khối.
    - Kích thước khối rất quan trọng cho cache vận hành.
- **Đặc trưng của cache:**
  - **Thành công cache** (cache hit): Khi bộ xử lý tìm gặp phần tử mà mình cần trong cache.
  - **Thất bại cache** (cache miss): Khi bộ xử lý không tìm gặp phần tử mà mình cần trong cache.
    Trừng phạt cache (cache penalty): Thời gian cần để xử lý một thất bại cache. Trừng phạt cache bao gồm:
    - Thời gian thâm nhập bộ nhớ trong
    - Thời gian chuyển từ bộ nhớ trong đến cache.

4. Vận hành của cache

Các vấn đề của cache:

Q1: Phải để một khối vào chỗ nào trong cache?

Tương ứng trực tiếp: Mỗi khối của bộ nhớ trong tương ứng với một khối trong cache

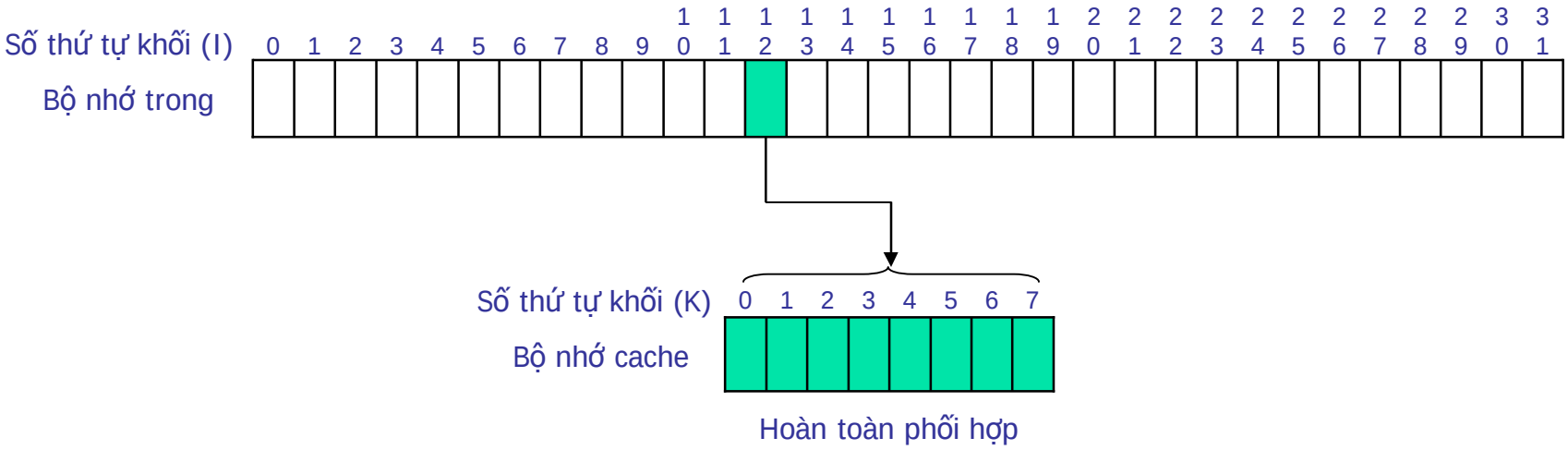


4. Vận hành của cache

Các vấn đề của cache:

Q1: Phải để một khối vào chỗ nào trong cache?

Hoàn toàn phối hợp: Một khối trong bộ nhớ trong có thể được đặt bất kỳ vị trí nào trong cache.

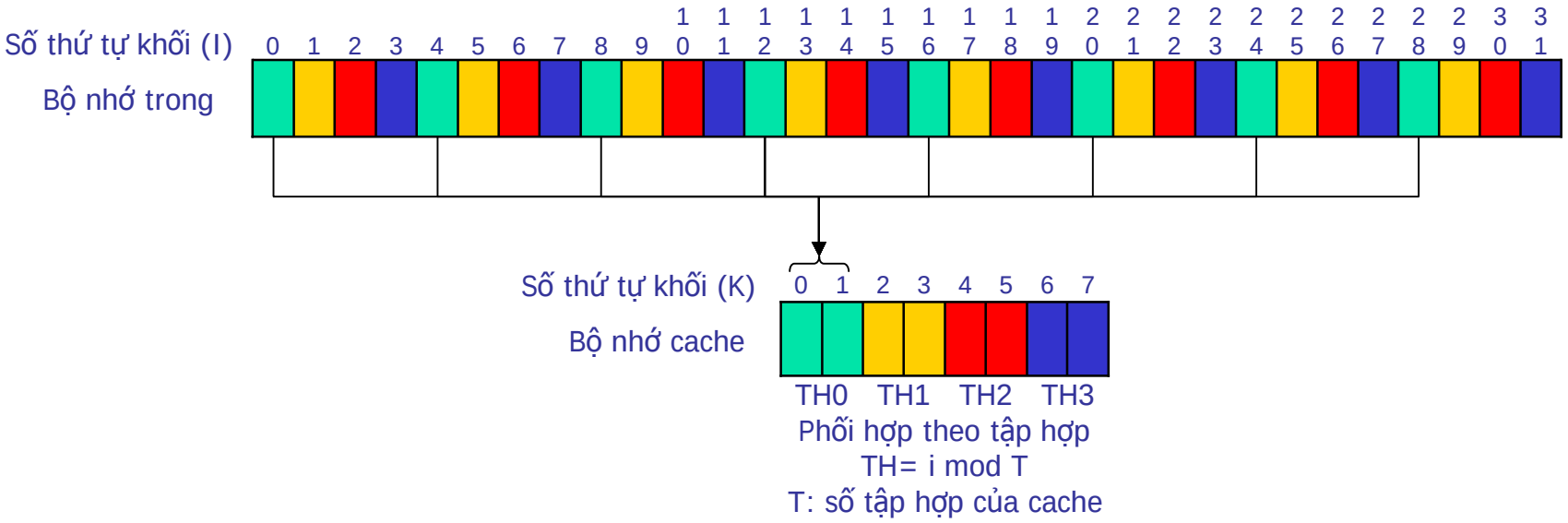


4. Vận hành của cache

Các vấn đề của cache:

Q1: Phải để một khối vào chỗ nào trong cache?

Phối hợp theo tập hợp: Một khối của bộ nhớ trong có thể để trong một số giới hạn các khối của cache



4. Vận hành của cache

Q2: Làm sao tìm một khối khi nó hiện diện trong cache?

- Mỗi khối của cache đều có một nhãn địa chỉ cho biết số thứ tự của các khối của bộ nhớ trong đang hiện diện trong cache.
- Bộ phận lưu trữ nhãn địa chỉ trong bộ nhớ cache gọi là TAG RAM

Ví dụ: Bộ nhớ có 65536 từ (16 bit), mỗi khối 16 từ (4 bit) ⇨ 4096 khối (12 bit)
Cache có 2048 từ (11 bit), mỗi khối 16 từ (4 bit) ⇨ 128 khối (7 bit)

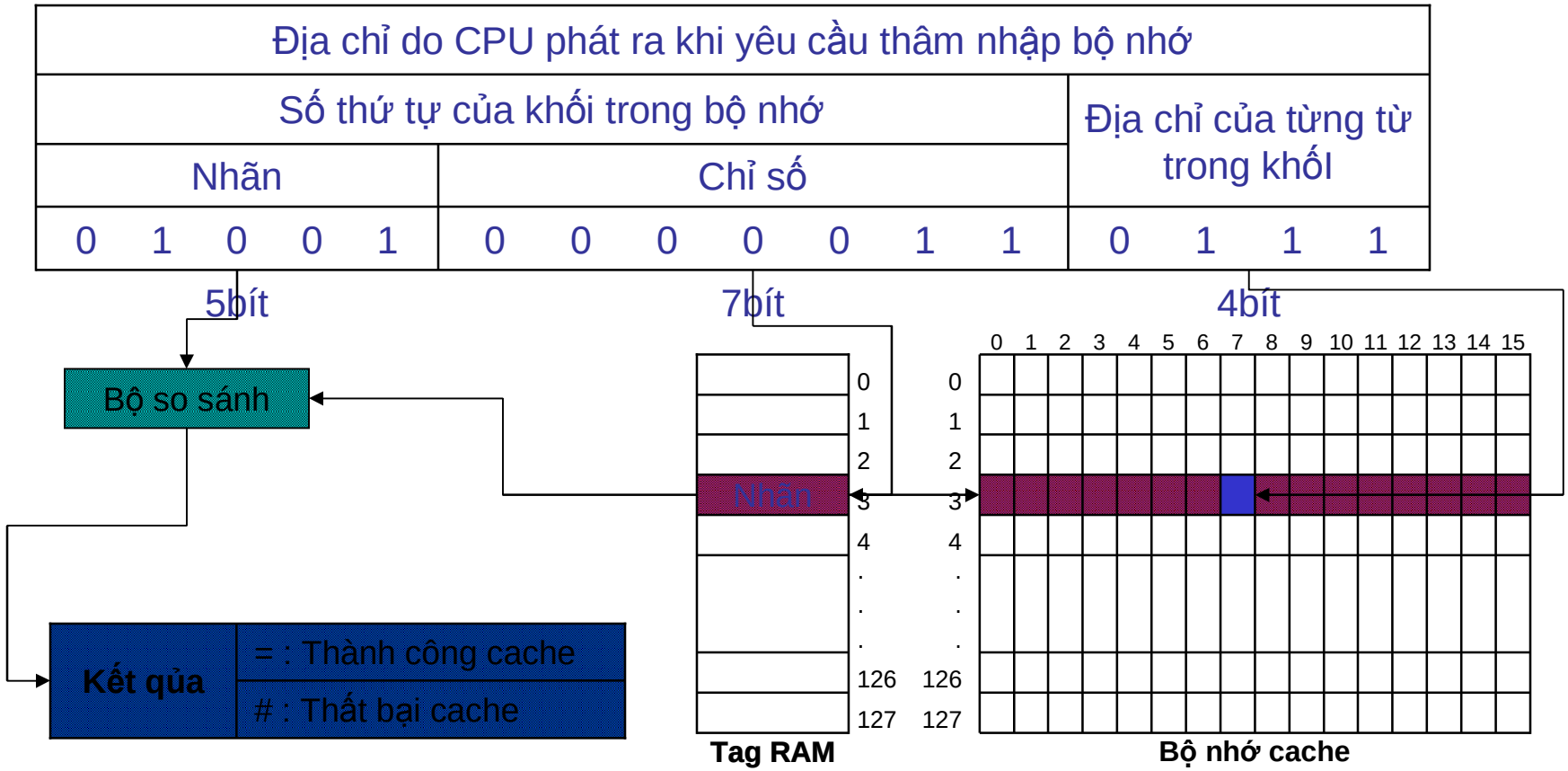
16 bit

Địa chỉ do CPU phát ra khi yêu cầu thâm nhập bộ nhớ		
Số thứ tự của khối trong bộ nhớ		Địa chỉ của từng từ trong khối
Nhãn	Chỉ số	Địa chỉ của từng từ trong khối
5bít	7bít	4bít

4. Vận hành của cache

Ví dụ: Địa chỉ bộ xử lý yêu cầu thâm nhập:18487 → 4837H → $(0100\ 1000\ 0011\ 0111)_2$

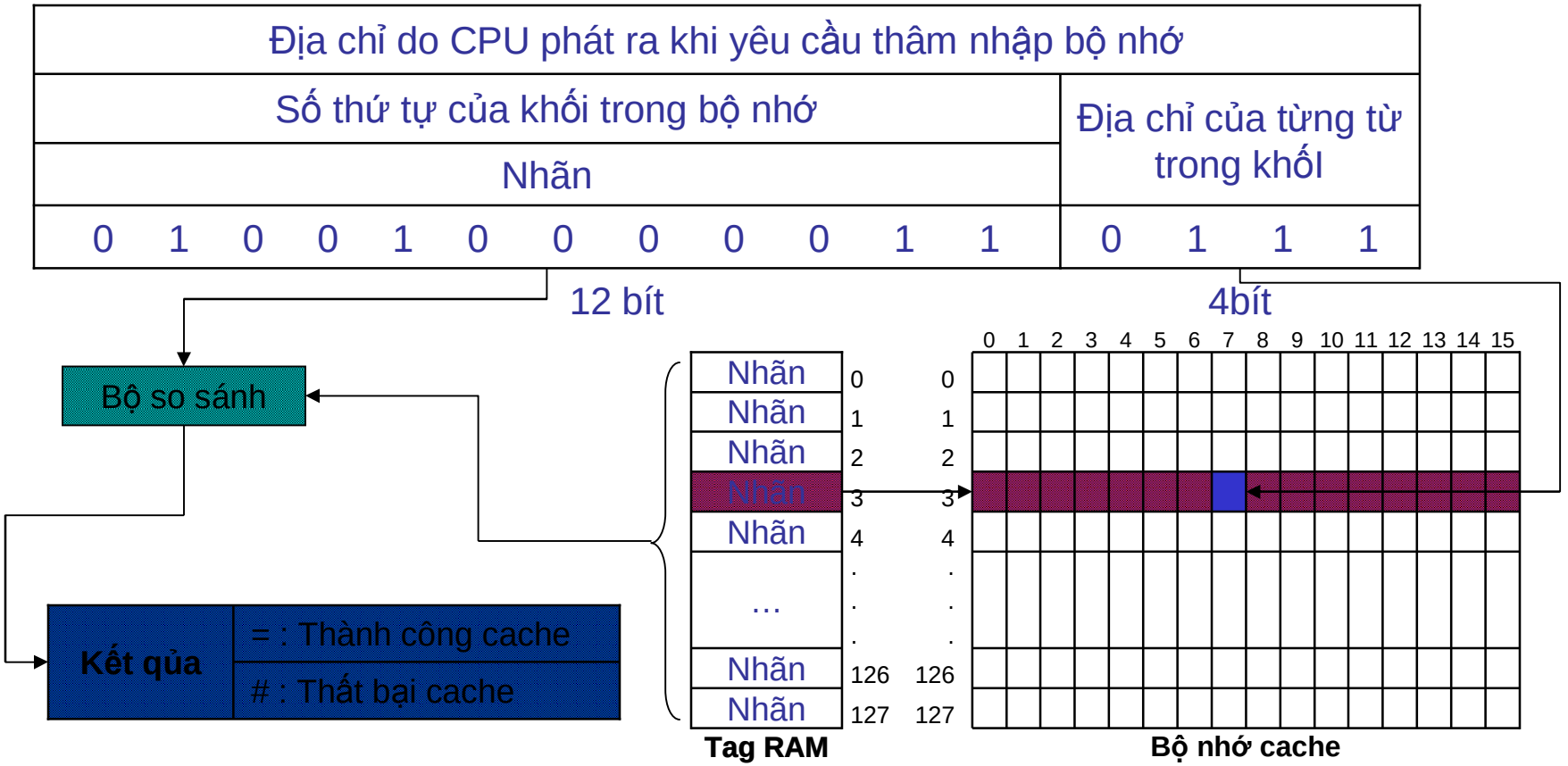
- Xếp khối tương ứng trực tiếp



4. Vận hành của cache

Ví dụ: Địa chỉ bộ xử lý yêu cầu thâm nhập:18487 → 4837H → $(0100100000110111)_2$

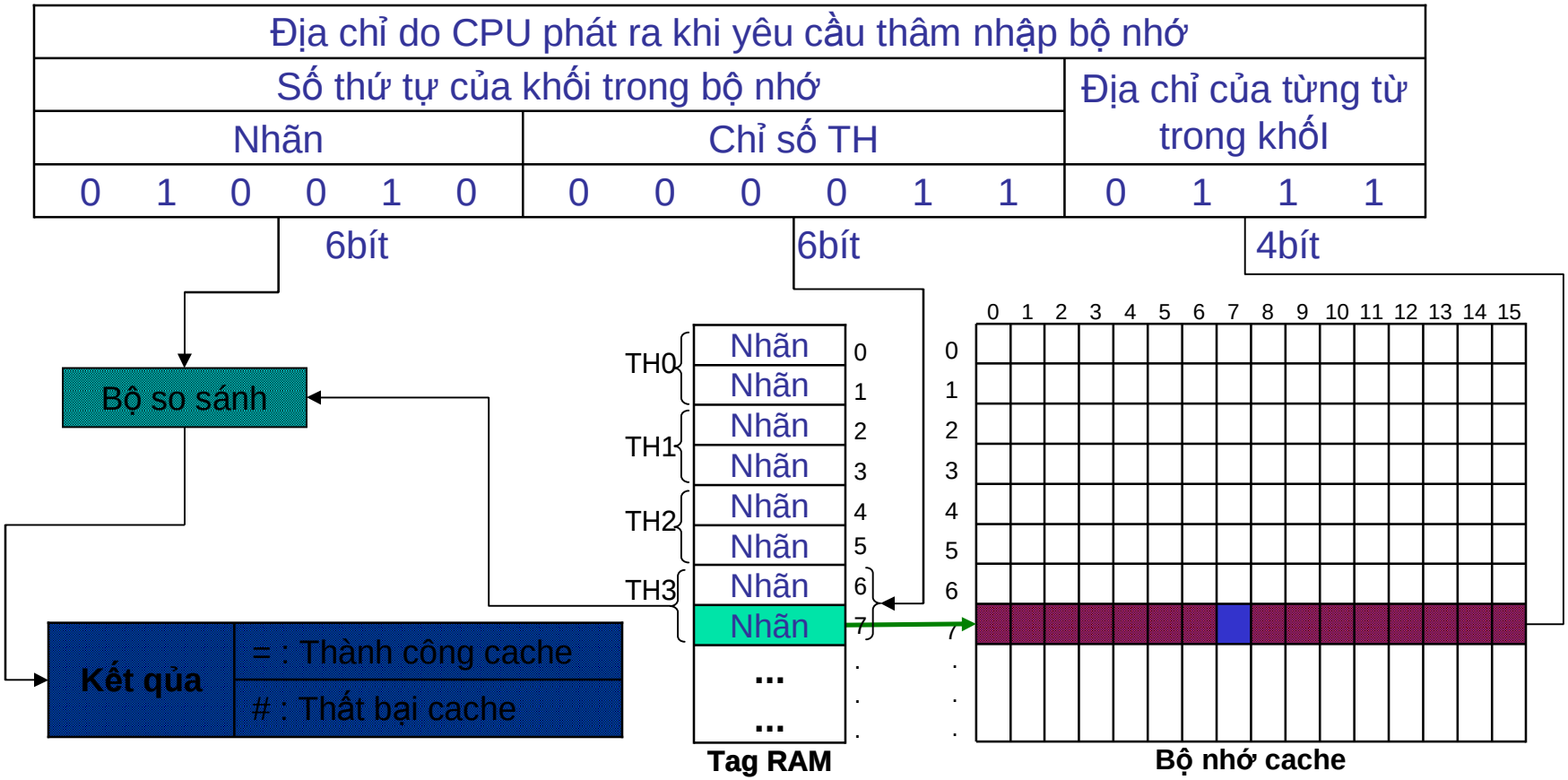
- Xếp khối hoàn toàn phối hợp



4. Vận hành của cache

Ví dụ: Địa chỉ bộ xử lý yêu cầu thâm nhập:18487_{10} → 4837H → $(0100100000110111)_2$

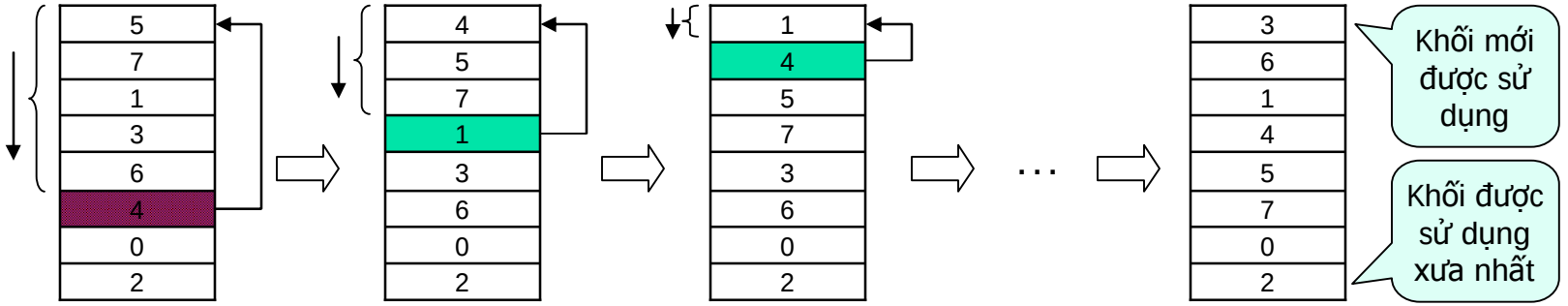
- Xếp khối phối hợp theo tập hợp (64 tập hợp)



4. Vận hành của cache

Q3: Khối nào phải được thay trong trường hợp thất bại cache.

- Khi có thất bại cache, bộ điều khiển cache, phải chọn lựa khối, để thay thế vào đó khối chứa dữ liệu mà bộ xữ lý cần.
- Trong cách xếp khối là hoàn toàn phối hợp hoặc phối hợp theo tập hợp. Có hai chiến thuật dùng để chọn khối phải thay thế:
  - **Chọn lựa ngẫu nhiên**: Để phân bố đồng đều việc thay thế.
  - **Vào trước ra trước** (FIFO: First In First Out): Khối được đưa vào cache đầu tiên, sẽ được thay thế trước nhất.
  - **Tần số sử dụng ít nhất** (LFU: Least Frequently Used): Khối trong cache được tham chiếu ít nhất
  - **Chọn khối xưa nhất** (LRU: Least Recently Used): Khối được chọn để thay thế là khối không được dùng từ lâu nhất.



Các cấp bộ nhớ

4. Vận hành của cache

Q4: Việc gì xảy ra khi ghi vào bộ nhớ ? (Chiến thuật ghi).

Phân tích quá trình đọc từ bộ nhớ và ghi vào bộ nhớ:

- Thông thường bộ xử lý thâm nhập cache là để đọc thông tin.
- Chỉ lối 25% các thâm nhập vào cache là để ghi số liệu.
- Ta cần tối ưu hóa việc đọc cache vì bộ xử lý phải đợi đến khi đọc hoàn thành nhưng sẽ không đợi đến khi ghi hoàn thành.
- Trường hợp đọc:
  - Một khối được đọc trong lúc nhản được đọc và so sánh
    ⇨ đọc khối có thể bắt đầu khi biết số thứ tự khối.
  - Nếu đọc thành công, dữ liệu sẽ được giao ngay cho bộ xử lý.
  - Nếu có thất bại thì bỏ qua trị số đã được đọc.
- Trường hợp ghi:
  - Việc thay đổi nội dung của một khối không thể bắt đầu trước khi biết có thành công cache hay không.
  - Vậy việc ghi vào bộ nhớ mất thời gian hơn việc đọc bộ nhớ.
  - Trong việc ghi bộ nhớ còn có một khó khăn nửa là bộ xử lý cho biết số byte phải ghi, thường từ 1 đến 8 byte.

4. Vận hành của cache

Q4 : Việc gì xảy ra khi ghi vào bộ nhớ ? (Chiến thuật ghi).

- **Trong trường hợp thành công cache:**
  - **Ghi đồng thời:** Thông tin được ghi đồng thời vào khối của cache và vào khốicủa bộ nhớ trong.
  - **Ghi lại:**
    - Thông tin được ghi vào khối trong cache.
    - Khối được chép lại vào bộ nhớ trong chỉ khi nó bị thay thế.
    - Người ta dùng một bít trạng thái để biết một khối đã được thay đổi hay chưa khi nó còn nằm trong cache.
    - Khi một khối chưa bị thay đổi thì không cần thiết phải ghi nó vào bộ nhớ trong khi nó bị thay thế.
- **Trong trường hợp thất bại cache:**
  - **Ghi có nạp:** Khối ghi vào bộ nhớ trong được đưa vào cache.
  - **Ghi không nạp:** Khối ghi vào bộ nhớ trong không được đưa vào cache.

5. Hiệu quả của cache

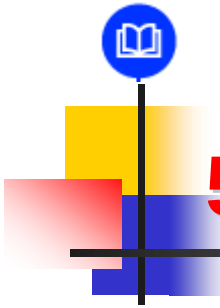
- Thông thường người ta dùng thời gian thâm nhập trung bình (TGTNTB) bộ nhớ trong để đánh giá hiệu quả của cache.

  TGTNTB =TGTN thành công + (tỷ lệ thất bại x trừng phạt thất bại)

  - **TGTN thành công:** Thời gian thâm nhập thành công là thời gian thâm nhập vào một thông tin trong một thành công cache.
  - **Tỉ số thất bại** là tỉ số giữa số thất bại cache và tổng số thâm nhập cache.
  - **Trừng phạt thất bại** là thời gian cần thiết để xử lý một thất bại cache.
  - Thời gian thâm nhập thành công và trừng phạt thất bại được đo bằng đơn vị thời gian hoặc bằng chu kỳ xung nhịp (clock cycle).

5. Hiệu quả của cache

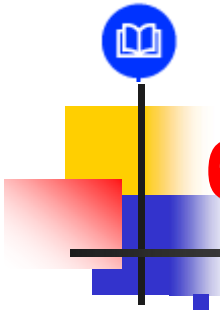
- Trong việc tìm kiếm thông tin trong cache phải chú ý làm giảm tỉ lệ thất bại mà các nguyên nhân chính là như sau :
  - **Khởi động:** Trong lần thâm nhập cache đầu tiên, không có thông tin cần tìm trong cache nên phải chuyển khối chứa thông tin đó vào cache.
  - **Khả năng:** Vì cache không thể chứa tất cả các khối cần thiết cho việc thi hành một chương trình nên gặp thất bại do cache thiếu khả năng (một khối bị lấy ra khỏi cache rồi lại được đưa vào sau nầy).
  - **Tranh chấp:** Nếu chiến thuật thay thế các khối là phối hợp theo tập hợp hay tương ứng trực tiếp, các thất bại do tranh chấp xảy ra vì một khối có thể bị đưa ra khỏi cache rồi được gọi vào sau đó nếu có nhiều khối phải được thay thế trong các tập hợp.



5. Hiệu quả của cache

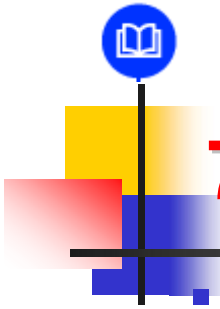
- Ba nguyên nhân trên cho ta ý niệm về nguyên nhân thất bại, nhưng mô hình đơn giản trên có những hạn chế của nó.
  - Mô hình nầy giúp ta thấy một số liệu trung bình nhưng chưa giải thích được từng thất bại một .

    Ví dụ: Nếu tăng kích thước cache thì giảm thất bại do tranh chấp và thất bại do khả năng vì cache càng lớn thì nhiều khối có thể được đưa vào.
  - Một thất bại có thể đi từ thất bại do khả năng đến thất bại do tranh chấp khi kích thước của cache thay đổi.
  - Khi nêu ba nguyên nhân trên ta đã không lưu ý đến cách thức thay thế các khối.
  - Cách thức nầy có thể dẫn đến những vận hành bất thường như là tỉ lệ thất bại cao lên khi độ phối hợp lớn lên.



6. Cache duy nhất hay cache riêng lẻ

- **Cache duy nhất:**
  - Bộ nhớ cache chứa đồng thời lệnh và dữ liệu.
  - Với một cache duy nhất, sẽ có tranh chấp khi một lệnh muốn thâm nhập một số liệu trong cùng một chu kỳ của giai đoạn đọc một lệnh khác.
- **Cache riêng lẻ:**
  - Phân biệt cache lệnh &ì cache dữ liệu (kiến trúc Harvard).
  - Giải pháp sau có lợi là tránh các khó khăn do kiến trúc khi thi hành các lệnh dùng kỹ thuật ống dẫn.
  - Cache riêng lẻ còn giúp tối ưu hóa mỗi loại cache về mặt kích thước tổng quát, kích thước các khối và độ phối hợp các khối.



7. Các mức cache

- Việc dùng cache có thể làm cho sự cách biệt giữa kích thước và thời gian thâm nhập giữa cache và bộ nhớ trong càng lớn.
- Người ta đưa vào nhiều mức cache:
  - **Cache mức một** (Cache L1): Thường là cache trong (nằm bên trong CPU), Cache loại này sử dụng để trao đổi dữ liệu giã CPU và bộ nhớ.
  - **Cache mức hai** (Cache L2): Thường là cache ngoài (cache nầy nằm bên ngoài CPU), Cache loại này sử dụng để trao đổi dữ liệu giữa bộ nhớ và ngoại vị .
  - **Cache mức ba** (Cache L3): Được sử dụng trong bộ xử lý thế hệ mới có nhu cầu trao đổi thông tin nhiều giữa bộ nhớ và ngoại vi (CPU của các máy chủ).
- Các nguyên tắc trình bày ở trên được áp dụng giữa các mức trong thang các cấp bộ nhớ.

8. Bộ nhớ trong

- Bộ nhớ trong thỏa mãn yêu cầu của cache và dùng làm đệm vào ra
  - Bộ nhớ trong là nơi chứa các thông tin từ ngoài đưa vào.
  - Bộ nhớ trong là nơi xuất ra các thông tin cho cache.
- Hiệu quả bộ nhớ dựa vào thời gian thâm nhập & bề rộng dải thông.
  - Thời gian thâm nhập bộ nhớ là phần tử quan trọng cho cache.
  - Dãi thông bộ nhớ là phần chính cho các tác vụ xuất nhập.
- Cải thiện hiệu quả bộ nhớ bằng cách nới rộng dải thông:
  - Cache cần bộ nhớ trong có thời gian thâm nhập nhỏ, nhưng thường thì dễ cải thiện dãi thông bộ nhớ nhờ nhiều cách tổ chức bộ nhớ mới, hơn là giảm thời gian thâm nhập cho cache.
  - Với việc dùng phổ biến các cache ngoài, dải thông của bộ nhớ trong cũng trở thành quan trọng cho cache.
  - Cache thụ hưởng các tiến bộ về dãi thông bằng cách tăng kích thước của mỗi khối của cache mà không tăng đáng kể trừng phạt thất bại cache.

8. Bộ nhớ trong

- **Dải thông (Bandwidth) của bộ nhớ trong là số lượng Mega bytes dữ liệu vận chuyển trên 1 giây.**

$$\text{Dải thông} = \frac{\text{Độ rộng Bus số liệu}}{8} \times \text{Tần số Bus}$$

Ví dụ:

- **DDR266**

$$\text{Dải thông} = \frac{64 \text{ bit}}{8} \times 266 \text{ MHz} = 2128 \text{ MB/s}$$

- **DDR400**

$$\text{Dải thông} = \frac{64 \text{ bit}}{8} \times 400 \text{ MHz} = 3200 \text{ MB/s}$$

- **RDRAM** (PC800)

$$\text{Dải thông} = \frac{2 \times 16 \text{ bit}}{8} \times 800 \text{ MHz} = 3200 \text{ MB/s}$$

8. Bộ nhớ trong

- **Các kỹ thuật nới rộng dải thông của bộ nhớ trong:**
  - **Nới rộng chiều dài ô nhớ trong:**
    - Thông thường cache và bộ nhớ trong có chiều rộng ô nhớ là chiều rộng 1 từ vì bộ xử lý thâm nhập vào một từ ô nhớ.
    - Nhân đôi, nhân bốn chiều rộng ô nhớ của cache và bộ nhớ trong làm lưu lượng thâm nhập bộ nhớ trong được nhân đôi hay nhân bốn.
    - Vậy phải chi tiêu thêm để nới rộng bus bộ nhớ (là bus nối bộ xử lý với bộ nhớ).
    - Ví dụ: Bộ xử lý có chiều dài ô nhớ trong lớn là bộ xử lý ALPHA AXP 21064. Cache ngoài, bộ nhớ trong và bus bộ nhớ đều có độ rộng là 256 bit.

8. Bộ nhớ trong

- **Các kỹ thuật nới rộng dải thông của bộ nhớ trong:**
  - Nếu sự vận hành bộ nhớ được thực hiện tuần tự, thì việc tính dải thông bộ nhớ khi biết thời gian thâm nhập là khá đơn giản.

Ví dụ:

Một bộ nhớ có thờ gian thâm nhập là T_L = 20ns, thời gian nạp lại là T_F = 5ns, độ rộng bus dữ liệu là 2 bytes. Tính dải thông bộ nhớ.

Giải:

Chu kỳ bộ nhớ là $T_C = T_L + T_F$ = 25ns.

Tốc độ truy cập (Throughput) là: $R = 1/T_C = 4.10^7$ operations/s

Dải thông của bộ nhớ là:

$B = 2$ bytes$\times 4.10^7$ operations/s = 80 Mbytes/s

8. Bộ nhớ trong

- **Các kỹ thuật nới rộng dải thông của bộ nhớ trong:**
  - Để tăng dải thông bộ nhớ, ta có thể áp dụng kỹ thuật giống như kỹ thuật ống dẫn trong bộ vi xử lý.

Ví dụ:

Một bộ nhớ có chu kỳ bộ nhớ là T_C = 40ns, độ rộng bus dữ liệu là 2 bytes. Dùng kỹ thuật ống dẫn cho phép gối đầu 4 truy xuất, giả sử bỏ qua thời gian đầu (từ khi đưa địa chỉ thứ I đến địa chỉ thứ II) Tính dải thông bộ nhớ.

Giải:

Dải thông của bộ nhớ là:

$$B = 2 \text{ bytes} \times (4/ T_C) = 200 \text{ Mbytes/s}$$

8. Bộ nhớ trong

- **Các kỹ thuật nới rộng dải thông của bộ nhớ trong:**
  - **Bộ nhớ đan chéo đơn giản:**
    - Các IC bộ nhớ có thể được tổ chức thành dải để đọc hay viết nhiều từ cùng một lúc thay vì chỉ đọc 1 từ, độ rộng của bus và của cache không thay đổi.
    - Khi gởi nhiều địa chỉ đến nhiều dải thì ta đọc được nhiều từ cùng một lúc.

Địa chỉ	Dãi 1	Địa chỉ	Dãi 2	Địa chỉ	Dãi 3	Địa chỉ	Dãi 4
0		1		2		3	
4		5		6		7	
8		9		10		11	
12		13		14		15	

Hình IV.7:Bộ nhớ đan chéo bậc 4

8. Bộ nhớ trong

- Bộ nhớ đan chéo cũng cho phép ghi vào bộ nhớ nhiều từ cùng một lúc.
- Tổ chức bộ nhớ đan chéo đơn giản không rắc rối nhiều so với tổ chức bình thường của bộ nhớ trong vì các dãy có thể dùng chung các đường địa chỉ với bộ điều khiển ô nhớ, và như thế mỗi dãy có thể dùng phần số liệu của bus bộ nhớ.

- **Bộ nhớ đan chéo tổ chức thành dải độc lập.**
  - Một tổ chức bộ nhớ đan chéo hiệu quả hơn, cho phép nhiều thâm thập bộ nhớ và như thế cho phép các dải làm việc độc lập với nhau.
    - Mỗi dải cần các đường địa chỉ riêng biệt và đôi khi cần bus số liệu riêng biệt.
    - Khi đó CPU có thể tiếp tục công việc trong lúc chờ đợi số liệu (thất bại cache).

8. Bộ nhớ trong

- **Bộ nhớ đan chéo tổ chức thành dải độc lập.**
  - Để tránh xung đột giữa các dải bộ nhớ. Các máy tính đa xử lý và máy tính vectơ, hệ thống bộ nhớ phải cho phép nhiều yêu cầu thâm nhập độc lập nhau.
  - Sự hiệu quả của hệ thống tùy thuộc vào tần số các trường hợp có yêu cầu độc lập thâm nhập vào các dải khác nhau.
  - Với sự đan chéo bình thường, các thâm nhập tuần tự sẽ gặp rắc rối nếu sự cách biệt giữa các địa chỉ là bội số của tổng số IC nhớ.
  - Một biện pháp mà các máy tính lớn dùng là làm giảm bớt các trường hợp xung đột tĩnh bằng cách tăng số lượng các dãi.
  - Ví dụ: Máy NEC SX/3 chia bộ nhớ trong ra 128 dãi.

8. Bộ nhớ trong

- **EDO RAM** (Extended Data Output): Là một loại DRAM áp dụng công nghệ mới. Nội dung của bộ nhớ loại này được lưu trữ lâu hơn và người ta có thể nới rộng chu kỳ làm tươi bộ nhớ. Nhờ vậy đọc dữ liệu nhanh hơn (40 MHz non Waitstates)
- **BEDO RAM** (Burst Extended Data Output): Là sự cải tiến của EDO RAM, nhờ cơ chế Burst - chuyển tải một khối dữ liệu khi biết địa chỉ của khối và chiều dài khối (=Burst). Cơ chế này cho phép tăng tốc độ truy cập dữ liệu (66MHz non Waitstates)
- **SDRAM** (Synchrone Dynamic RAM): Đây là sự cải tiến của DRAM cổ điển. Bên trong SDRAM gồm hai dãy bộ nhớ, những dữ liệu có thể được đọc một cách xen kẽ trên hai dãy nhờ vào một thủ tục đan xen đặc biệt. Với SDRAM ta có thể truy nhập dữ liệu bằng với tốc độ của CPU không cần thời gian chờ (Waitstates), tốc độ 100-166MHz (6-10ns).
- **DDRAM** (Double Data Rate SDRAM): Tương tự như SDRAM nhưng nhanh hơn gấp đôi SDRAM (DDR200, DDR266, DDR333, …).
- **RDRAM** (RAM BUS): Là thiết kế bộ nhớ hoàn toàn mới được phát triển từ Bus bộ nhớ chip-to-chip vớI thiết kế đặc biệt để truyền thông tin tốc độ cao.

9. Bộ nhớ ảo

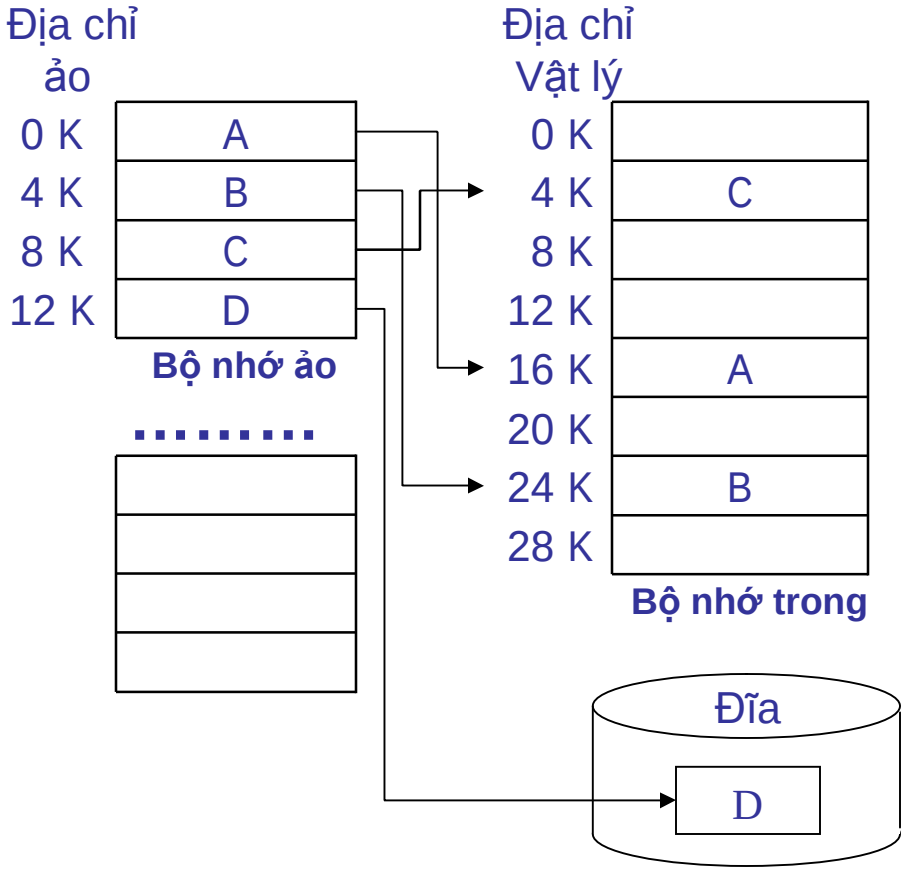
- **Thực thi chương trình có yêu cầu bộ nhớ lớn hơn bộ nhớ trong:**
  - Khi độ dài của chương trình vượt giới hạn dung lượng bộ nhớ thì người lập trình phải chia chương trình thành nhiều phần tự loại bỏ nhau, gọi là những phần phủ lắp (overlays).
    - Phần phủ lắp 0 sẽ chạy đầu tiên. Khi nó hoàn thành sẽ gọi những phần phủ lắp kế tiếp…
    - Người lập trình phải tự quản lý việc trao đổi thông tin giữa bộ nhớ & đĩa từ.
    - Trong một số hệ điều hành, việc trao đổi thông tin giữa bộ nhớ & đĩa từ của những phần phủ lắp có thể được thực hiện bởi hệ điều hành, nhưng việc phân chia chương trình thành các phần phủ lắp phải được thực hiện bởi lập trình viên.
  - Bộ nhớ ảo làm nhẹ trách nhiệm của các nhà lập trình bằng cách làm cho việc phân chia chương trình thành nhiều phần và quản lý việc trao đổi thông tin giữa đĩa từ và bộ nhớ được thực hiện một cách tự động.

9. Bộ nhớ ảo

- **Dùng bộ nhớ ảo cho các hệ thống máy tính đa chương trình:**
  - Trong các bộ xử lý hiện đại, bộ nhớ ảo được dùng để cho phép thực hiện cùng lúc nhiều tiến trình, mỗi tiến trình có một không gian định vị riêng.
  - Nếu tất cả các không gian định vị nầy thuộc bộ nhớ trong thì rất tốn kém.
  - Cần có cơ chế bảo vệ các tiến trình trong các hệ thống đa chương trình.

9. Bộ nhớ ảo

- **Tổ chức bộ nhớ ảo:**
  - Bộ nhớ ảo bao gồm bộ nhớ trong và bộ nhớ ngoài được phân tích thành khối
  - Cần bảo vệ và quản lý tự động các cấp bộ nhớ
  - Bộ nhớ ảo đơn giản hóa việc nạp chương trình vào bộ nhớ nhờ cơ chế gọi là sự tái định địa chỉ (address relocation).
  - Cơ chế tái định địa chỉ cho phép một chương trình có thể được thi hành khi nó nằm ở bất cứ vị trí nào trong bộ nhớ.



Hình IV.8: Một chương trình gồm 4 trang A,B,C,D. trong đó trang D nằm trong ổ đĩa

9. Bộ nhớ ảo

- **Sự khác biệt giữa cache và bộ nhớ ảo:**
  - **Sự khác biệt định lượng:**

Tham số	Cache	Bộ nhớ ảo
Chiều dài khối	16 – 128 bytes	4096 - 65536 bytes
Thời gian thâm nhập thành công	1 – 2 Xung nhịp	40 - 100 xung nhịp
Trừng phạt khi thất bại - Thời gian thâm nhập - Di chuyển số liệu	8 – 100 xung nhịp 6 – 60 Xung nhịp 2 – 20 Xung nhịp	700.000 - 6 triệu xung 500.000 - 4 triệu xung 200.000 - 2 triệu xung
Tỷ số thất bại	0.5 % – 10 %	0,00001% - 0,001%
Dung lượng	16 KB – 1000 KB	16 MB - 8192 MB

Hình IV.9. Đại lượng điển hình cho bộ nhớ cache và bộ nhớ ảo.

9. Bộ nhớ ảo

- **Sự khác biệt giữa cache và bộ nhớ ảo:**

  Khác biệt về cơ chế vận hành:

  - Khi thất bại cache, sự thay thế một khối trong cache được điều khiển bằng phần cứng, trong khi sự thay thế trong bộ nhớ ảo chủ yếu do hệ điều hành.
  - Không gian định vị mà bộ xử lý quản lý là không gian định vị của bộ nhớ ảo, trong lúc đó thì dung lượng bộ nhớ cache không tùy thuộc vào không gian định vị bộ xử lý.
  - Bộ nhớ ngoài còn được dùng để lưu trữ tập tin ngoài nhiệm vụ là hậu phương của bộ nhớ trong (trong các cấp bộ nhớ).

9. Bộ nhớ ảo

- **Các hệ thống bộ nhớ ảo có thể được chia thành 3 loại:**
  - Loại với khối có dung lượng cố định gọi là trang. Định vị trang xác định một địa chỉ trong trang, giống như định vị trong cache.
  - Loại với khối có chiều dài thay đổi gọi là đoạn. Trong định vị đoạn cần 2 từ : một từ chứa số thứ tự đoạn và một từ chứa độ dời trong đoạn. Chương trình dịch gặp khó khăn nhiều trong định vị đoạn.
  - Do việc thay thế các đoạn, ngày nay ít máy tính dùng định vị đoạn thuần túy. Một vài máy dùng cách hỗn hợp gọi là đoạn trang. Trong đó mỗi đoạn chứa một số nguyên các trang.

9. Bộ nhớ ảo

Các vấn đề của bộ nhớ ảo:

Q1: Một khối được đặt tại đâu trong bộ nhớ trong ?

- Việc trừng phạt bộ nhớ ảo khi có thất bại rất lớn, do phải thâm nhập ổ đĩa.
- Việc thâm nhập nầy rất chậm nên người ta chọn phương án hoàn toàn phối hợp trong đó các khối (trang) có thể nằm ở bất kỳ vị trí nào trong bộ nhớ trong. Cách này cho tỉ lệ thất bại thấp.

Q2: Làm sao để tìm một khối khi nó đang nằm trong bộ nhớ trong ?

- Định vị trang và định vị đoạn đều dựa vào một cấu trúc dữ liệu trong đó số thứ tự trang hoặc số thứ tự đoạn được có chỉ số.
- Cho định vị đoạn, địa chỉ ảo cuối cùng được xác lập bằng cách cộng địa chỉ trong trang và địa chỉ trong đoạn.
- Cho định vị trang, địa chỉ cuối cùng là việc đặt kề nhau số thứ của trang với địa chỉ trong trang (hình IV.10).

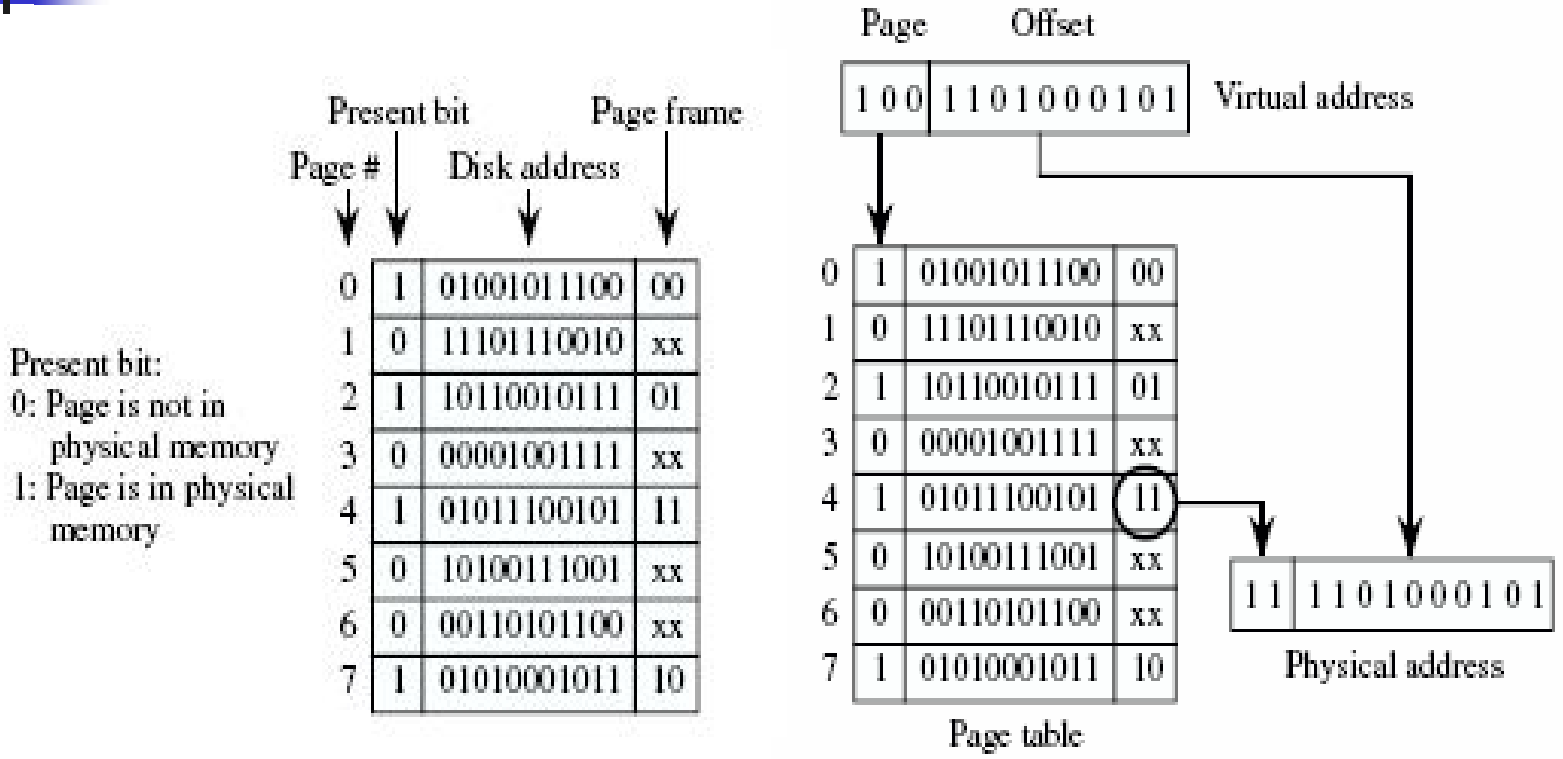
Q3: Khối nào phải được thay thế khi có thất bại trang ?

Hầu hết các hệ điều hành đều cố gắng thay thế khối ít dùng gần đây nhất (LRU: Least Recent Utilized) vì nghĩ rằng đây là khối ít cần nhất.

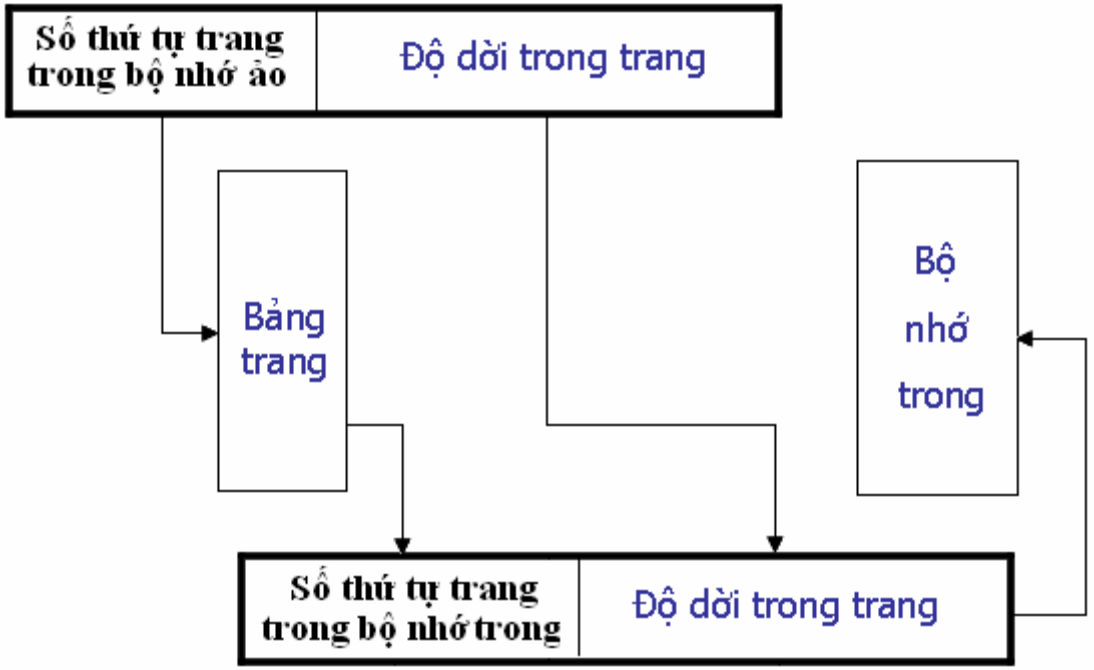
Q4: Việc gì xảy ra khi cần ghi số liệu ?

Chiến thuật ghi là ghi lại nghĩa là thông tin chỉ được viết vào khối của bộ nhớ trong. Khối có thay đổi thông tin, được chép vào đĩa nếu khối này bị thay thế.

9. Bộ nhớ ảo


Present bit
Page frame
Page #
Disk address
Present bit:
0: Page is not in physical memory
1: Page is in physical memory
0 1 01001011100 00
1 0 11101110010 xx
2 1 10110010111 01
3 0 00001001111 xx
4 1 01011100101 11
5 0 10100111001 xx
6 0 00110101100 xx
7 1 01010001011 10
Page
Offset
1 0 0 1 1 0 1 0 0 0 1 0 1
Virtual address
0 1 01001011100 00
1 0 11101110010 xx
2 1 10110010111 01
3 0 00001001111 xx
4 1 01011100101 11
5 0 10100111001 xx
6 0 00110101100 xx
7 1 01010001011 10
Page table
1 1 1 1 0 1 0 0 0 1 0 1
Physical address

9. Bộ nhớ ảo



Ánh xạ địa chỉ ảo vào địa chỉ vật lý thông qua bảng trang.

9. Bộ nhớ ảo

Ví dụ:

Cho 1 bộ nhớ ảo có 16 trang, mỗi trang có dung lượng là 4KB, mỗi ô nhớ 1 byte. Bộ nhớ trong có 8 trang.

1. Tính số bit của địa chỉ ảo và của địa chỉ thực.
2. Cho biết địa chỉ thực khi CPU xuất ra địa chỉ ảo là 8196 để truy xuất bộ nhớ. Cho biết thành công hay thất bại, tìm địa chỉ thật.
3. Cho biết địa chỉ thực khi CPU xuất ra địa chỉ ảo là 8A3CH để truy xuất bộ nhớ. Cho biết thành công hay thất bại, tìm địa chỉ thật.

Giả sử trang thật ít dùng nhất hiện thời là trang 4 và bảng trang như hình bên.

0	010	1
1	001	1
2	110	1
3	000	1
4	100	1
5	011	1
6	xxx	0
7	xxx	0
8	xxx	0
9	101	1
10	xxx	0
11	111	1
12	xxx	0
13	xxx	0
14	xxx	0
15	xxx	0

9. Bộ nhớ ảo

1. Số trang ảo: 16 → STT trang 4 bit,

   Dung lượng trang là 4kB → độ dời trang 12 bit.

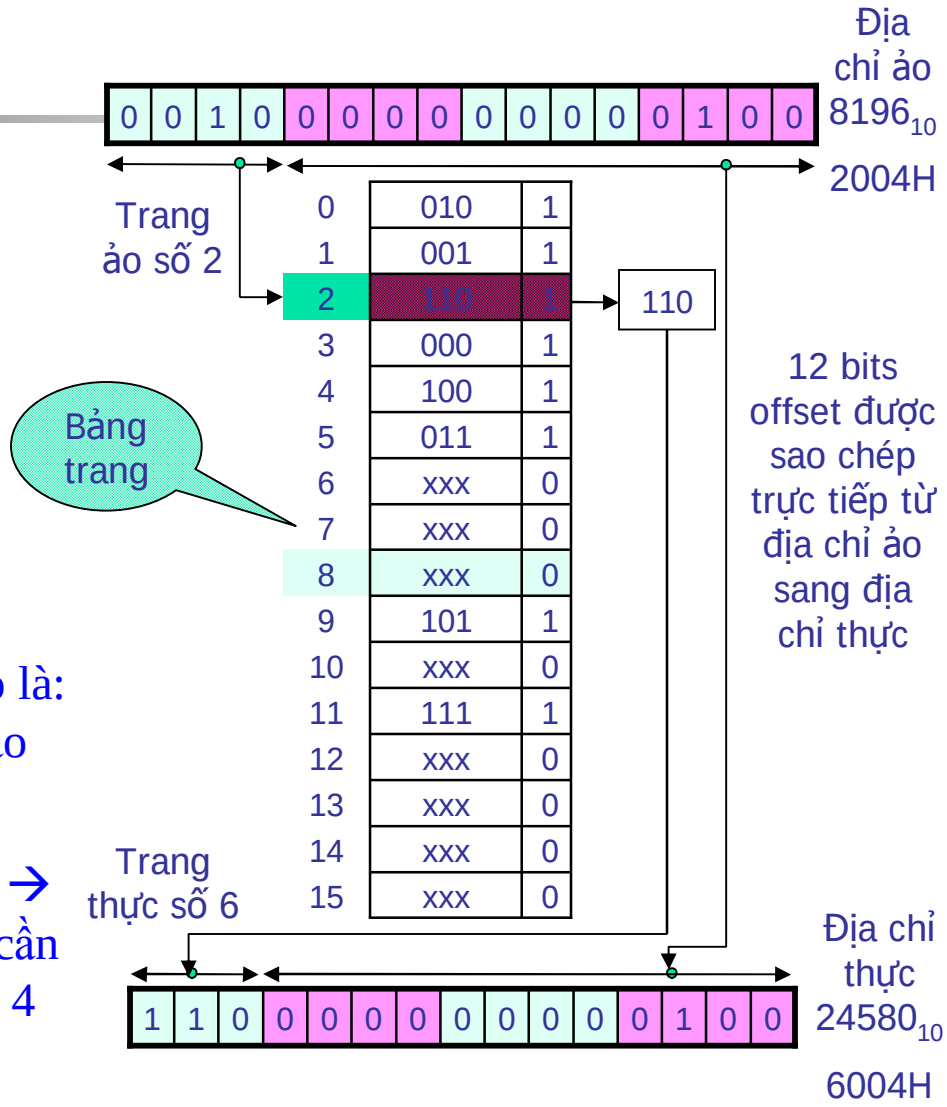
   → Địa chỉ ảo 16 bit

Số trang thật: 8 → STT trang 3 bit

→ Địa chỉ thật 16 bit

2. Địa chỉ 8196_{10} = 2004H → số trang ảo là: $0010_2 = 2_{10}$ → Thành công bộ nhớ ảo
   → Địa chỉ thật là 24580_{10}

3. Địa chỉ 8A3CH ở trong trang ảo thứ 8 → thật bại → Xuống bộ nhớ lấy trang cần thiết lê đặt và bộ nhớ trong tại trang 4
   → Địa chỉ thật là 4A3CH = 19004



9. Bộ nhớ ảo

000	1	00
001	0	
010	1	11
011	0	
100	0	
101	0	
110	1	01
111	0	

Ví dụ 2: Một tổ chức bộ nhớ như sau: bộ nhớ thật 64 byte, bộ nhớ ảo 128 byte, kích thước trang 16 byte.

1. Lập bảng trang mô tả cơ chế nhận diện trang trong tổ chức bộ nhớ ảo.
2. Cho bảng trang và dữ liệu trong bộ nhớ trong như hình sau, khi CPU xuất ra địa chỉ 1100001 để đọc bộ nhớ. Cho biết số liệu đọc được.

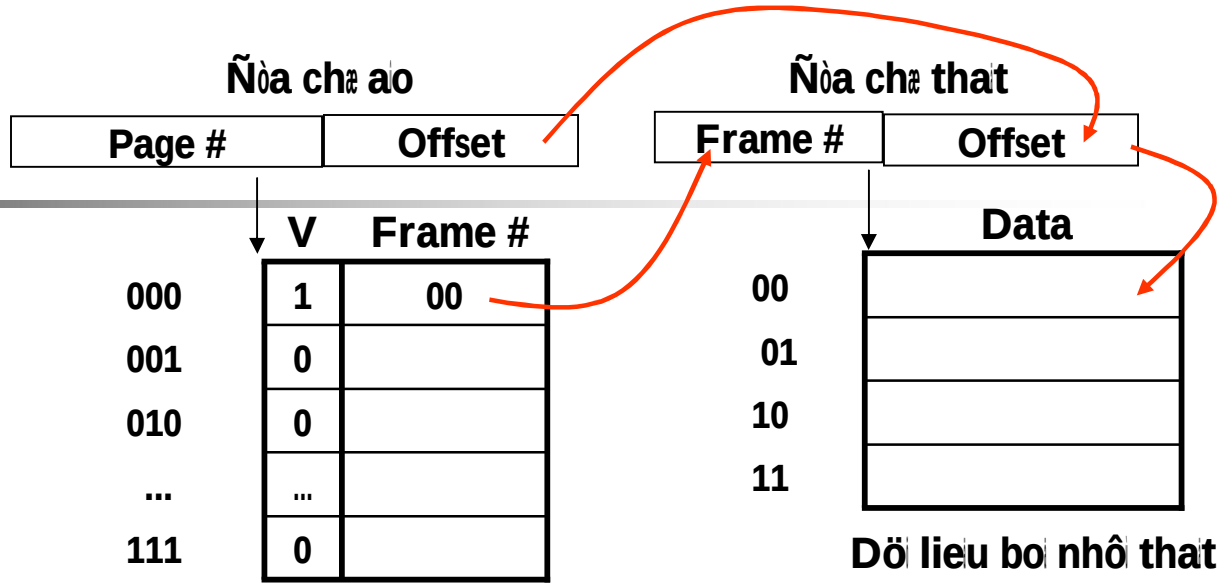
00	C7 45 F9 E2 99 17 02 1120 21 23 67 25 11 81A8
01	36 45 33 2F 1E 56 11 35 89 44 92 33 66 71 22 21
10	E4 .....
11	41 .... 12 71

9. Bộ nhớ ảo

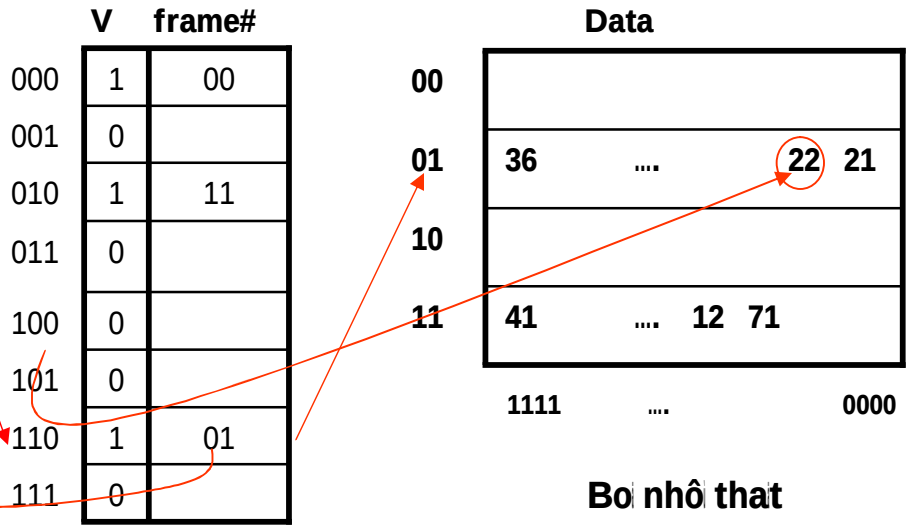
Ví dụ:

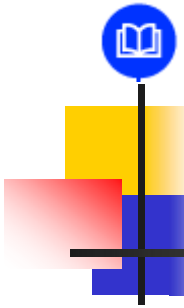
Giải:

- Số trang thật = 64/16 = 4
  → Frame #: 2 bit
- Mỗi trang có 16 byte → Offset: 4 bit
- Số trang ảo = 128/16 = 8 → Page #: 3 bit



- Khi CPU ñoïc ñòa chæ ao: 110 0001
  Ñòa chæ that töông öng: 01 0001
  Data: 22, Virtual mem. hit.





10. Bảo vệ các tiến trình bằng bộ nhớ ảo

- **Lý do cần phải bảo vệ các tiến trình:**
  - Đa chương trình (multi program): Nhiều chương trình chạy song song nhau.
  - Trong máy tính đa chương trình, bộ xử lý và bộ nhớ được các chương trình chia xẽ một cách tương tác (interactive). Phải có một cơ chế chuyển đổi từ một tiến trình sang một tiến trình khác đảm bảo tiến trình vận hành đúng đắn.
- **Trách nhiệm của nhà thiết kế máy tính và nhà thiết kê hệ điều hành:**
  - Nhà thiết kế hệ điều hành phải đảm bảo các tiến trình không ảnh hưởng lên nhau bằng cách chia bộ nhớ cho các tiến trình.
  - Nhà thiết kế máy tính phải đảm bảo bộ xử lý có thể lưu giữ và phục hồi trạng thái của các tiến trình. Ngoài ra có thêm 3 trách nhiệm trợ giúp nhà thiết kế HĐH:
    - Cung cấp 2 chế độ vận hành (người sử dụng và hệ thống).
    - Cung cấp một tập hợp con trạng thái của bộ xủ lý để tiến trình NSD dùng nhưng không được sửa đổi.
    - Cung cấp các cơ chế chuyển đổi từ chế độ người dùng sang chế độ người diều hành và ngược lại.
- **Sự bảo vệ tiến trình được thực hiện bằng cách dùng bộ nhớ ảo:**

  Địa chỉ bộ xử lý đưa ra được biến đổi từ địa chỉ ảo sang địa chỉ vật lý nên có thể phát hiện sự thâm nhập trái phép bộ nhớ trước khi sự thâm nhập này gây hư hại.

Câu hỏi ?

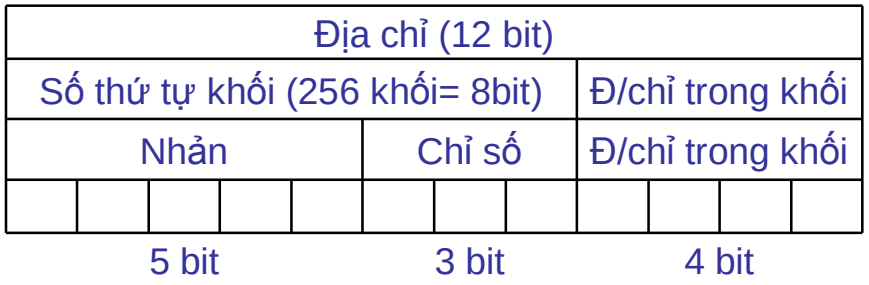
1. Điểm khác biệt của ROM và RAM là gì?
2. Trong máy tính bộ nhớ ROM được dùng làm gì?
3. Nguyên nhân chính làm cho RAM động (DRAM) có chu kỳ bộ nhớ lớn hơn hai lần thời gian thâm nhập là gì ?
4. Trong máy tính bộ nhớ RAM động (DRAM) được dùng làm gì?
5. Trong máy tính bộ nhớ RAM tỉnh (SRAM) được dùng làm gì?
6. Mục tiêu của các cấp bộ nhớ là gì?
7. Trong bộ nhớ cache, nguyên tắt thời gian được áp dụng như thế nào nào?
8. Trong bộ nhớ cache, nguyên tắt không gian được áp dụng thế nào nào?
9. Trong cách xếp khối tương ứng trực tiếp điểm bất lợi lớn nhất là gì?
10. Trong cách xếp khối hoàn toàn phối hợp điểm bất lợi lớn nhất là gì?
11. Trong ba cách xếp khối, cách nào có phần nhản nhỏ nhất?
12. Tại sao nhản của địa chỉ và dữ liệu trong cache được đọc cùng lúc khi thâm nhập bộ nhớ?
13. Tại sao trong các cách thay thế khối người ta ít áp dụng đối với cách xếp khối tương ứng trực tiếp ?
14. Trong trường hợp ghi, cách ghi lại được sử dụng phổ biến. Tại sao?

Câu hỏi ?

15. Tại sao cách thay thế khối bằng cách chọn khối đã không sử dụng từ lâu nhất (LRU) được dùng nhiều nhất ?
16. Với tổ chức một bộ nhớ cache duy nhất, sẽ gặp khó khăn khi dùng kỹ thuật ống dẫn, tại sao ?
17. Ngoài việc khắc phục được các khó khăn do kiến trúc khi dùng kỹ thuật ống dẫn, tổ chức cach riêng lẻ (cache lệnh và cache dữ liệu) còn lợi điểm gì ?
18. Trong các cố gắng nâng cao hiệu quả hoạt động của bộ nhớ, nới rộng dãy thông bộ nhớ có lợi gì trong hoạt động của bộ nhớ cache ?
19. Giải pháp nới rộng chiều dài ô nhớ để nới rộng dãy thông có điểm bất lợi gì?
20. Giải pháp dùng bộ nhớ đan chéo xếp thành dãy độc lập sẽ giảm được thành phần nào sau đây khi xử lý thất bại cache ?
21. Bộ nhớ ảo giúp ích gì trong thực hiện các mục tiêu của các cấp bộ nhớ ?
22. Tai sao trong bộ nhớ ảo cách xếp khối hoàn toàn phối hợp được chọn ?
23. Điểm khác biệt về cơ chế vận hành giữa bộ nhớ ảo và bộ nhớ cache là gì ?
24. Nhiệm vụ biến đổi địa chỉ ảo thành địa chỉ vật lý trong bộ nhớ ảo là nhiệm vụ của bộ phận nào ?
25. Việc bảo vệ các tiến trình trong các hệ thống đa chương là nhiệm vụ của bộ phận nào ?

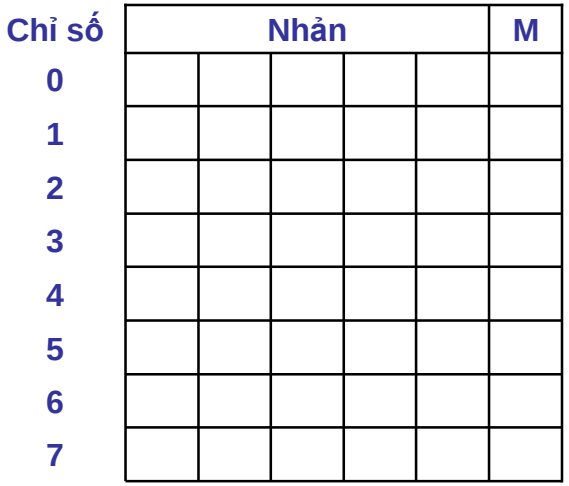
Bài tập

Cho một bộ nhớ cache tương ứng trực tiếp có 8 khối, mỗi khối có 16 byte. Bộ nhớ trong có 256 khối. Khi thành công cache sử dụng cách ghi lại; Khi thất bại cache dùng cách ghi có nạp. Giả sử lúc khởi động, 8 khối sau đây của bộ nhớ trong đã được đưa lên cache: 8, 17, 23, 34, 38, 67, 69, 132.



- Viết bảng nhãn của các khối hiện đang nằm trong cache.
- Cập nhật bảng nhản khi CPU lần lượt đưa ra các địa chỉ sau đây để đọc / ghi vào bộ nhớ trong:

1-Đọc: 43FH, 2-Đọc: 82AH,
3-Đọc: 915H, 4-Ghi: 08CH,
5-Ghi: B4AH, 6-Ghi: 45DH,
7-Đọc: 5E9H, 8-Ghi: C7AH,
9-Đọc: D85H, 10-Ghi: 92AH,
11-Đọc: 6C5H, 12-Ghi: 458H,



Bảng nhản

Bài tập

- Bảng nhản sau khi khởi động:

Khối	Nhản					Chỉ số		
8	0	0	0	0	1	0	0	0
17	0	0	0	1	0	0	0	1
23	0	0	0	1	0	1	1	1
34	0	0	1	0	0	0	1	0
38	0	0	1	0	0	1	1	0
67	0	1	0	0	0	0	1	1
69	0	1	0	0	0	1	0	1
132	1	0	0	0	0	1	0	0

Chỉ số	Nhản					M
0	0	0	0	0	1	0
1	0	0	0	1	0	0
2	0	0	1	0	0	0
3	0	1	0	0	0	0
4	1	0	0	0	0	0
5	0	1	0	0	0	0
6	0	0	1	0	0	0
7	0	0	0	1	0	0

Bảng nhản

Bài tập

- **Cập nhật bảng nhản khi CPU lần lượt đưa ra các địa chỉ sau đây để đọc / ghi vào bộ nhớ trong:**

1-Đọc:43FH: **010000111111**
2-Đọc:82AH: **100000101010**
3-Đọc:915H: **100100010101**
4-Ghi:08CH: **000010001100**
5-Ghi:B4AH: **101101001010**
6-Ghi:45DH: **010001011101**
7-Đọc:5E9H: **010111101001**
8-Ghi:C7AH: **110001111010**
9-Đọc:D85H: **110110001101**
10-Ghi:92AH: **100100101010**
11-Đọc:6C5H: **011011000101**
12-Ghi:458H: **010001011000**

Chỉ số	Nhản					M
0	0	0	0	0	1	0
1	0	0	0	1	0	0
2	0	0	1	0	0	0
3	0	1	0	0	0	0
4	1	0	0	0	0	0
5	0	1	0	0	0	0
6	0	0	1	0	0	0
7	0	0	0	1	0	0

Bảng nhản

1-Đọc:43FH: 01000011111

Chỉ số	Nhản					M
0	0	0	0	0	1	0
1	0	0	0	1	0	0
2	0	0	1	0	0	0
3	0	1	0	0	0	0
4	1	0	0	0	0	0
5	0	1	0	0	0	0
6	0	0	1	0	0	0
7	0	0	0	1	0	0

Trước khi đọc

Thành công

Chỉ số	Nhản					M
0	0	0	0	0	1	0
1	0	0	0	1	0	0
2	0	0	1	0	0	0
3	0	1	0	0	0	0
4	1	0	0	0	0	0
5	0	1	0	0	0	0
6	0	0	1	0	0	0
7	0	0	0	1	0	0

Sau khi đọc

M=0 ⇨ Đọc dữ liệu từ bộ nhớ cache, bảng nhản không thay đổi.

2-Đọc:82AH: 100000101010

Chỉ số	Nhản					M
0	0	0	0	0	1	0
1	0	0	0	1	0	0
2	0	0	1	0	0	0
3	0	1	0	0	0	0
4	1	0	0	0	0	0
5	0	1	0	0	0	0
6	0	0	1	0	0	0
7	0	0	0	1	0	0

Trước khi đọc

Thất bại

Chỉ số	Nhản					M
0	0	0	0	0	1	0
1	0	0	0	1	0	0
2	1	0	0	0	0	0
3	0	1	0	0	0	0
4	1	0	0	0	0	0
5	0	1	0	0	0	0
6	0	0	1	0	0	0
7	0	0	0	1	0	0

Sau khi đọc

M=0 ⇨ Nạp khối mới lên cache, cập nhật bảng nhản, đọc dữ liệu từ bộ nhớ cache.

3-Đọc:915H: 100100010101

Chỉ số	Nhản					M
0	0	0	0	0	1	0
1	0	0	0	1	0	0
2	1	0	0	0	0	0
3	0	1	0	0	0	0
4	1	0	0	0	0	0
5	0	1	0	0	0	0
6	0	0	1	0	0	0
7	0	0	0	1	0	0

Trước khi đọc

Thất bại

Chỉ số	Nhản					M
0	0	0	0	0	1	0
1	1	0	0	1	0	0
2	1	0	0	0	0	0
3	0	1	0	0	0	0
4	1	0	0	0	0	0
5	0	1	0	0	0	0
6	0	0	1	0	0	0
7	0	0	0	1	0	0

Sau khi đọc

M=0 ⇨ Nạp khối mới lên cache, cập nhật bảng nhản, đọc dữ liệu từ bộ nhớ cache.

4-Ghi:08CH: 000010001100

Chỉ số	Nhản					M
0	0	0	0	0	1	0
1	1	0	0	1	0	0
2	1	0	0	0	0	0
3	0	1	0	0	0	0
4	1	0	0	0	0	0
5	0	1	0	0	0	0
6	0	0	1	0	0	0
7	0	0	0	1	0	0

Trước khi ghi

Thành công

Chỉ số	Nhản					M
0	0	0	0	0	1	1
1	1	0	0	1	0	0
2	1	0	0	0	0	0
3	0	1	0	0	0	0
4	1	0	0	0	0	0
5	0	1	0	0	0	0
6	0	0	1	0	0	0
7	0	0	0	1	0	0

Sau khi ghi

M=0 ⇨ Chỉ ghi vào bộ nhớ cache, cập nhật bit M=1

5-Ghi:B4AH: 101101001010

Chỉ số	Nhản					M
0	0	0	0	0	1	1
1	1	0	0	1	0	0
2	1	0	0	0	0	0
3	0	1	0	0	0	0
4	1	0	0	0	0	0
5	0	1	0	0	0	0
6	0	0	1	0	0	0
7	0	0	0	1	0	0

Trước khi ghi

Thất bại

Chỉ số	Nhản					M
0	0	0	0	0	1	1
1	1	0	0	1	0	0
2	1	0	0	0	0	0
3	0	1	0	0	0	0
4	1	0	1	1	0	0
5	0	1	0	0	0	0
6	0	0	1	0	0	0
7	0	0	0	1	0	0

Sau khi ghi

M=0 ⇨ Ghi vào bộ nhớ trong và nạp khối lên cache, cập nhật bảng nhản

6-Ghi:45DH: 01000101101

Chỉ số	Nhản					M
0	0	0	0	0	1	1
1	1	0	0	1	0	0
2	1	0	0	0	0	0
3	0	1	0	0	0	0
4	1	0	1	1	0	0
5	0	1	0	0	0	0
6	0	0	1	0	0	0
7	0	0	0	1	0	0

Trước khi ghi

Thành công

Chỉ số	Nhản					M
0	0	0	0	0	1	1
1	1	0	0	1	0	0
2	1	0	0	0	0	0
3	0	1	0	0	0	0
4	1	0	1	1	0	0
5	0	1	0	0	0	1
6	0	0	1	0	0	0
7	0	0	0	1	0	0

Sau khi ghi

M=0 ⇨ Chỉ ghi vào bộ nhớ cache, cập nhật bit M=1

7-Đọc:5E9H: 010111101001

Chỉ số	Nhản					M
0	0	0	0	0	1	1
1	1	0	0	1	0	0
2	1	0	0	0	0	0
3	0	1	0	0	0	0
4	1	0	1	1	0	0
5	0	1	0	0	0	1
6	0	0	1	0	0	0
7	0	0	0	1	0	0

Trước khi đọc

Thất bại

Chỉ số	Nhản					M
0	0	0	0	0	1	1
1	1	0	0	1	0	0
2	1	0	0	0	0	0
3	0	1	0	0	0	0
4	1	0	1	1	0	0
5	0	1	0	0	0	1
6	0	1	0	1	1	0
7	0	0	0	1	0	0

Sau khi đọc

M=0 ⇨ Nạp khối mới lên cache, cập nhật bảng nhản, đọc dữ liệu từ bộ nhớ cache.

8-Ghi:C7AH: 110001111010

Chỉ số	Nhản					M
0	0	0	0	0	1	1
1	1	0	0	1	0	0
2	1	0	0	0	0	0
3	0	1	0	0	0	0
4	1	0	1	1	0	0
5	0	1	0	0	0	1
6	0	1	0	1	1	0
7	0	0	0	1	0	0

Trước khi ghi

Thất bại

Chỉ số	Nhản					M
0	0	0	0	0	1	1
1	1	0	0	1	0	0
2	1	0	0	0	0	0
3	0	1	0	0	0	0
4	1	0	1	1	0	0
5	0	1	0	0	0	1
6	0	1	0	1	1	0
7	1	1	0	0	0	0

Sau khi ghi

M=0 ⇨ Ghi vào bộ nhớ trong và nạp khối lên cache, cập nhật bảng nhản.

9-Đọc:D85H: 110110001101

Chỉ số	Nhản					M
0	0	0	0	0	1	1
1	1	0	0	1	0	0
2	1	0	0	0	0	0
3	0	1	0	0	0	0
4	1	0	1	1	0	0
5	0	1	0	0	0	1
6	0	1	0	1	1	0
7	1	1	0	0	0	0

Trước khi đọc

Thất bại

Chỉ số	Nhản					M
0	1	1	0	1	1	0
1	1	0	0	1	0	0
2	1	0	0	0	0	0
3	0	1	0	0	0	0
4	1	0	1	1	0	0
5	0	1	0	0	0	1
6	0	1	0	1	1	0
7	1	1	0	0	0	0

Sau khi đọc

M=1 ⇨ Chép khối vào bộ nhớ, nạp khối mới lên cache, cập nhật bảng nhản và đặt bit M=0, đọc dữ liệu từ bộ nhớ cache.

10-Ghi:92AH: 100100101010

Chỉ số	Nhản					M
0	1	1	0	1	1	0
1	1	0	0	1	0	0
2	1	0	0	0	0	0
3	0	1	0	0	0	0
4	1	0	1	1	0	0
5	0	1	0	0	0	1
6	0	1	0	1	1	0
7	1	1	0	0	0	0

Trước khi đọc

Thất bại

Chỉ số	Nhản					M
0	1	1	0	1	1	0
1	1	0	0	1	0	0
2	1	0	0	1	0	0
3	0	1	0	0	0	0
4	1	0	1	1	0	0
5	0	1	0	0	0	1
6	0	1	0	1	1	0
7	1	1	0	0	0	0

Sau khi đọc

M=0 ⇨ Ghi vào bộ nhớ trong và nạp khối lên cache, cập nhật bảng nhản.

11-Đọc:6C5H: 011011000101

Chỉ số	Nhản					M
0	1	1	0	1	1	0
1	1	0	0	1	0	0
2	1	0	0	1	0	0
3	0	1	0	0	0	0
4	1	0	1	1	0	0
5	0	1	0	0	0	1
6	0	1	0	1	1	0
7	1	1	0	0	0	0

Trước khi đọc

Thất bại

Chỉ số	Nhản					M
0	1	1	0	1	1	0
1	1	0	0	1	0	0
2	1	0	0	1	0	0
3	0	1	0	0	0	0
4	0	1	1	0	1	0
5	0	1	0	0	0	1
6	0	1	0	1	1	0
7	1	1	0	0	0	0

Sau khi đọc

M=0 ⇨ Nạp khối mới lên cache, cập nhật bảng nhản, đọc dữ liệu từ bộ nhớ cache.

12-Ghi:458H: 010001011000

Chỉ số	Nhản					M
0	1	1	0	1	1	0
1	1	0	0	1	0	0
2	1	0	0	1	0	0
3	0	1	0	0	0	0
4	0	1	1	0	1	0
5	0	1	0	0	0	1
6	0	1	0	1	1	0
7	1	1	0	0	0	0

Trước khi đọc

Thành công

Chỉ số	Nhản					M
0	1	1	0	1	1	0
1	1	0	0	1	0	0
2	1	0	0	1	0	0
3	0	1	0	0	0	0
4	0	1	1	0	1	0
5	0	1	0	0	0	1
6	0	1	0	1	1	0
7	1	1	0	0	0	0

Sau khi đọc

M=1 ⇨ Chỉ ghi vào bộ nhớ cache, bảng nhản không thay đổi