Cache

1. *Các Loại cục bộ*

### *1.Temporal locality(Cục bộ về thời gian)*

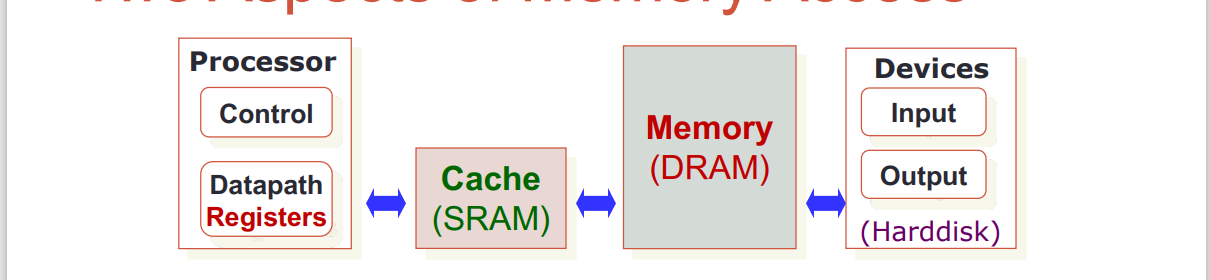
* Nếu một ô nhớ được dùng đến **trong thời điểm hiện tại** thì nó dễ có **khả năng được dùng đến lần nữa trong tương lai gần**

### *2.Spatial locality(Cục bộ về không gian)*

* Nếu **một ô nhớ được dùng đến** trong thời điểm hiện tại thì **những ô lân cận có khả năng sắp được dùng đến**

## *Tìm hiểu khái quát Cache*

\*Hình ảnh trong sơ đồ phần cứng



* Cache là **một bộ nhớ nhỏ về dung lượng** nhưng **tốc độ nhanh** được đặt gần CPU dùng để quản lý phần cứng
* Do **Memory có dữ liệu lớn nên tốc độ sẽ chậm** nên họ sẽ cho dữ liệu truy xuất qua thằng Cache thay vì truy xuất qua thằng Memory
* Ví dụ như thay vì mỗi lần Processor muốn học tổ chức máy tính thì phải ra thằng Memory mua dữ liệu làm bài tập mà mỗi lần vậy rất mệt nên nó mới nghĩ ra cách xây dựng một cái kệ sách tên Cache và đi mua dữ liệu trước cho cái kệ đó để mốt chỉ cần học thì lấy dữ liệu trên cái kệ Cache đó thôi là được

⇒ Phải có quá trình ánh xạ thông tin đưa vào trong Cache thì Processor mới truy xuất được

1. *Công thức tính thời gian truy xuất trung bình của bộ vi xử lý*

* Khi CPU cần đọc 1 ô nhớ trong bộ nhớ chính thì sẽ kiểm tra xem có trong Cache hay chưa
  + Nếu đã có(Cache Hit) thì đọc nội dung trong Cache mà không cần truy xuất bộ nhớ chính(Memory)
  + Nếu chưa có(Cache Miss) thì chép khối nhớ chứa ô nhớ cần đọc từ bộ nhớ chính vào Cache rùi vào CPU.Thời gian để xử lí Cache Miss gọi là Miss Penalty.

Hit là khi Processor truy xuất vô thằng Cache mà có dữ liệu đó sẵn trong Cache trong đó được chia ra 2 loại:

* Hit rate:Tỉ lệ hit
* Hit time:Thời gian truy xuất vô trong cache mà có thông tin

⇒Nguyên lý hoạt động của Cache:

\* Miss là khi Processor truy xuất vô thằng Cache mà không có dữ liệu đó sẵn trong Cache trong đó được chia ra 2 loại:

* Miss rate(Tỷ lệ miss)=1-Hit rate
* Miss penalty(Thời gian mà truy xuất dữ liệu không có trong cache)=**Thời gian thay thế khối dữ liệu**+**Thời gian di chuyển dữ liệu**
* **Thời gian truy xuất trung bình**=(Hit rate\*Hit time)+(Miss rate\*Miss penalty)

Trong mọi trường hợp:Hit Time<Miss Penalty

\* Ví dụ bài toán: Một bộ vi xử lý có tỉ lệ trật cache (cache miss) là 65%**(Miss rate)**. Thời gian truy xuất bộ nhớ cache là 2 ns**(Hit time)**, thời gian truy xuất bộ nhớ chính là 18 ns.Thời gian truy xuất trung bình của bộ vi xử lý này là:

* Hit rate=1-Miss Rate=1-0,65=0.35
* Miss Penalty=18ns+2ns=20ns
* **Thời gian truy xuất trung bình=0.35**2+20**0.65=13.7ns**

## *IV.Tìm hiểu kỹ về Cache*

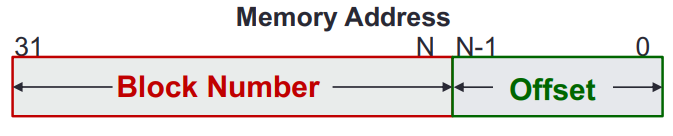
### 1.Các loại Cache Miss(Các lý do Cache Miss)

* Compusory/Cold miss:Lần đầu truy xuất bộ nhớ mà không có
* Conflict Miss:Có nhiều Memory Block cùng hướng về 1 block của Cache
* Capacity Miss:Bị thiếu bộ nhớ của Cache

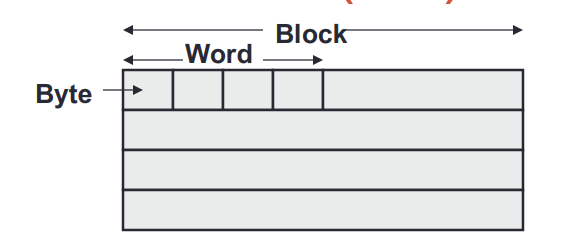
### 2.Cache Block/Cache Line

* Cache Block/Line:Đơn vị dữ liệu di chuyển giữa bộ nhớ và Cache
* Block có thể là có 1 hoặc nhiều hơn 1 word
* Ví dụ:
  + 16-byte block(1 block có 16 byte)~4-byte block(1 block có 4 byte)
  + 32-byte block(1 block có 32 byte)~8-byte block(1 block có 8 byte)

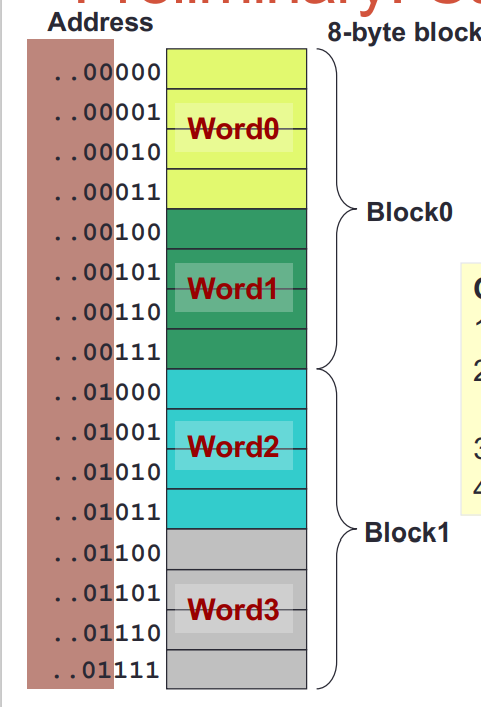
\* Hình ảnh về 1 ô địa chỉ bộ nhớ:



\*Hình ảnh quy định về các cách gọi trong Block number:



\* Ví dụ một mặt cắt của 2 ô block:



⇒Trong 1 bộ nhớ có dung lượng lớn chia làm nhiều block nên mỗi block có 1 số thứ tự(Block Number) và trong các block lại chứa nhiều byte nên mỗi byte cần có 1 số thứ tự để phân biệt(Byte Offset) và số đó được phân vào khu Offset

⇒Muốn có thông tin vào trong Cache thì có tổng cộng 3 phương pháp ánh xạ

### *3.Phương pháp ánh xạ trực tiếp-Ánh xạ cố định(Direct Mapping)*

A.Khái niệm cơ bản

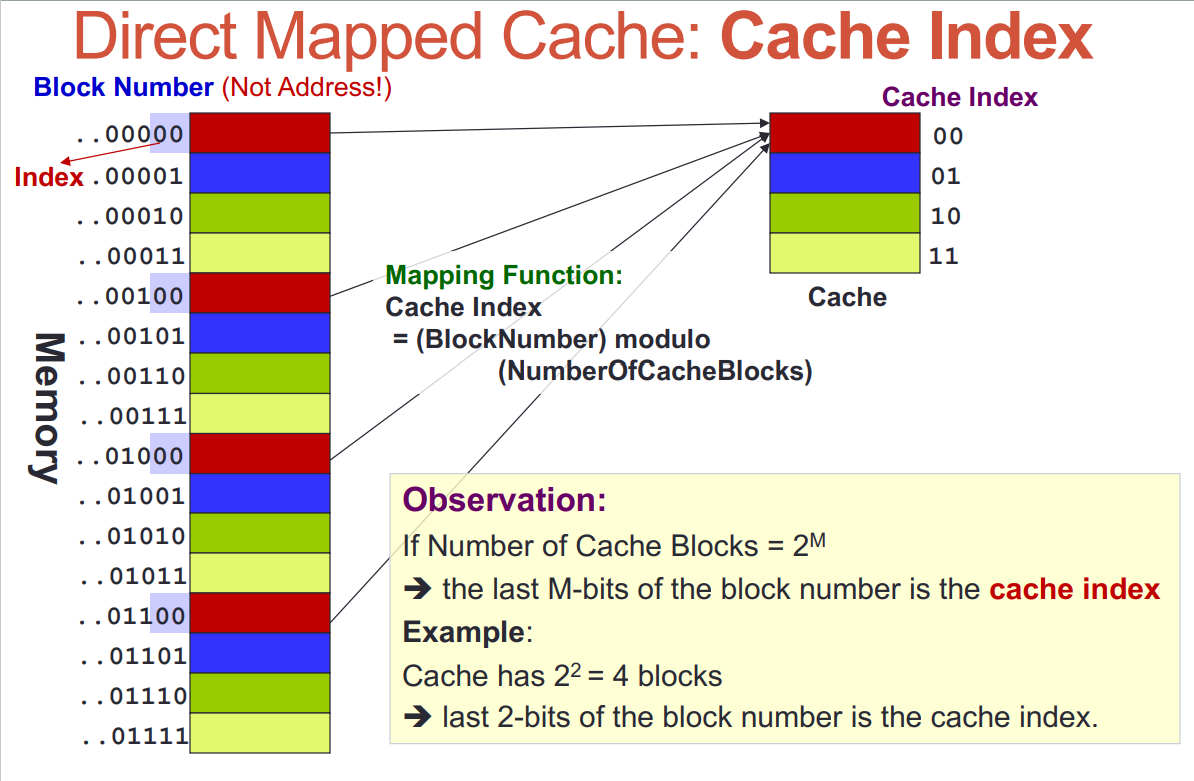
**Cache Block:**Là các block trong Cache Memory Block:Là các block trong Memory

**Cache Index** là thứ để giúp mình phân biệt block trong Cache

**Block number** là mã số để phân biệt các block bên Memory

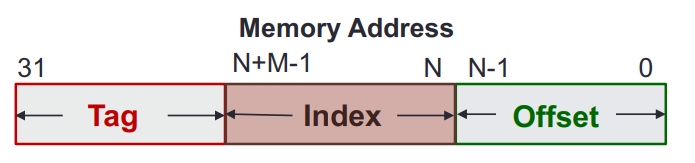
Nếu số block trong Cache là **2n** thì **n bit cuối** của block number là số bits Cache Index

\*Hình ảnh minh họa



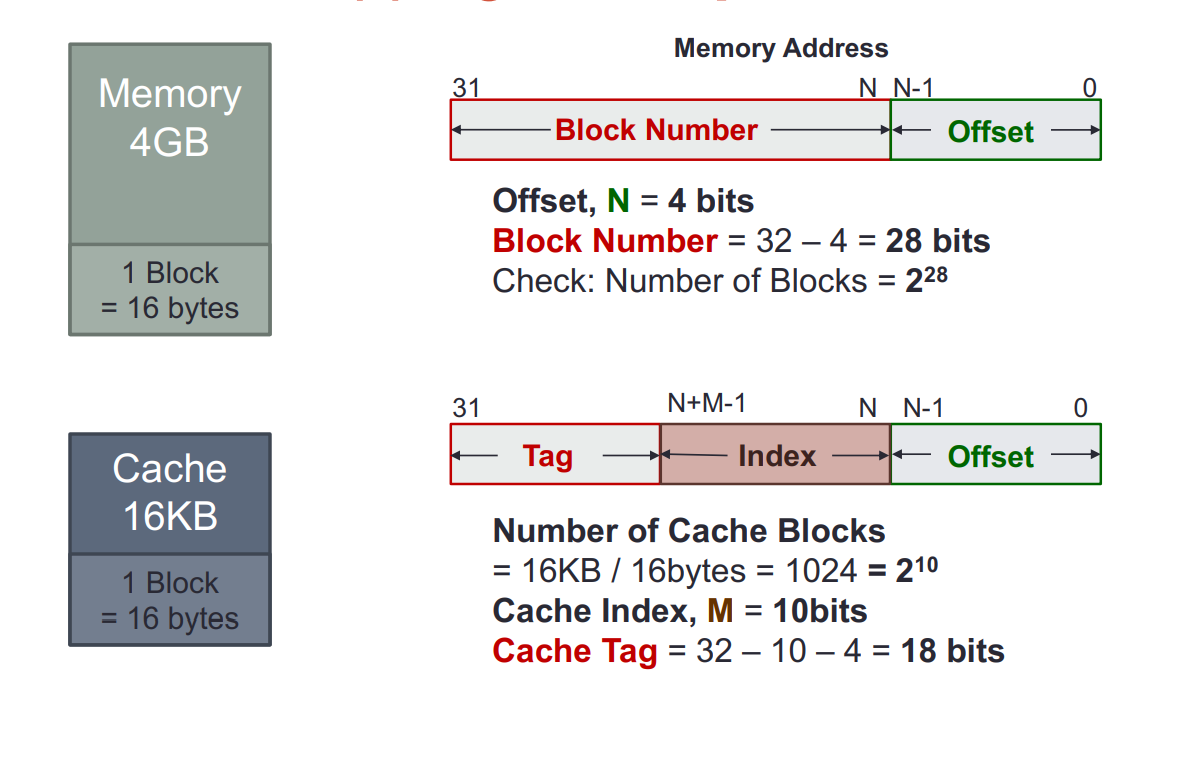
⇒Nguyên lý: tách block number thành Index và Tag

\*Hình ảnh tổng kết



⇒ Có 2 cách tính số lượng block trong Cache và Memory(Ví dụ dưới sẽ rõ) + Đầu tiên tính số bits của Block number=32-số bits Offset rồi lấy Số lượng block=2^Block Number + Thứ hai là lấy số bits tổng của Cache hoặc Memory chia cho số bits của từng block bên trong đó(Lưu ý nhớ cùng đơn vị rùi mới chia)

\* Ví dụ cho hình ảnh tổng kết:

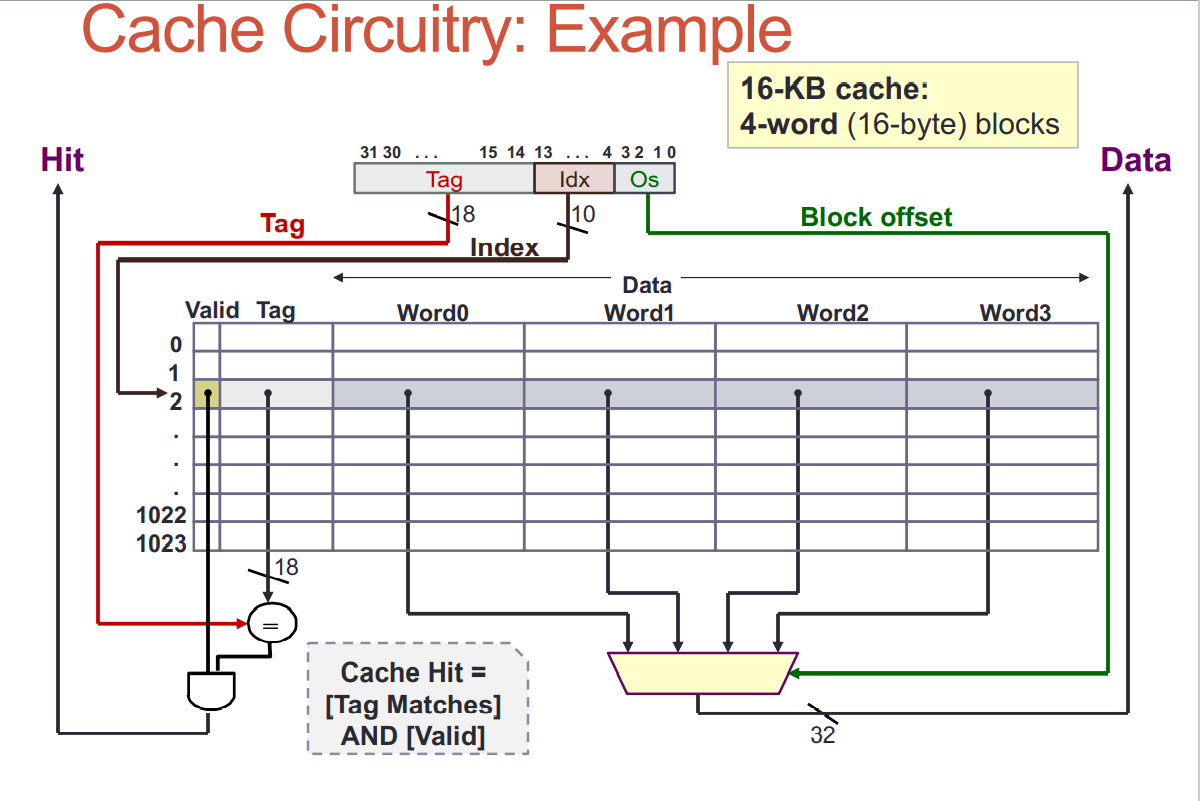


* Trong Memory
  + Offset=4 bits do 1 block có 16 bytes là 2^4 byte
  + Block number=32-offset=32-4=28 bits
  + Số lượng Block:2^28
* Trong Cache:
  + Offset=4 bits do 1 block có 16 bytes là 2^4 byte
  + Số lượng block trong Cache=16kb/16 bytes=1024=2^10
  + Index=10 bits
  + Số lượng tag=32-10-4=18 bits

B.Cấu trúc của Cache

* Ngoài đi kèm với data block thì Cache còn có
  + Tag
  + Valid bit chỉ ra cái line đó có chứa dữ liệu đó có hợp lệ không
* Điều kiện để Cache hit:
  + Valid bit=1(True)
  + Tag Index=Tag memory address

\* Hình ảnh minh họa:



C.Cơ chế truy xuất dữ liệu từ Cache mới hoàn toàn ra Processor(Không cần thuộc chỉ cần hiểu là được)

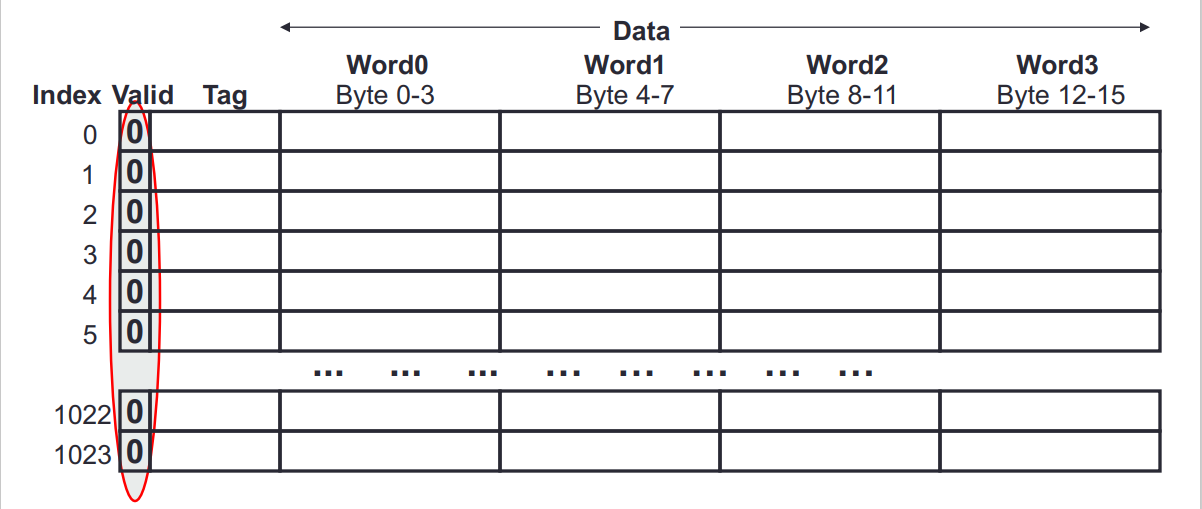
Ví dụ cho 1 cache 16kb với mỗi 1 block tương ứng 16 byte

**Bước 1**:Tính số block trong Cache(Lưu ý cùng đơn vị để chia cho đúng)

* + 16\*1024/16=1024 block

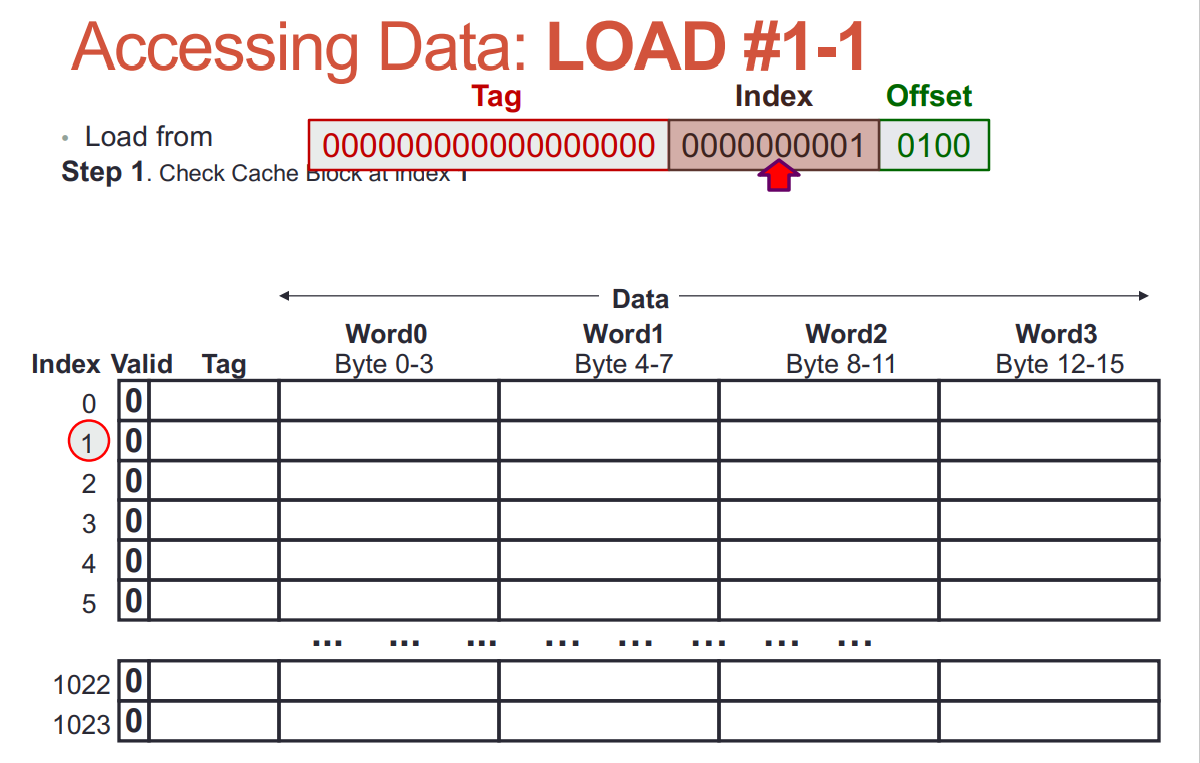
**Bước 2**:Thiết lập Cache trống(Các word theo các bits trong Cache,đánh số trong khoảng [0,số block-1] cho index,đặt các valid là 0)

\* Hình ảnh minh họa



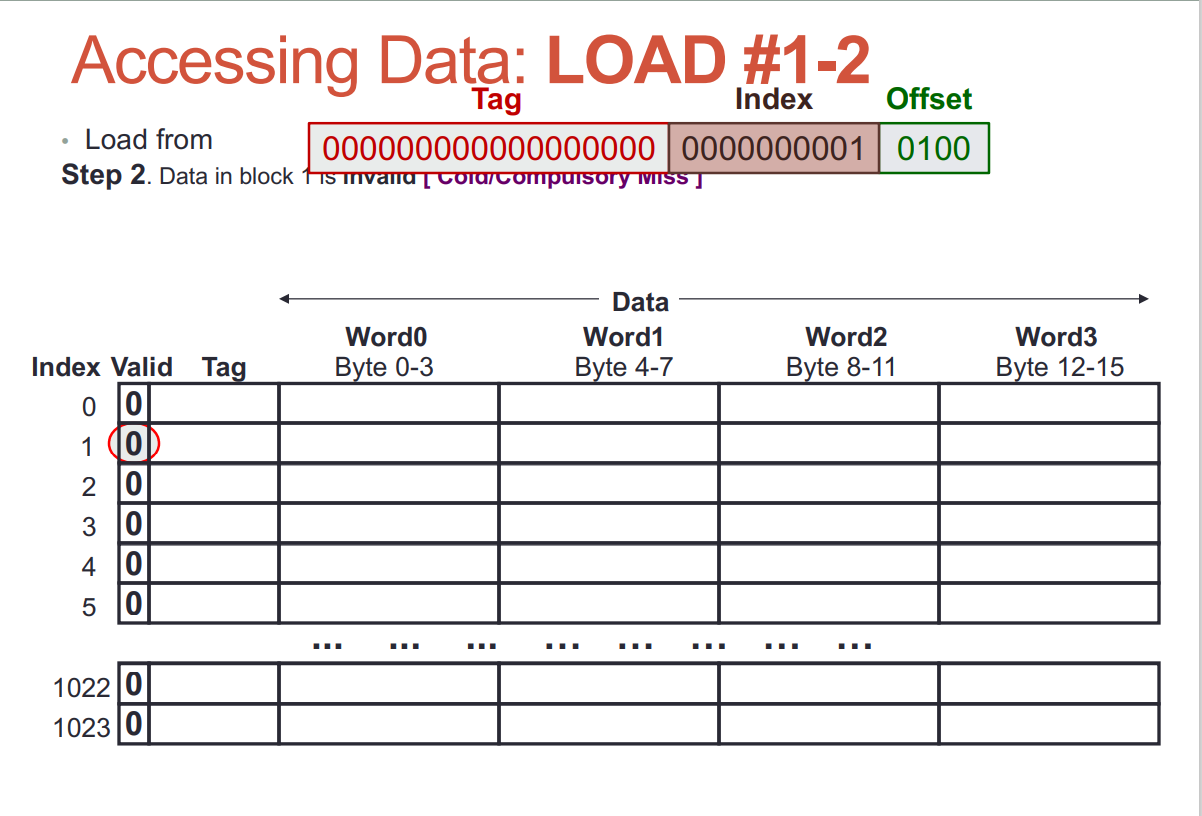
**Bước 3**:Kiểm tra Index đầu tiên để biết coi dữ liệu ở block nào

\* Hình ảnh minh họa



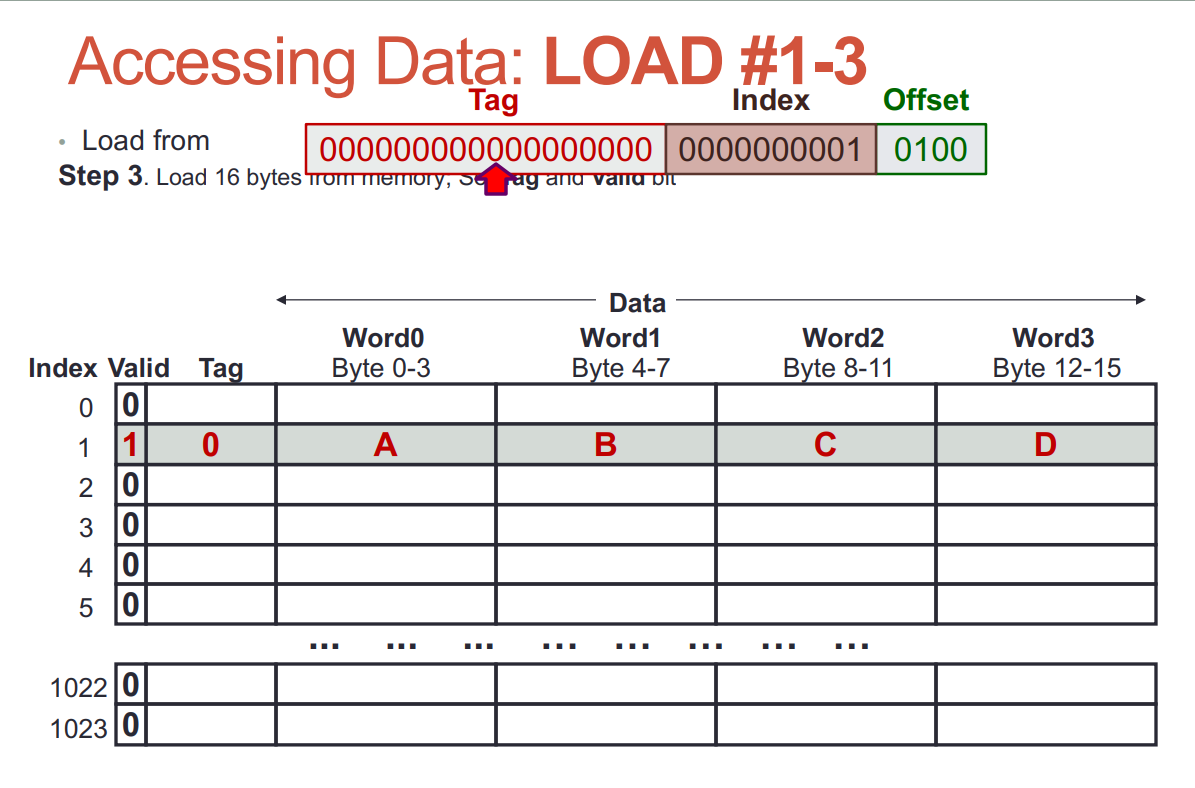
**Bước 4**:Kiểm tra dữ liệu xem trong Cache có không(Nếu không có thì Miss)

\* Hình ảnh minh họa



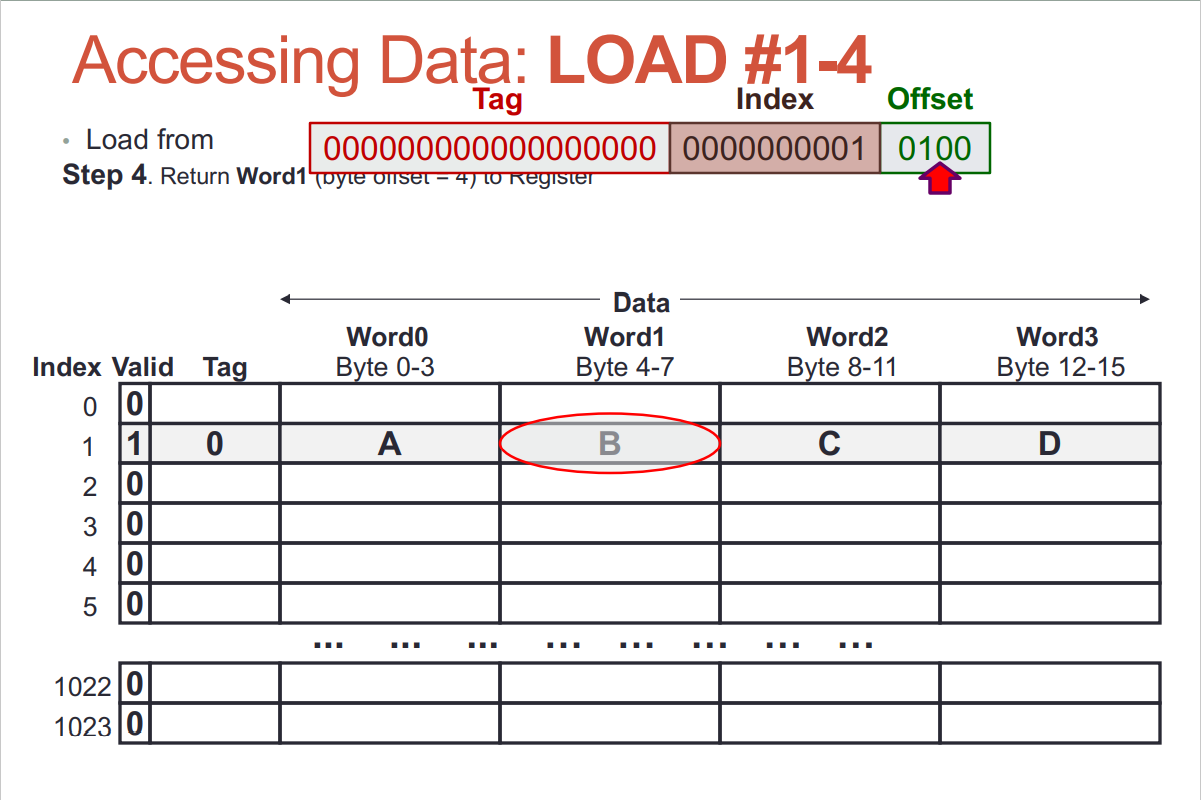
**Bước 5**:Load thông tin từ Memory vô Cache nếu không có và set valid là 1 và set tag theo tag của dữ liệu(Nhớ đổi về thập phân cho ngắn nhen) và ghi word là dữ liệu từ memory(Trong trường hợp Cache Miss còn Cache hit thì bỏ qua bước này luôn)

\* Hình ảnh minh họa



**Bước 6**:Tìm word theo offset

\* hình ảnh minh họa

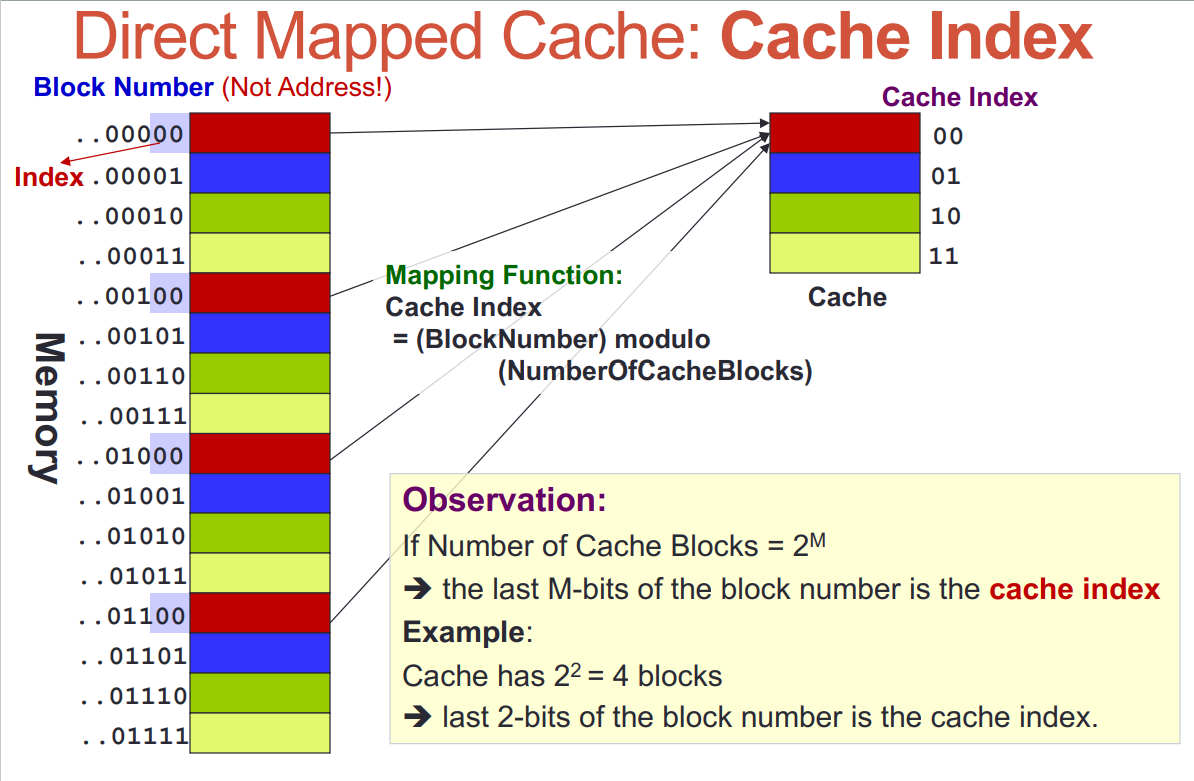


nếu cùng index mà khác tag thì vẫn được gọi Cache Miss và lúc này sẽ thực hiện bước 5 thay thế dữ liệu cũ thành dữ liệu mới

D.Đặc điểm

* Mỗi block của một bộ nhớ chính chỉ có có thể nạp vào 1 line duy nhất của Cache

\* hình ảnh minh họa



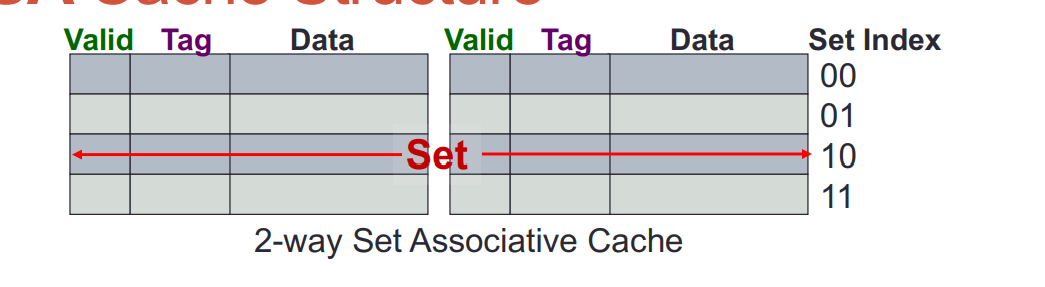
E.Ưu/nhược điểm

* Ưu điểm
  + Thiết kế đơn giản và nhanh do không tốn nhiều thời gian truy tìm địa chỉ ô nhớ trong Cache
* Nhược điểm
  + Dễ gây xung đột(Do mỗi ô cứ chỉa vào 1 line) và hiệu quả tận dụng không gian Cache không cao và hệ số hit(Hit rate) thấp

### *4.Phương pháp ánh xạ tập kết hợp(Set Associate Cache)*

A.Đặc điểm

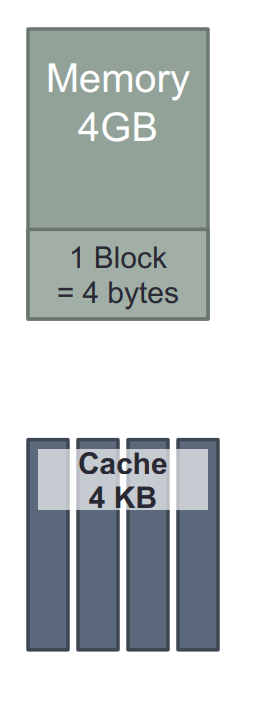
* Cache bao gồm 1 số set
* Mỗi set chứa N block
* Mỗi block trong bộ nhớ ánh xạ tới 1 set duy nhất trong Cache
* Trong set đó,memory block có thể đặt ở bất cứ đâu

\* hình ảnh minh họa 

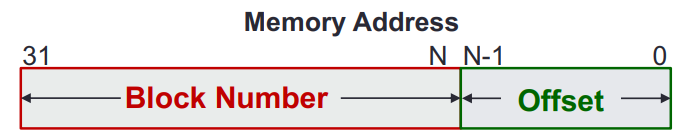
1. Khái niệm cơ bản



\* Ví dụ bài toán:



- Phần giải Memory:

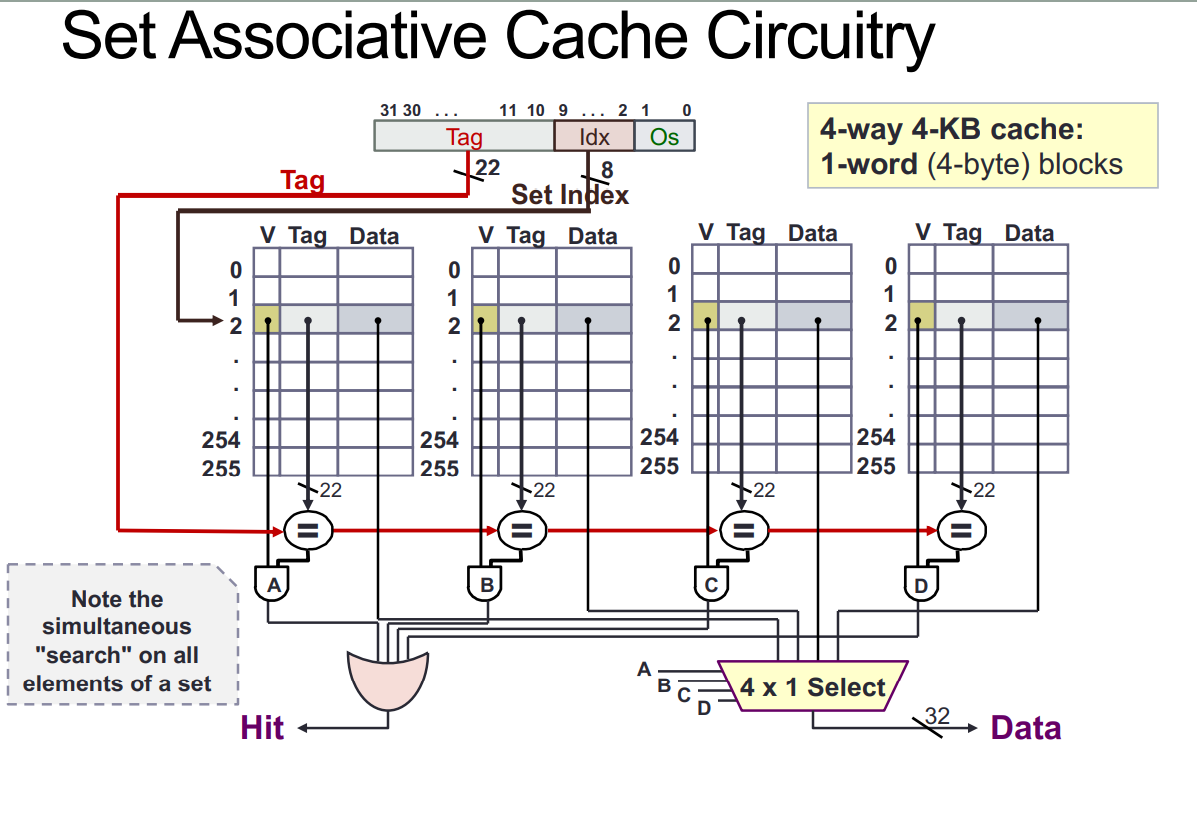


* Offset=2 bits do 1 block có 4 bytes là 2^2 byte
* Block number=32-offset=32-2=30 bits
* Số lượng Block:2^30
* Phần giải Cache:



* Offset=2 bits do 1 block có 4 bytes là 2^2 byte
* Số lượng Block trong Cache=4GB/4KB=1024=2^10
* Do thanh Cache là 4-way set nên số set là 1024/4=256=2^8
* Set index=8 bits
* Số lượng tag=32-8-2=22 bits

1. Cấu trúc của Cache



* Điều kiện hit(Cơ chế hoạt động của các cổng AND y hệt sơ đồ trên nên để tránh đề cương dài thêm thì mạn phép không soạn phần đó :< mà chỉ soạn phần sau):
  + Chỉ cần 1 cổng AND A,B,C,D có output là 1 thì sẽ hit⇒1 block hit thì Cache sẽ hit

D.Ưu/nhược điểm

* Ưu điểm:
  + Mềm dẻo,ít xung đột và phân bố không gian Cache đồng đều hơn và đạt hệ số hit cao hơn
* Nhược điểm
  + Độ phức tạp rất cao

E.Ví dụ về tính toán

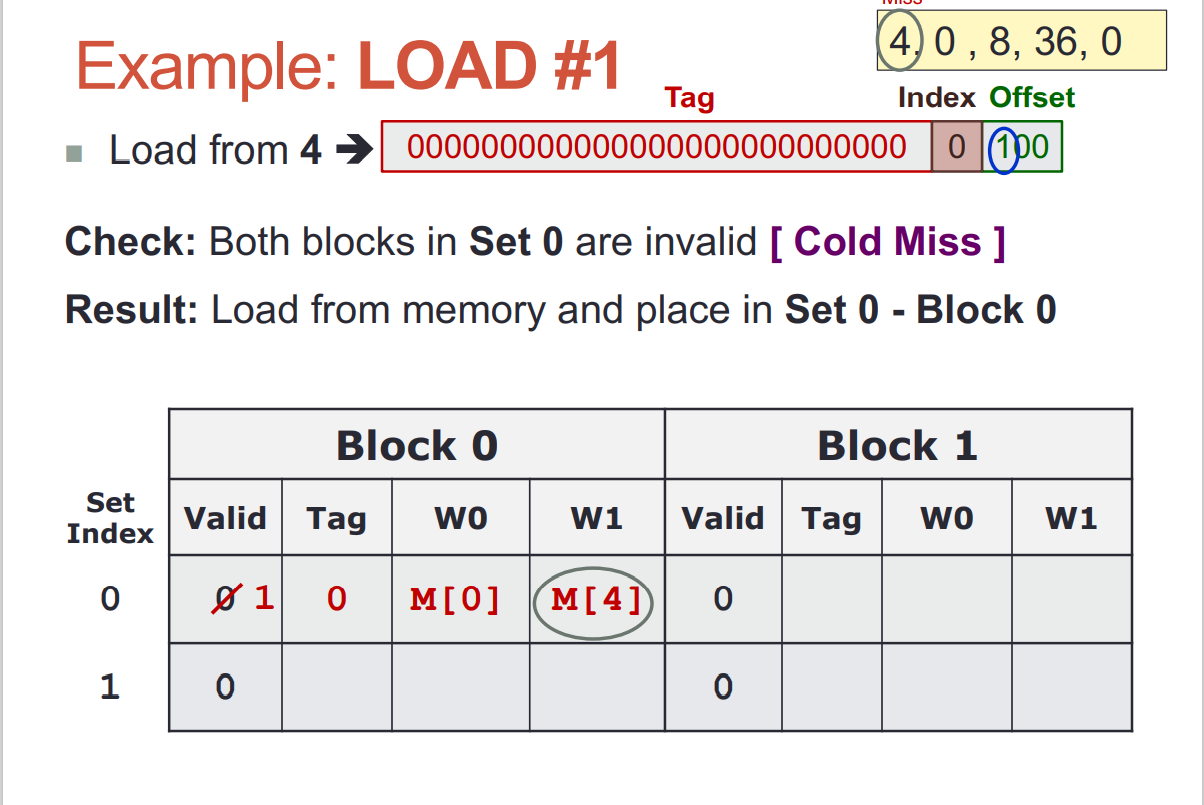
* Bài toán:
  + Cho địa chỉ lần lượt là 4, 0, 8, 36, 0
  + Cho Cache là 2-way set với tổng cộng là 4 block và mỗi block tương đương 8 byte
  + Tìm tỉ lệ hit/Miss
* Phân tích
  + Cho Cache là 2-way set với tổng cộng là 4 block và mỗi block tương đương 8 byte⇒2-way set thì mỗi set có 2 block⇒Tổng cộng cache có 4 set
* Giải:
  + Ta có 8=2^3⇒Offset có N=3 bits
  + Hình vẽ ta có được



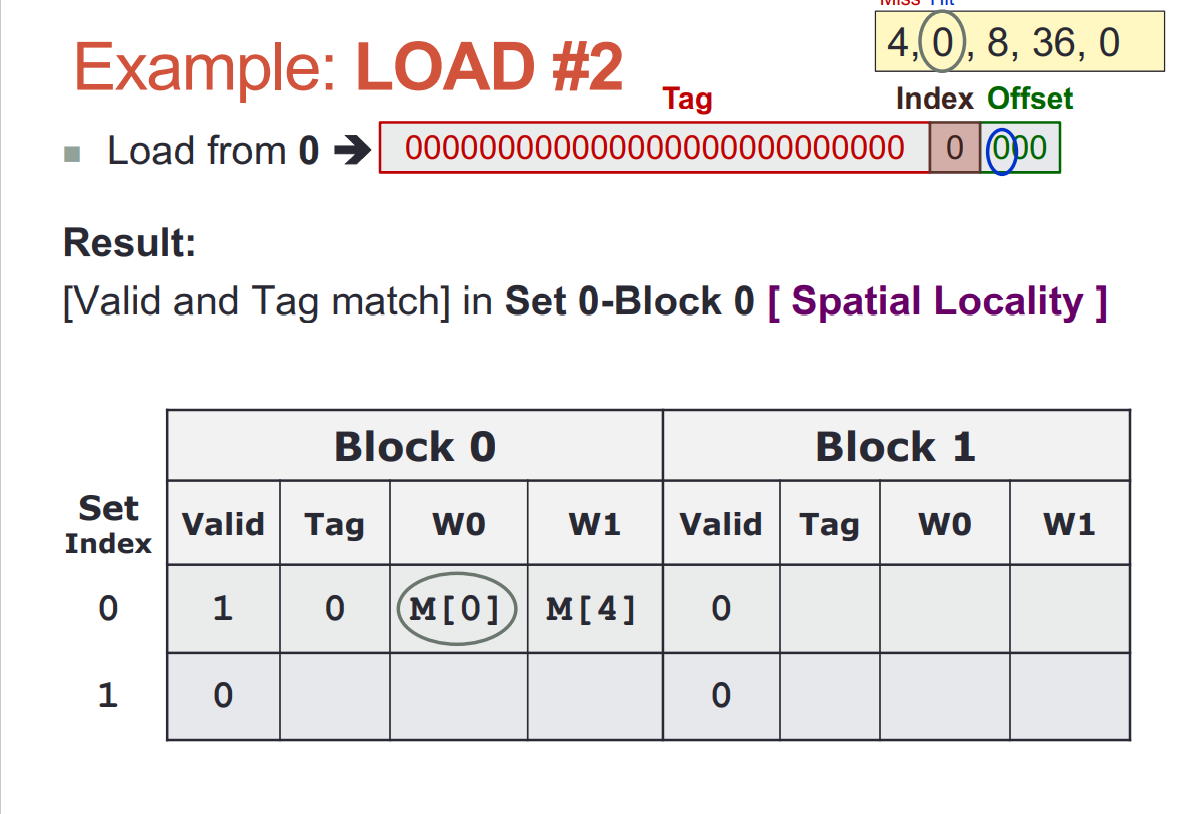
* Kiểm tra:

nếu cùng index mà khác tag thì vẫn được gọi Cache Miss và lúc này sẽ ghi lên block kế bên nếu block kế bên nó còn

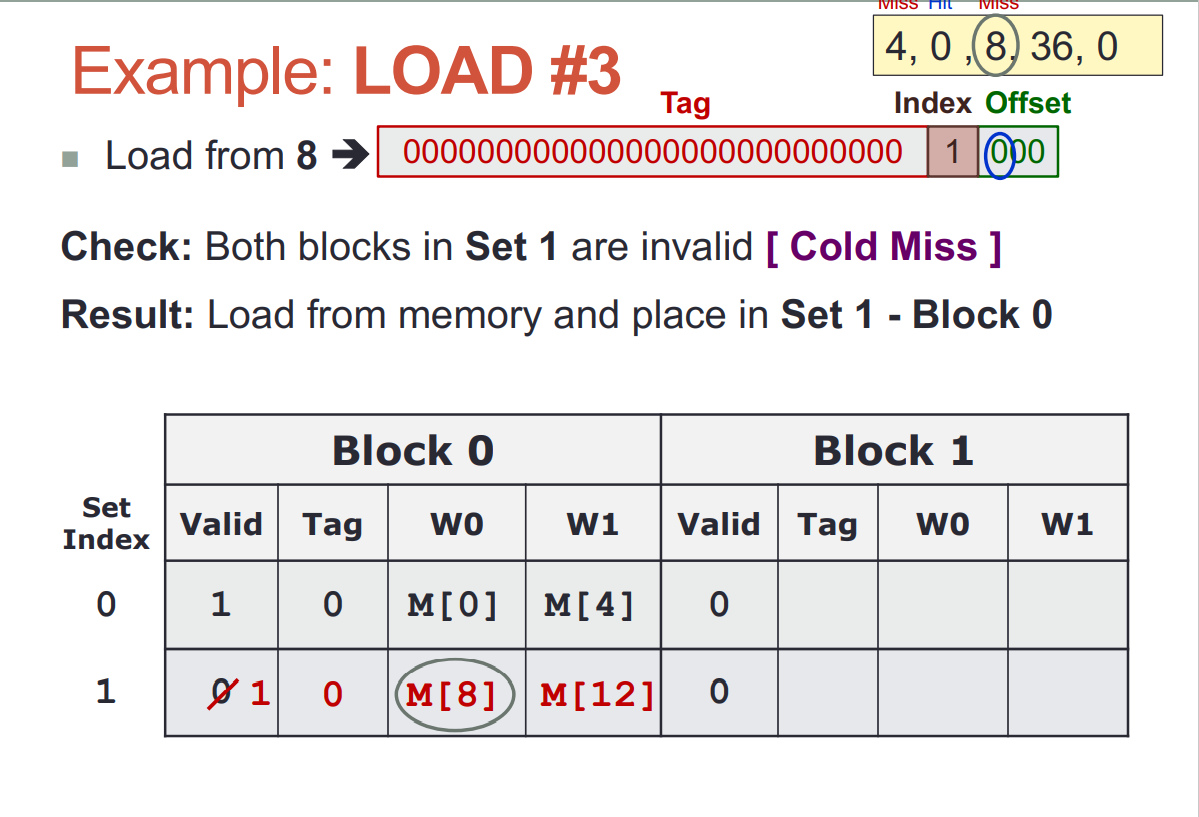
\* lần 1



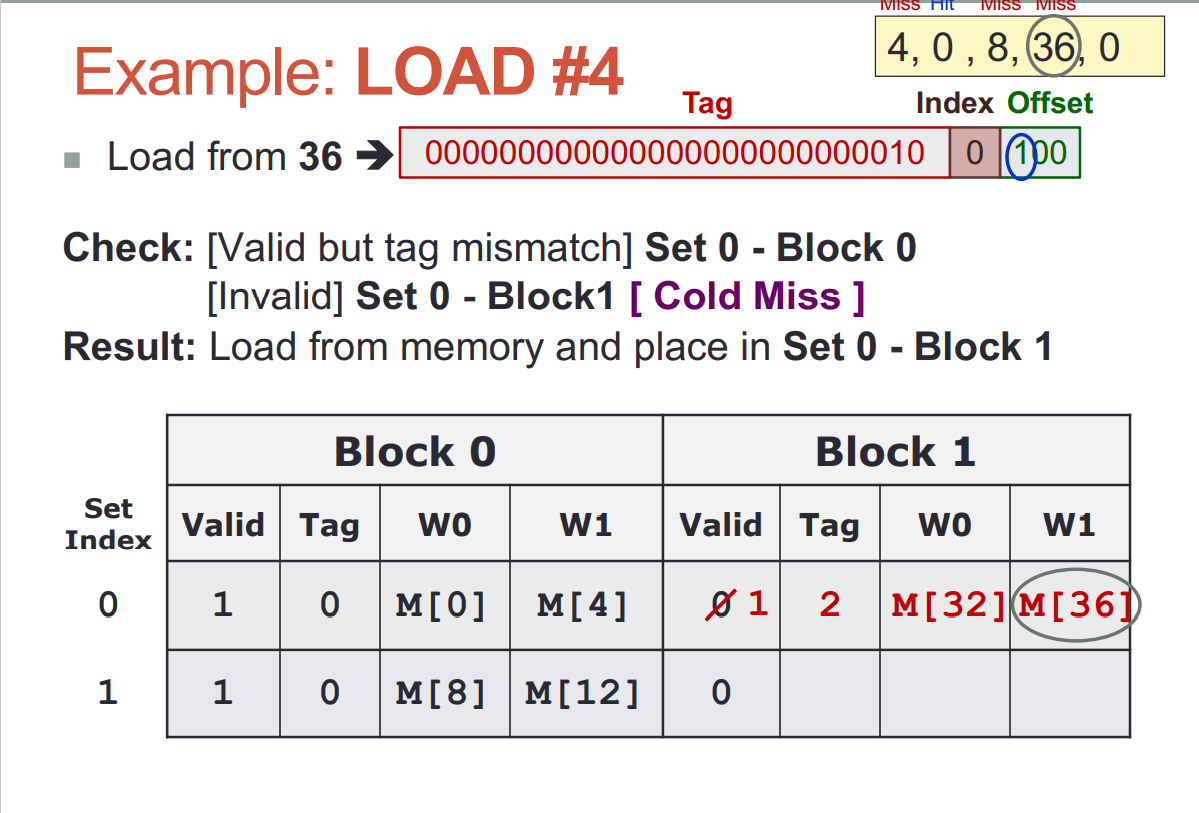
\* Lần 2



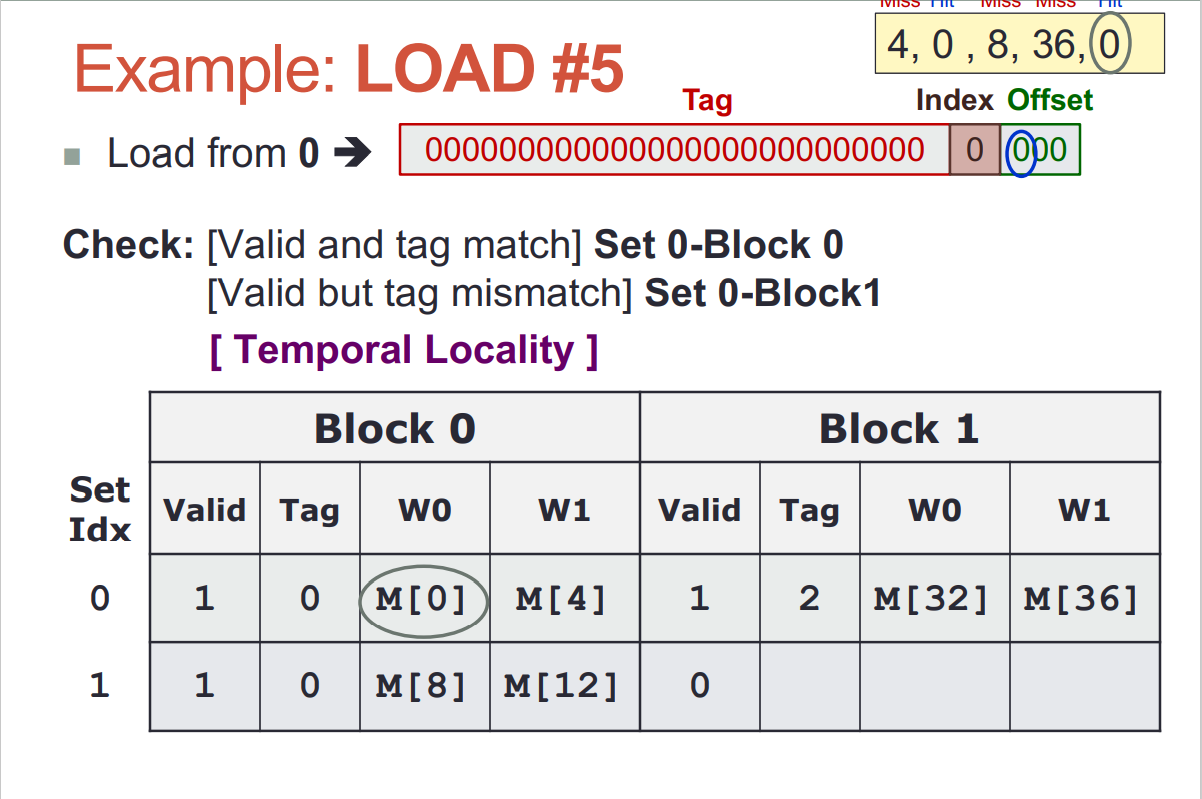
\* Lần 3



\* Lần 4



\* Lần 5



⇒Ví dụ về Temporal Locality vì số 0 đã được dùng trước ở trường hợp 2

### *5.Phương pháp ánh xạ kết hợp toàn phần(Full Associative hay viết tắt là FA)*

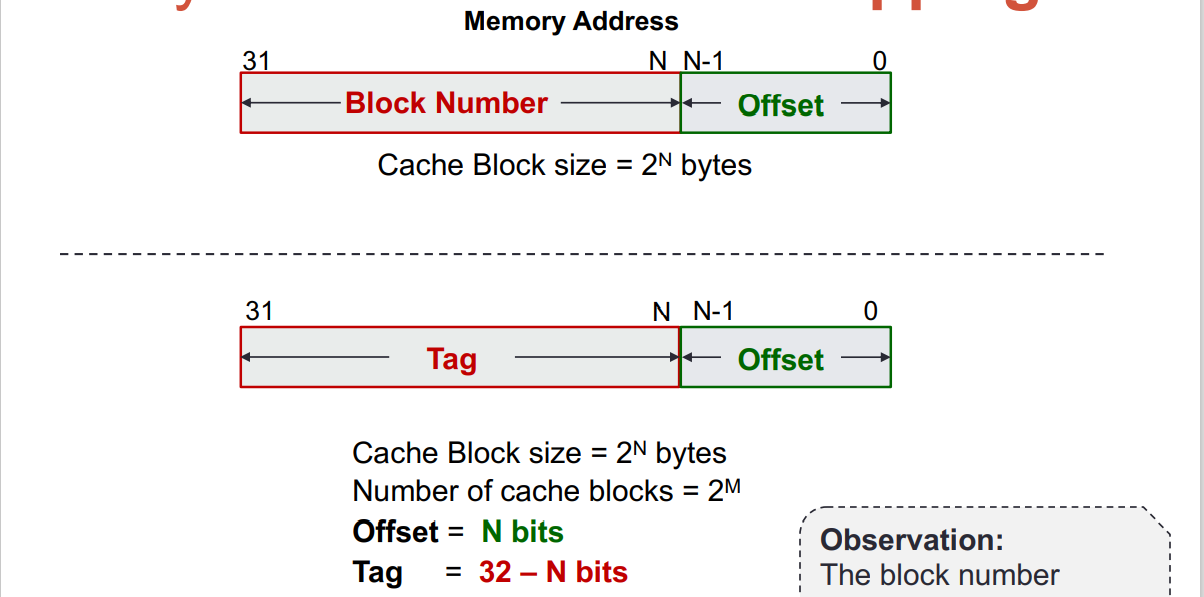
A.Đặc điểm

* Mỗi block trong bộ nhớ chính có thể được ánh xạ vào bất kỳ line nào trong Cache

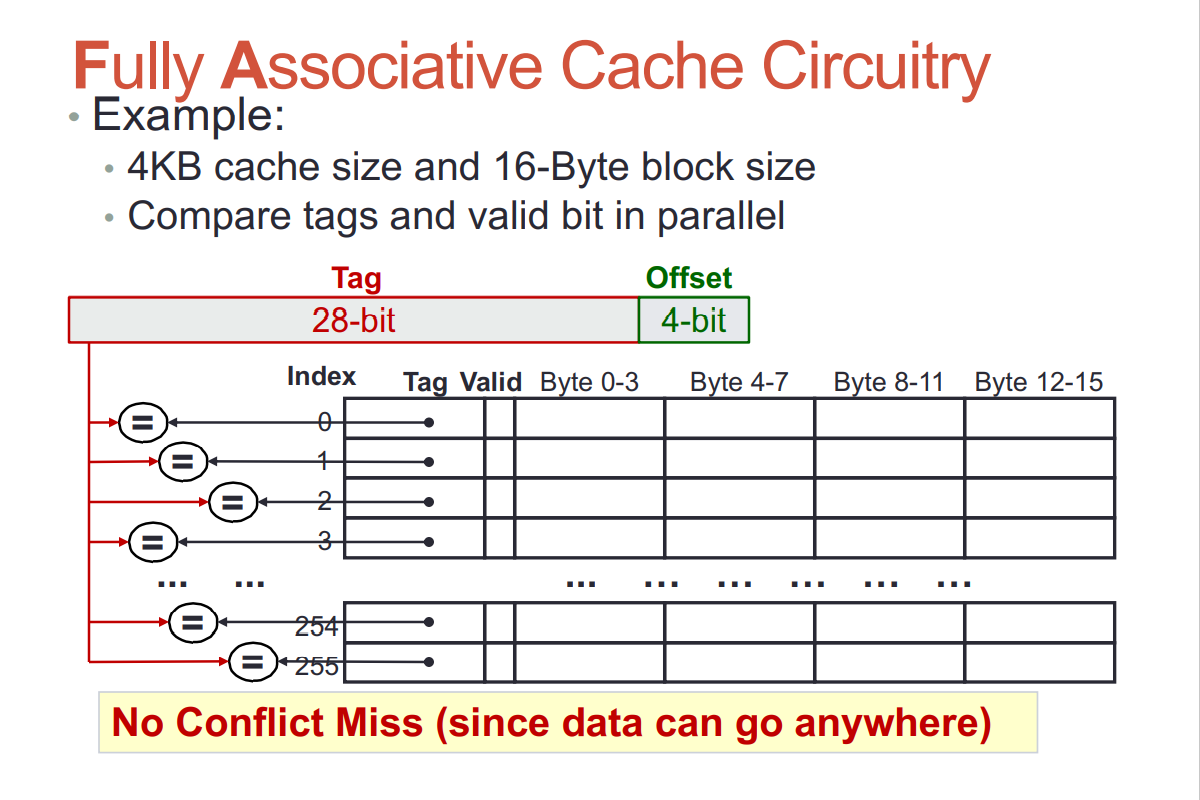
B.Ưu điểm/Nhược điểm

* Ưu điểm:
  + Mềm dẻo,ít xung đột và hiệu suất sử dụng không gian Cache cao hơn và có khả năng cho hệ số hit cao hơn
* Nhược điểm
  + Thiết kế phức tạp nên chỉ thích hợp thiết kế Cache có dung lượng nhỏ

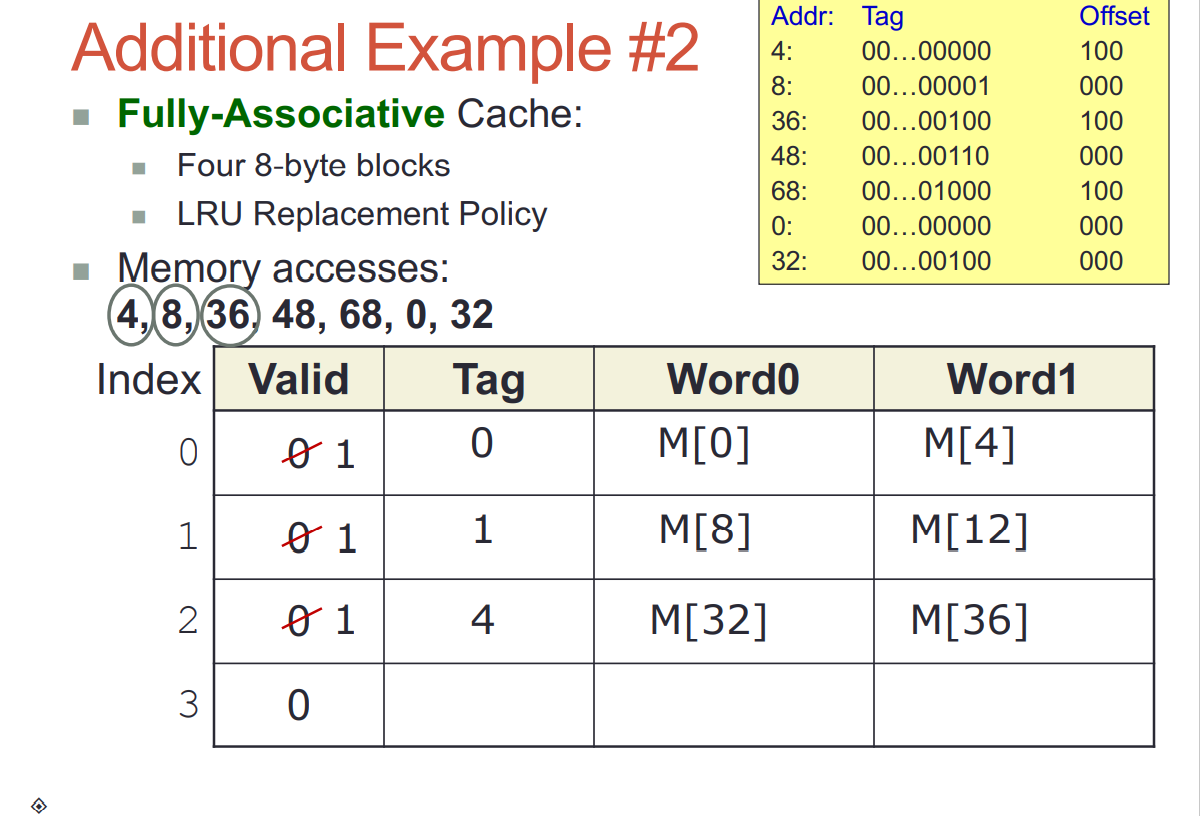
C.Khái niệm cơ bản



1. Cấu trúc của Cache



1. Cơ chế truy xuất dữ liệu từ Cache mới hoàn toàn ra Processor(Không cần thuộc chỉ cần hiểu là được)



\* Ở số 4

* Kiểm tra từ dữ liệu index 0
* Do Cache không có dữ liệu ở index 0 sẽ load dữ liệu từ Mem
* Return word 1 do offset là 4

\* ở số 8

* Kiểm tra từ dữ liệu index 0
* Do Cache ở index 1 không có sẽ load dữ liệu từ Mem
* Return word 0 do offset là 0

\* ở số 36

* Kiểm tra từ dữ liệu index 0
* Do Cache ở index 2 không có sẽ load dữ liệu từ Mem
* Return word 1 do offset là 4

## *V.Mở rộng*

### *1.GHI DỮ LIỆU KHI CACHE HIT*

* Ghi xuyên qua (Write throught):
  + Ghi cả cache và bộ nhớ chính

⇒Tốc độ chậm

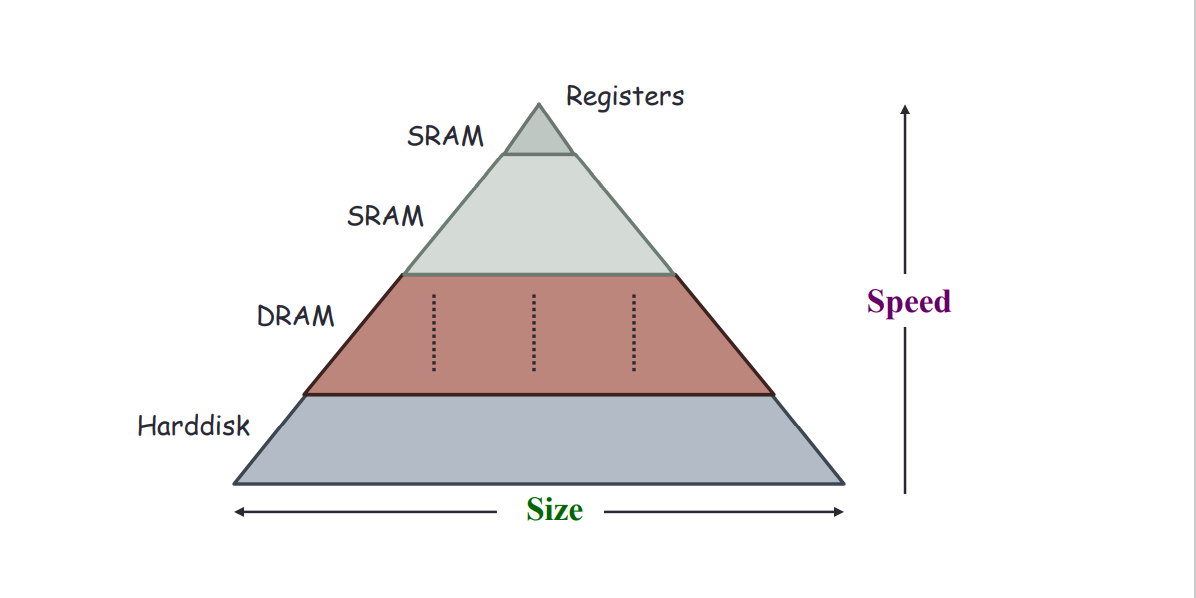
* Ghi trả sau (Write back):
  + Chỉ ghi ra cache

⇒Tốc độ nhanh

### *2.Chức năng của Cache Level 1,Cache Level 2*

* Khi CPU xử lý, Cache Level 1 sẽ tiến hành kiểm tra Cache Level 2 xem có dữ liệu mình cần không trước khi truy cập vào bộ nhớ hệ thống

### *3.Tốc độ truy cập bộ nhớ của các loại bộ nhớ*



### *4.Cache cấu tạo từ loại bộ nhớ nào?*

Cache cấu tạo từ bộ nhớ SRAM