Chương 4. Bộ nhớ cache

4.1 Tổng quan về bộ nhớ máy tính

4.2 Nguyên lý của bộ nhớ cache

4.3 Các thành phần trong thiết kế bộ nhớ cache

4.4 Tổ chức cache của Pentium 4

4.5 Tổ chức cache trong ARM

Một số khái niệm

- Từ (word): đơn vị "tự nhiên" của bộ nhớ. Kích thước từ thường bằng số bit biểu diễn một số nguyên và kích thước lệnh. Intel x86 có kích thước từ là 32b.
- Đơn vị đánh địa chỉ: ở các hệ thống khác nhau, đơn vị đánh địa chỉ có thể là byte hoặc word. Trong bất cứ trường hợp nào, mối quan hệ giữa số lượng các đơn vị đánh địa chỉ N và số bit địa chỉ A là $2^A = N$
- Đơn vị truyền:
  - Với bộ nhớ chính, đơn vị truyền bằng số lượng các bit được gửi đến hoặc đi từ bộ nhớ.
  - Với bộ nhớ ngoài, đơn vị truyền thường lớn hơn rất nhiều, thường được gọi là các khối (block)

Ví dụ

1. VXL Intel x86-32b, kết nối bus (32 đường địa chỉ, 16 đường dữ liệu) với bộ nhớ tổ chức dưới dạng các ngăn nhớ 16b. Hãy cho biết:

a. Kích thước word của BN trên

b. Dung lượng tối đa của bộ nhớ mà VXL có thể quản lý được.

c. Đơn vị truyền của BN trên. Để thực hiện một lệnh: cộng 2 số (trong bộ nhớ) và ghi kết quả vào 1 ngăn nhớ khác thì VXL sẽ phải thực hiện bao nhiêu thao tác đọc, ghi BN

4.1 Tổng quan về bộ nhớ máy tính

Phân loại bộ nhớ máy tính

Vị trí Bên trong (vd: thanh ghi, cache, bộ nhớ chính Bên ngoài (vd: đĩa quang, đĩa từ, băng từ) **Dung lượng** Số lượng từ Số lượng byte **Đơn vị truyền** Từ Khối **Phương pháp truy cập** Tuần tự Trực tiếp Ngẫu nhiên Kết hợp	**Hiệu suất** Thời gian truy cập Chu kỳ xung nhịp Tốc độ truyền tải **Loại vật lý** Bán dẫn Từ Quang học Quang từ **Tính chất vật lý** Điện động/điện tĩnh (Dữ liệu có bị mất khi mất điện) Có thể xóa/không xóa được **Tổ chức** Module bộ nhớ

Phân loại bộ nhớ

a. **Vị trí**
   - Bộ nhớ có thể ở trong và ngoài máy tính
   - Bộ nhớ chính là bộ nhớ trong
   - Bộ xử lý cần có bộ nhớ cục bộ riêng của nó: thanh ghi
   - Cache là một dạng khác của bộ nhớ trong
   - Bộ nhớ ngoài bao gồm các thiết bị lưu trữ ngoại vi có thể truy cập vào bộ xử lý thông qua bộ điều khiển I/O

b. **Dung lượng**
   - Bộ nhớ thường được biểu diễn dưới dạng byte

c. **Đơn vị truyền**
   - Đối với bộ nhớ trong, đơn vị truyền bằng số lượng đường điện đi vào và ra khỏi module bộ nhớ

Phân loại bộ nhớ (tiếp)

d. Phương pháp truy cập các khối dữ liệu

Truy cập tuần tự	Truy cập trực tiếp	Truy cập ngẫu nhiên	Kết hợp
• Bộ nhớ được tổ chức thành các đơn vị dữ liệu được gọi là bản ghi (record) • Truy cập được thực hiện tuần tự • Thời gian truy cập biến đổi • Ví dụ: băng từ	• Có một cơ chế đọc-ghi chia sẻ • Mỗi khối hoặc bản ghi có một địa chỉ duy nhất dựa trên vị trí vật lý • Thời gian truy cập biến đổi • Ví dụ: đĩa từ	• Mỗi vị trí trong bộ nhớ có một cơ chế định địa chỉ riêng • Thời gian truy cập vào một vị trí nhất định không đổi và phụ thuộc vào chuỗi các truy cập trước đó • Một vị trí bất kỳ có thể được chọn ngẫu nhiên, định địa chỉ và truy cập trực tiếp • Ví dụ: bộ nhớ chính và một số bộ nhớ cache	• Một word được truy xuất dựa trên một phần nội dung thay vì địa chỉ của nó • Mỗi vị trí có cơ chế định địa chỉ riêng. Thời gian truy xuất là không đổi, phụ thuộc vào vị trí hoặc các truy cập trước đó • Bộ nhớ Cache có thể sử dụng truy cập kết hợp

e. Hiệu năng

Hai đặc điểm quan trọng nhất của bộ nhớ: dung lượng và hiệu năng

Ba tham số hiệu năng được sử dụng:

Thời gian truy cập (độ trễ)

- Đối với bộ nhớ truy cập ngẫu nhiên, nó là thời gian cần để thực hiện 1 thao tác đọc hoặc ghi
- Đối với bộ nhớ truy cập không ngẫu nhiên, nó là thời gian cần để đặt cơ chế đọc-ghi vào vị trí mong muốn

Chu kỳ bộ nhớ

- Với bộ nhớ truy cập ngẫu nhiên: Thời gian truy cập cộng với thời gian cần trước khi truy cập thứ hai có thể bắt đầu
- Có thể cần thêm thời gian để các transients chết trên đường tín hiệu hoặc để khôi phục lại dữ liệu bị hỏng
- Liên quan đến hệ thống bus, không liên quan bộ xử lý

Tốc độ truyền tải

- Tốc độ truyền dữ liệu vào hoặc ra khỏi bộ nhớ
- Đối với bộ nhớ truy cập ngẫu nhiên, tốc độ truyền tải bằng 1/(chu kỳ)

f. Đặc tính vật lý của bộ nhớ

- Các dạng phổ biến nhất là: Bộ nhớ bán dẫn, Bộ nhớ bề mặt từ, Bộ nhớ quang, Bộ nhớ quang từ

- Một số đặc điểm vật lý quan trọng:

1. Đặc điểm lưu trữ dữ liệu

- Bộ nhớ điện động (Volatile memory): thông tin bị suy yếu hoặc bị mất khi nguồn điện tắt

  VD: RAM, Cache

- Bộ nhớ điện tĩnh (Non-volatile memory): thông tin một khi đã được ghi thì sẽ không bị mất trừ khi cố tình thay đổi kể cả không có nguồn cung cấp

  VD: ROM, USB, HDD,…

2. Công nghệ sản xuất:

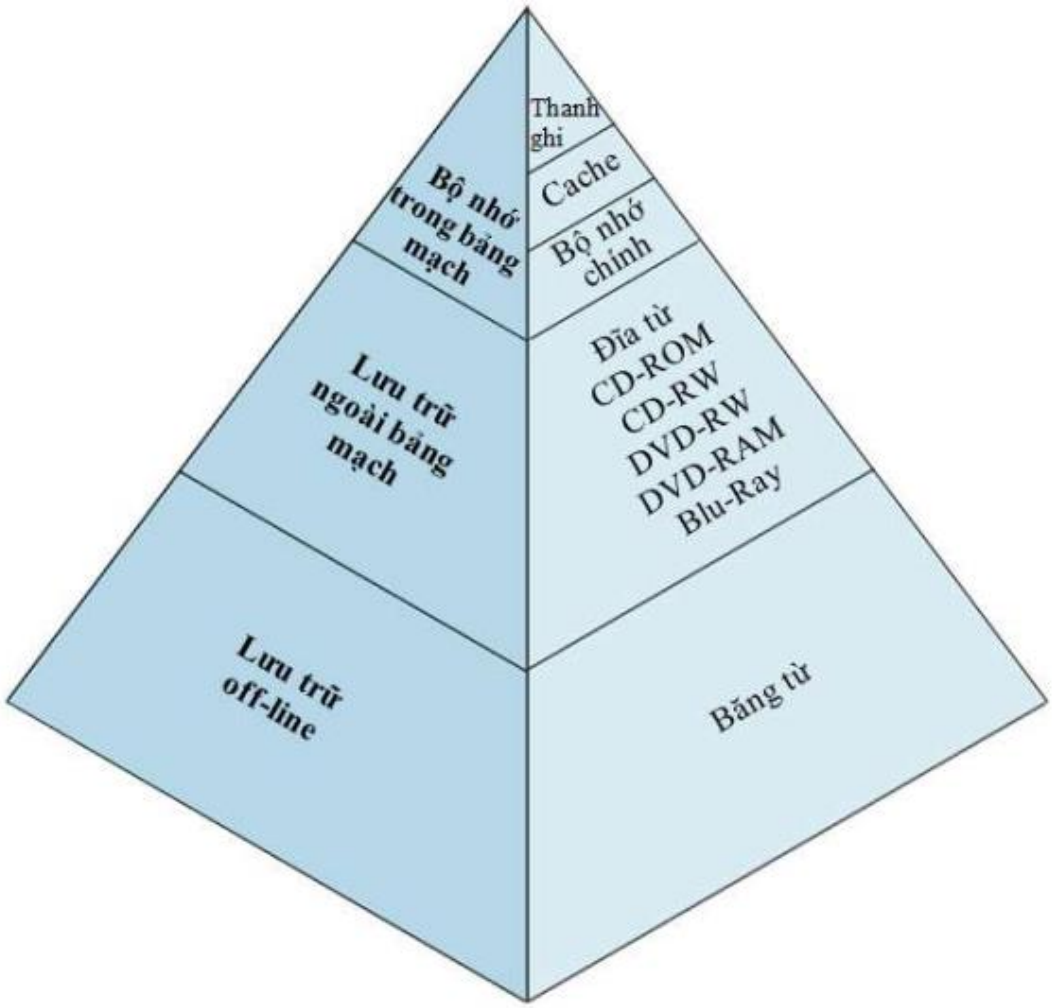
- Bộ nhớ bề mặt từ (Magnetic-surface memories): HDD, Tape
- Bộ nhớ bán dẫn (Semiconductor memory): RAM, ROM, Cache,…
- Bộ nhớ không xoá được (Nonerasable memory): Không thể thay đổi, trừ khi phá hủy các khối lưu trữ. VD: ROM

g. Tổ chức bộ nhớ: mô hình phân cấp bộ nhớ

- Thiết kế bộ nhớ của máy tính cần trả lời ba câu hỏi:
  - How much? How fast? How expensive?
- Cần có sự cân đối giữa dung lượng, thời gian truy cập và chi phí
  - Thời gian truy cập nhanh hơn, chi phí lớn hơn cho mỗi bit
  - Dung lượng lớn hơn, chi phí nhỏ hơn cho mỗi bit
  - Dung lượng lớn hơn, thời gian truy cập chậm hơn
- Giải pháp:
  - Không dựa hoàn toàn vào một thành phần hoặc công nghệ bộ nhớ
  - Sử dụng một hệ thống phân cấp bộ nhớ

Bộ nhớ phân cấp - Sơ đồ

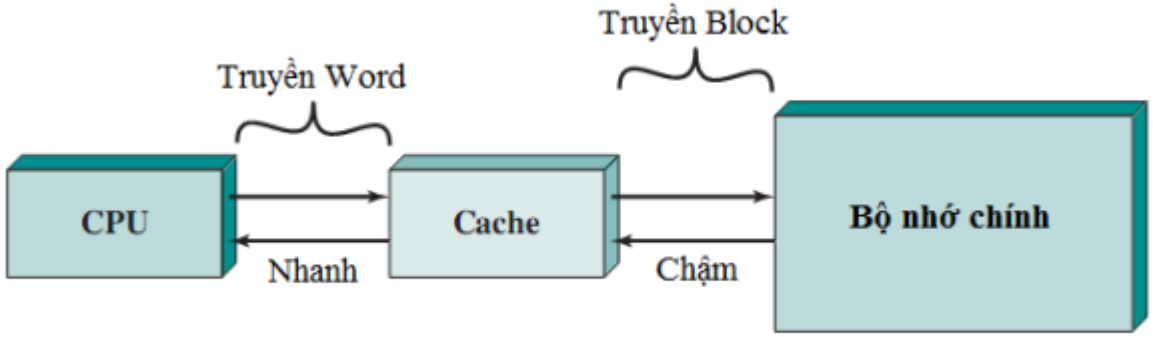
- **Chi phí trên bit giảm**
- **Dung lượng tăng**
- **Thời gian truy cập tăng**
- **Tần suất truy cập bộ nhớ của VXL giảm**



4.2. Nguyên lý bộ nhớ cache

Bộ nhớ cache và bộ nhớ chính

- BXL truy cập xuất lệnh/dữ liệu từ BN chính theo đơn vị byte hoặc word → tốc độ chậm (do tốc độ BN chính, bus chậm hơn VXL)
- Bộ nhớ cache được thiết kế để cải thiện thời gian truy cập bộ nhớ:
  - Dựa vào tính cục bộ của dữ liệu và lệnh lưu trữ trong BN chính
  - BN cache có tốc độ cao nhưng dung lượng thấp hơn bộ nhớ chính
  - Bộ nhớ cache chứa **bản sao** của một phần của bộ nhớ chính.

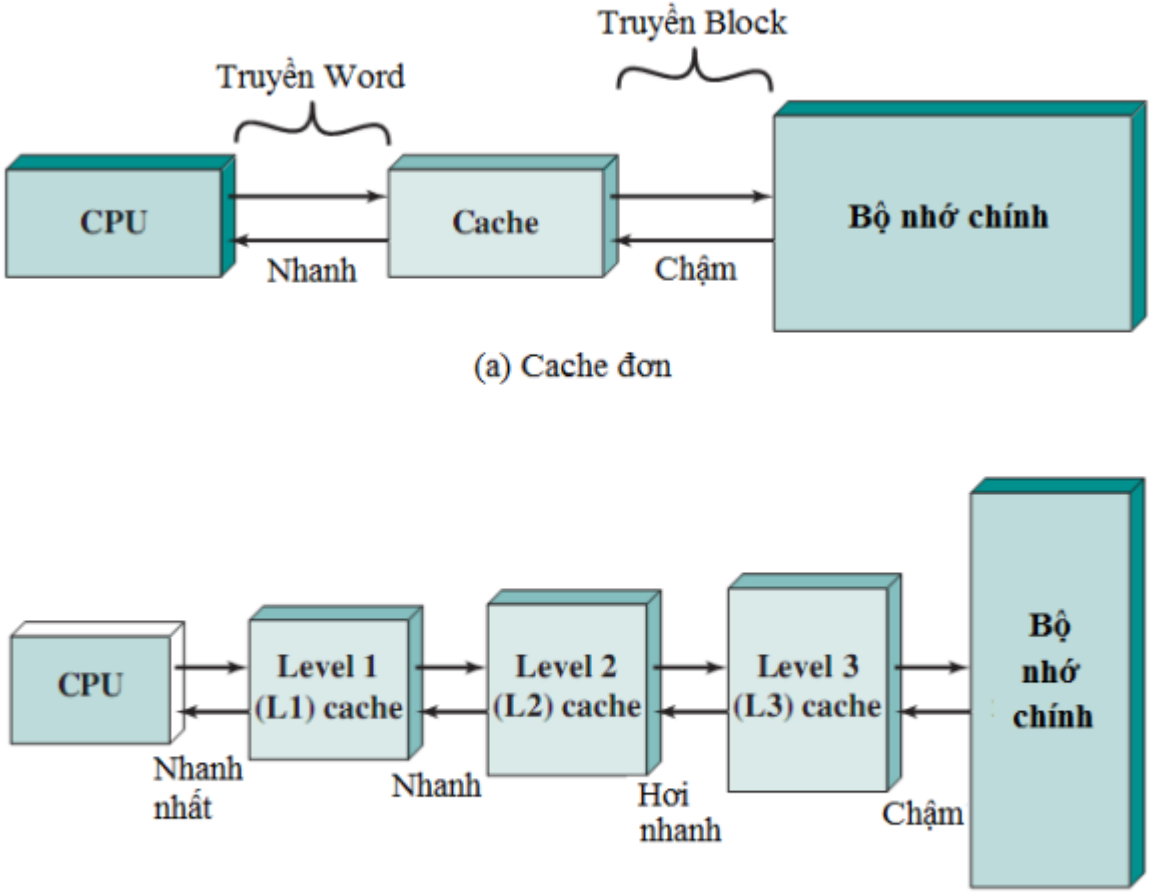


(a) Cache đơn

Nguyên lý

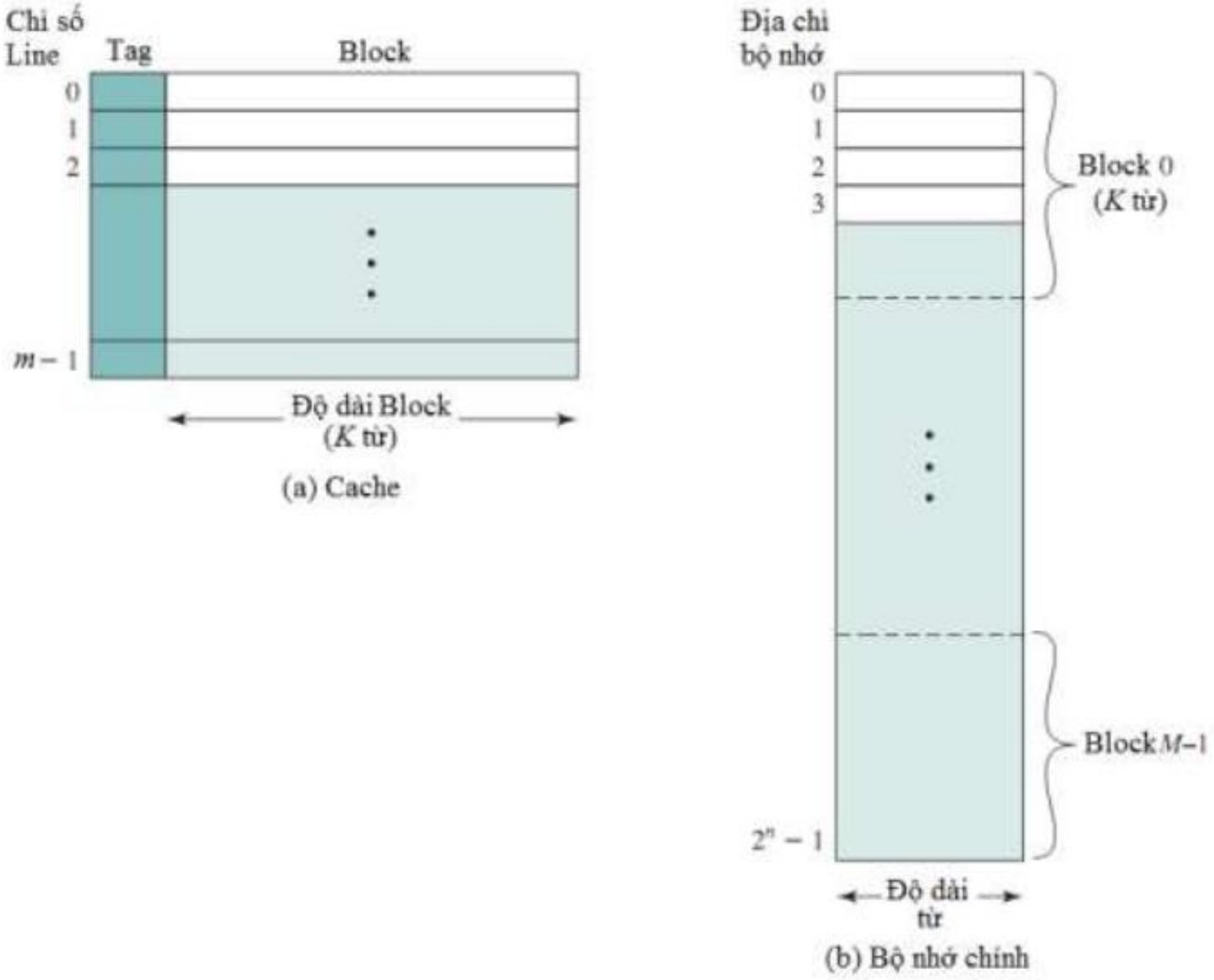
- BN chính gồm 2^n ***từ nhớ (word)*** được đánh địa chỉ: n bit địa chỉ
- BN chính được chia thành các ***khối (block)*** có kích thước cố định: ***K word***.
  - Như vậy, BN chính có $\frac{2^n}{K} = \boldsymbol{M}$ **khối**
- BN cache được chia thành các ***đường (line)***, mỗi đường có ***K word.***
- Mỗi **block** của BN chính được ánh xạ vào một **line** của Cache
- Khi bộ xử lý muốn đọc một word của bộ nhớ nó sẽ kiểm tra xem word đó có nằm trong bộ nhớ cache hay không.
  - Nếu có: word này được gửi đến bộ vi xử lý.
  - Nếu không: một khối dữ liệu từ bộ nhớ chính (chứa từ mà VXL đang muốn truy cập), được đọc vào bộ nhớ cache và sau đó từ được gửi đến bộ VXL.

- Tổ chức cache

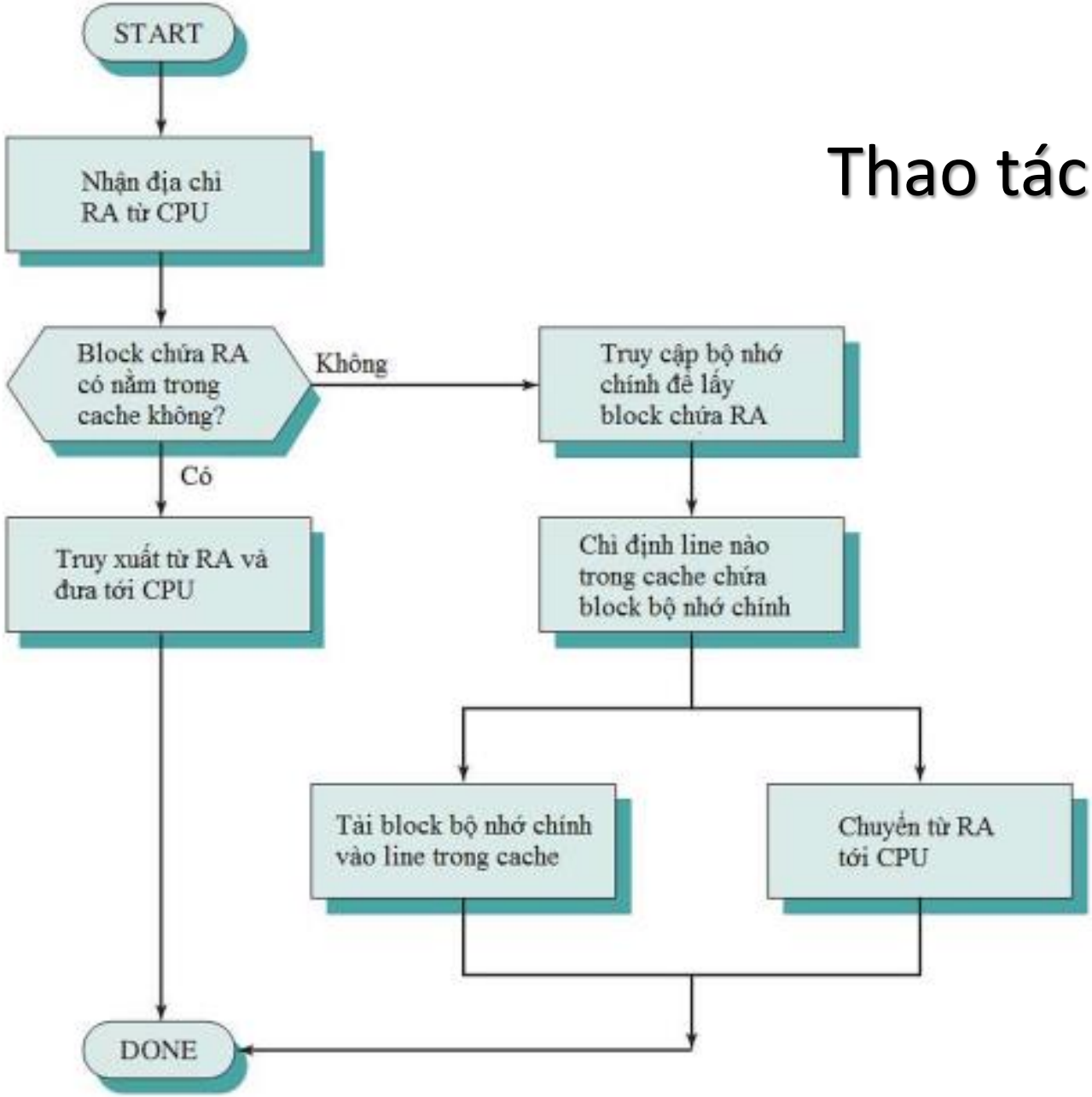


Hình 4.2 Cache và bộ nhớ chính

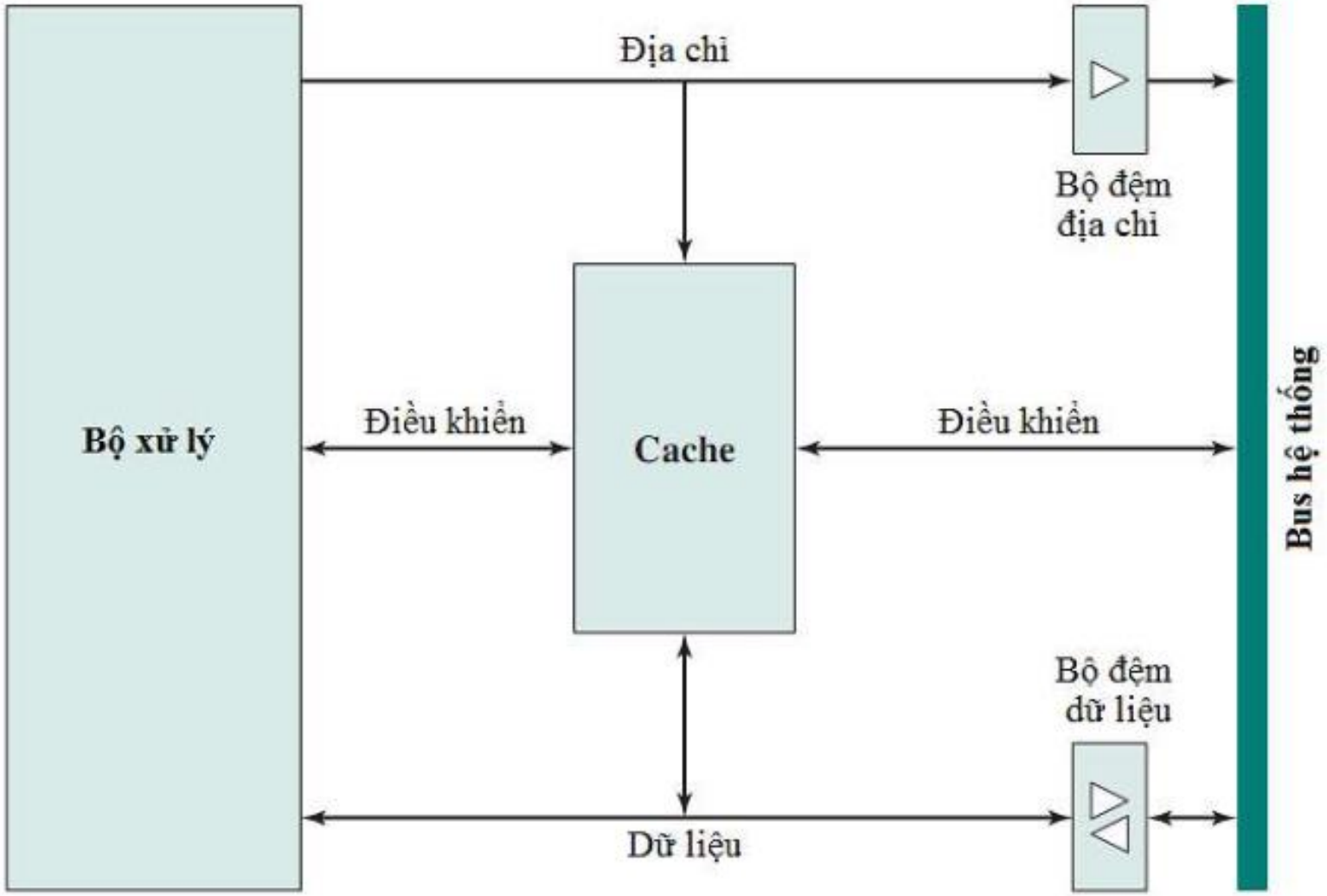
Cấu trúc bộ nhớ chính/cache



Thao tác Đọc Cache



Tổ chức bộ nhớ cache điển hình



4.3. Các yếu tố khi thiết kế Cache

a. Địa chỉ bộ nhớ cache
- Logic
- Vật lý

b. Kích thước bộ nhớ cache

c. Ánh xạ bộ nhớ
- Trực tiếp
- Kết hợp
- Tập kết hợp

d. Thuật toán thay thế
- Least recently used (LRU)
- First in first out (FIFO)
- Least frequently used (LFU)
- Random

e. Chính sách ghi
- Ghi xuôi
- Ghi ngược

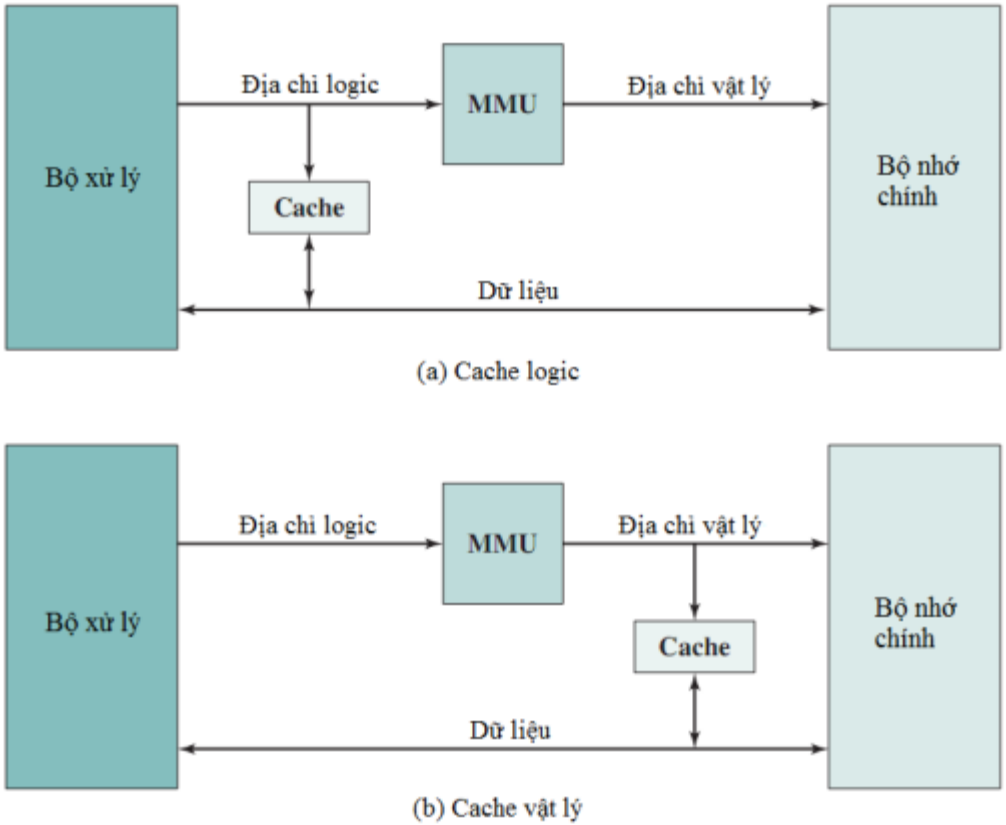
f. Kích thước line

g. Cache nhiều cấp
- Một hoặc hai cấp
- Thống nhất hoặc phân chia

a. Địa chỉ bộ nhớ cache

- Địa chỉ ảo: bộ xử lý hỗ trợ bộ nhớ ảo:
  - ✓Quản lý bộ nhớ thông qua địa chỉ logic
  - ✓Các trường địa chỉ trong lệnh là các địa chỉ ảo
  - ✓Để thực hiện các thao tác đọc/ghi vào bộ nhớ chính, khối quản lý bộ nhớ (MMU – Memory Management Unit) sẽ dịch từng địa chỉ ảo sang địa chỉ vật lý trong bộ nhớ chính

- Cache ảo (cache logic): bn cache đặt giữa BXL và MMU
  - ✓ Địa chỉ được sử dụng là địa chỉ ảo
- Cache vật lý: bn cache đặt giữa MMU và bộ nhớ chính
  - ✓ Địa chỉ được sử dụng là địa chỉ vật lý



b. Kích thước cache (cache size)

- Kích thước cache phải đủ nhỏ để không làm giá thành tăng cao
- Kích thước cache phải đủ lớn để giảm thời gian truy cập, tăng hiệu suất hệ thống
- Ngoài ra, kích thước cache quá lớn sẽ làm tăng số cổng để định địa chỉ cho các vị trí nhớ trong cache

  → giảm hiệu quả truy cập ngay cả khi cache nằm trong cùng chip hoặc board với VXL

Processor	Type	Year of Introduction	L1 Cache[a]	L2 cache	L3 Cache
IBM 360/85	Mainframe	1968	16 to 32 kB	—	—
PDP-11/70	Minicomputer	1975	1 kB	—	—
VAX 11/780	Minicomputer	1978	16 kB	—	—
IBM 3033	Mainframe	1978	64 kB	—	—
IBM 3090	Mainframe	1985	128 to 256 kB	—	—
Intel 80486	PC	1989	8 kB	—	—
Pentium	PC	1993	8 kB/8 kB	256 to 512 KB	—
PowerPC 601	PC	1993	32 kB	—	—
PowerPC 620	PC	1996	32 kB/32 kB	—	—
PowerPC G4	PC/server	1999	32 kB/32 kB	256 KB to 1 MB	2 MB
IBM S/390 G6	Mainframe	1999	256 kB	8 MB	—
Pentium 4	PC/server	2000	8 kB/8 kB	256 KB	—
IBM SP	High-end server/ supercomputer	2000	64 kB/32 kB	8 MB	—
CRAY MTA[b]	Supercomputer	2000	8 kB	2 MB	—
Itanium	PC/server	2001	16 kB/16 kB	96 KB	4 MB
Itanium 2	PC/server	2002	32 kB	256 KB	6 MB
IBM POWER5	High-end server	2003	64 kB	1.9 MB	36 MB
CRAY XD-1	Supercomputer	2004	64 kB/64 kB	1MB	—
IBM POWER6	PC/server	2007	64 kB/64 kB	4 MB	32 MB
IBM z10	Mainframe	2008	64 kB/128 kB	3 MB	24-48 MB
Intel Core i7 EE 990	Workstaton/ server	2011	6 × 32 kB/32 kB	1.5 MB	12 MB
IBM zEnterprise 196	Mainframe/ Server	2011	24 × 64 kB/ 128 kB	24 × 1.5 MB	24 MB L3 192 MB L4

b. Kích thước cache trong một số bộ xử lý

a, Hai giá trị cách nhau bằng dấu / là cache chỉ thị và cache dữ liệu.
b, Cả hai cache đều là cache chỉ thị; Không có cache dữ liệu.

c. Ánh xạ bộ nhớ

- Bởi vì số đường cache ít hơn số khối bộ nhớ chính, cần có một thuật toán ánh xạ các khối bộ nhớ chính vào các đường bộ nhớ cache
- Ba kỹ thuật có thể được sử dụng:

Trực tiếp

- Mỗi khối của bộ nhớ chính được ánh xạ vào một đường cache duy nhất
- Đơn giản nhất

Kết hợp

- Cho phép một khối nhớ chính được nạp vào bất kỳ đường cache nào
- Logic điều khiển cache diễn giải địa chỉ bộ nhớ bằng một trường Tag và trường Word
- Để xác định một khối có ở trong một cache không, logic điều khiển cache phải cùng lúc kiểm tra Tag của tất cả các đường

Set Associative

- Kết hợp hai phương pháp trên
- Thể hiện ưu điểm của cả phương pháp trực tiếp và kết hợp, đồng thời giảm nhược điểm

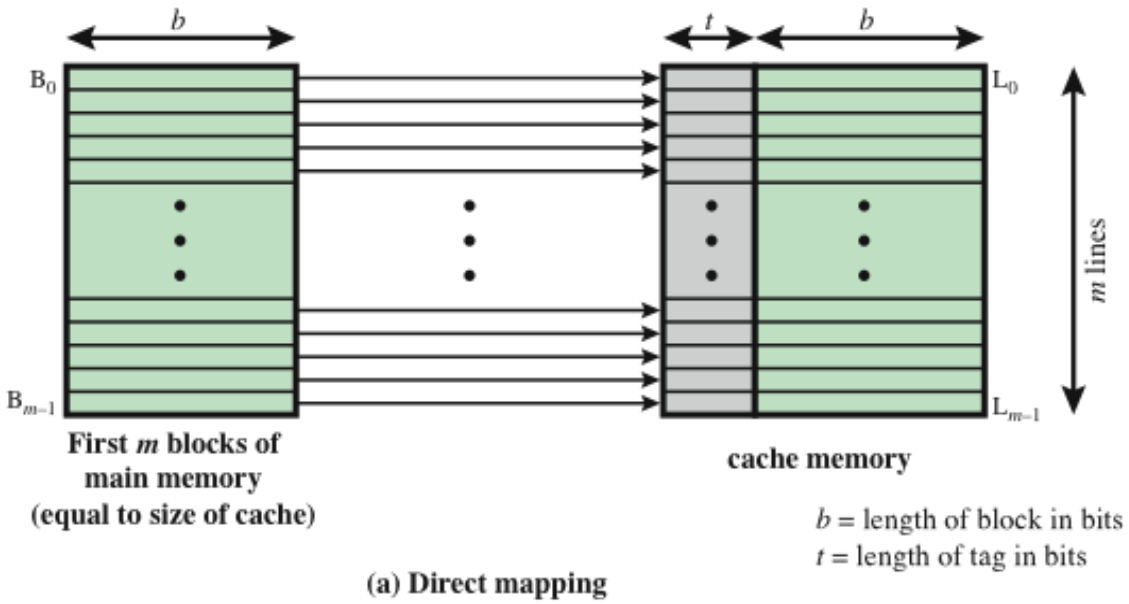
Ánh xạ trực tiếp (Direct mapping)

- Mỗi **khối (block)** của bộ nhớ chính được ánh xạ vào một **đường (line)** nhất định của bộ nhớ cache.
- Cách xác định: giả sử BN cache có ***m*** line. Vậy, block thứ ***j*** trong BN chính sẽ được ánh xạ vào line nào trong BN cache?

Với i là số thứ tự line mà block đó được ánh xạ vào, ta có

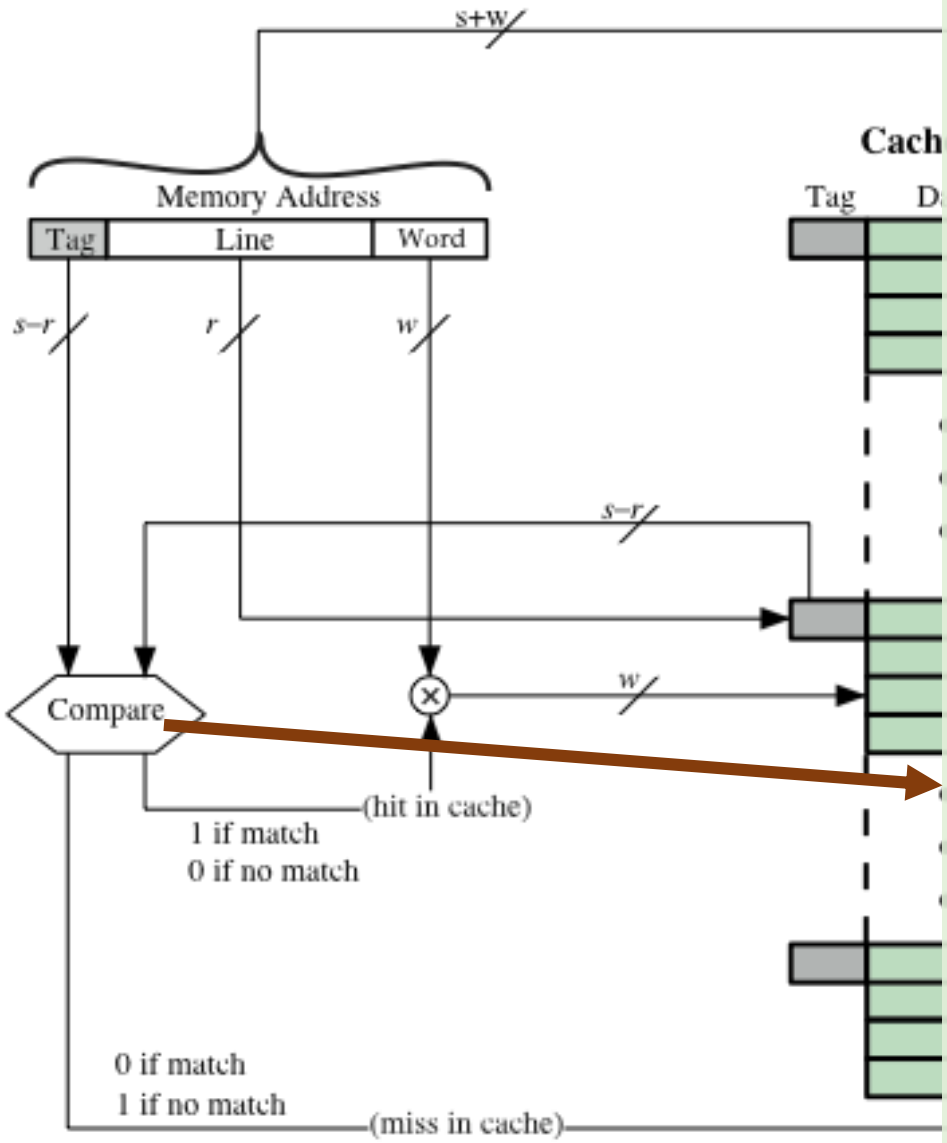
$$i = j \ modulo \ m$$ **(phép chia lấy dư)**

- Do vậy, nhiều block sẽ được ánh xạ vào một line. Để xác định block nào đang được ánh xạ vào cache: sử dụng trường tag



(a) Direct mapping

Tổ chức cache ánh xạ trực tiếp

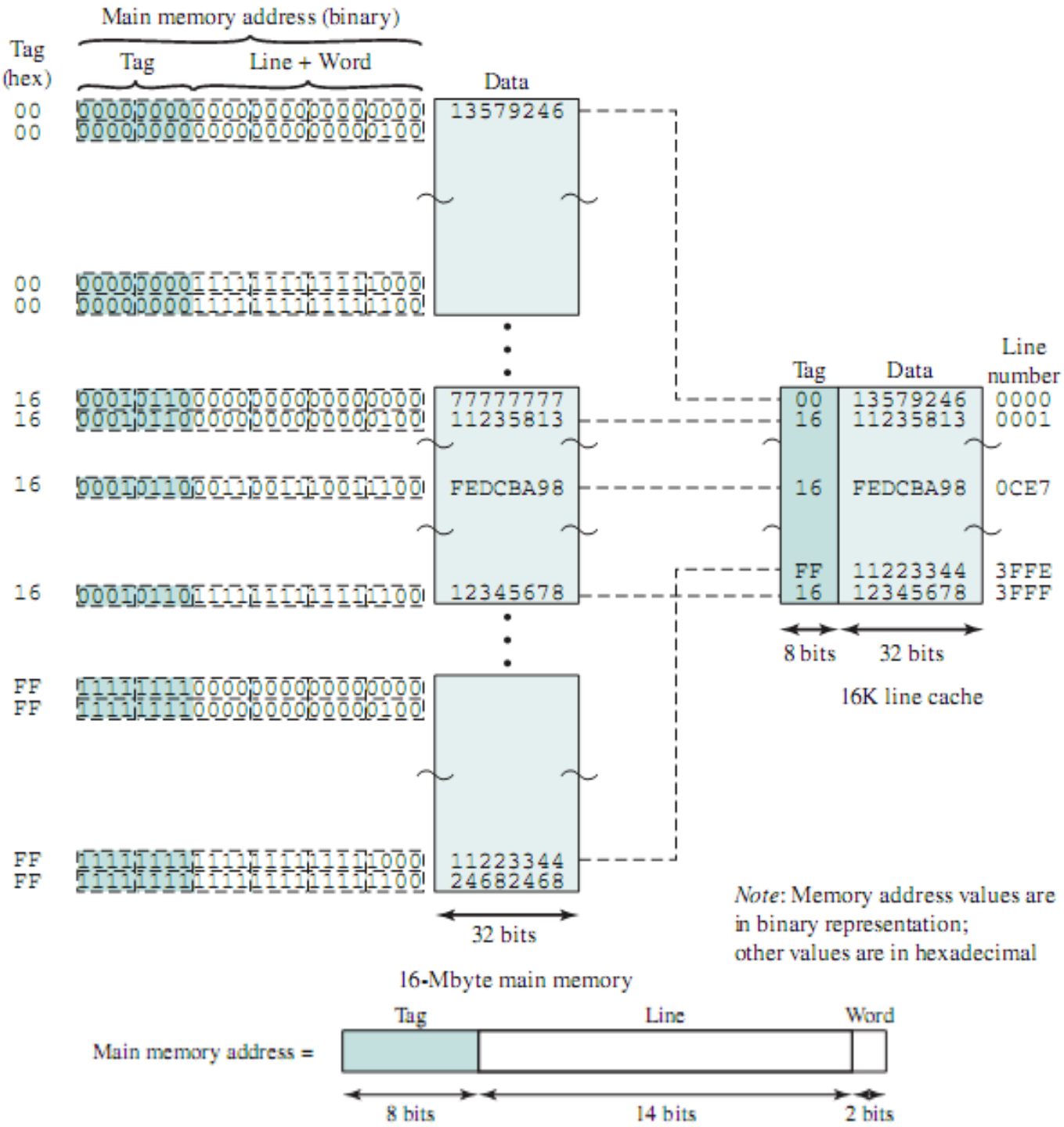


Khi truy xuất một word, logic cache tách **địa chỉ BN** thành 3 trường:

- Word (w bit): xác định một word trong block
- Line (r bit): xác định block đó được ánh xạ vào line thứ tự bao nhiêu trong cache
- Tag (s-r bit): xác định block nào đang được ánh xạ vào line đó

So sánh **tag** của đc này với **tag** của line trong cache để xác định xem có phải block đó đang được ánh xạ vào cache không

Ví dụ ánh xạ trực tiếp



Tổng kết ánh xạ trực tiếp

- Độ dài địa chỉ = $(s + w)$ bit
- Số ô nhớ trong bộ nhớ chính = 2^{s+w} word hoặc byte
- Kích thước khối = kích thước đường = 2^{w} word hoặc byte
- Số khối trong bộ nhớ chính = $2^{s+w}/2^{w} = 2^{s}$
- Số đường trong bộ nhớ cache $= m = 2^{r}$
- Kích thước của tag = $(s - r)$ bit

Nhược điểm: các khối lưu cố định tại 1 đường trong bộ nhớ cache. Vậy nếu chương trình tham chiếu các từ lặp lại từ hai khối mà cùng ánh xạ đến 1 đường thì cache liên tục phải đổi dl từ memory vào, làm giảm hiệu suất (hiện tượng *thrashing)*

Bài tập ví dụ

Bộ nhớ chính: 2^{16} byte, kích thước khối 8 byte, ánh xạ trực tiếp vào cache 32 đường.

a. 16 bit địa chỉ được chia thành các trường Tag, Line và Word như thế nào?

b. Các địa chỉ sau sẽ được lưu ở đường nào của cache?

0001 0001 0001 1011 1101 0000 0001 1101

1100 0011 0011 0100 1010 1010 1010 1010

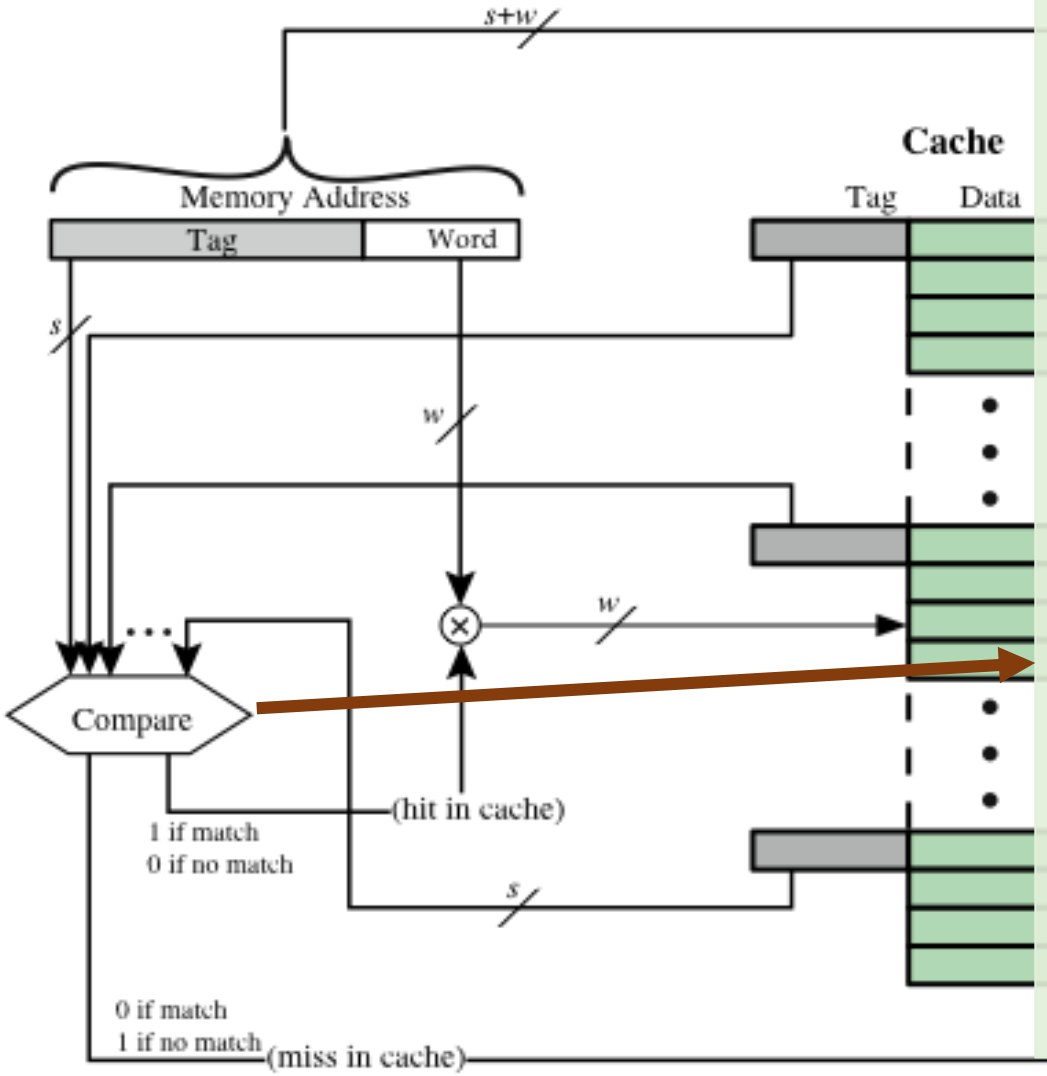
c. Giả sử byte có địa chỉ 0001 1010 0001 1010 được lưu ở cache, các byte nào của bộ nhớ chính cũng được lưu trên đường đó?

d. Có bao nhiêu byte có thể được lưu trên cache?

Ánh xạ kết hợp (Associative Mapping)

- Ánh xạ kết hợp khắc phục nhược điểm của ánh xạ trực tiếp bằng cách cho phép mỗi khối được nạp vào bất kỳ đường nào của bộ nhớ cache
- Trong trường hợp này, *bộ logic điều khiển bộ nhớ cache (cache control logic)* tách địa chỉ bộ nhớ thành hai trường: Tag và Word. Trường Tag hiển thị duy nhất một khối bộ nhớ chính.
- Để xác định liệu một khối có trong bộ nhớ cache, *bộ logic điều khiển bộ nhớ cache* phải cùng lúc kiểm tra mỗi Tag của một đường để so sánh

Tổ chức ánh xạ kết hợp

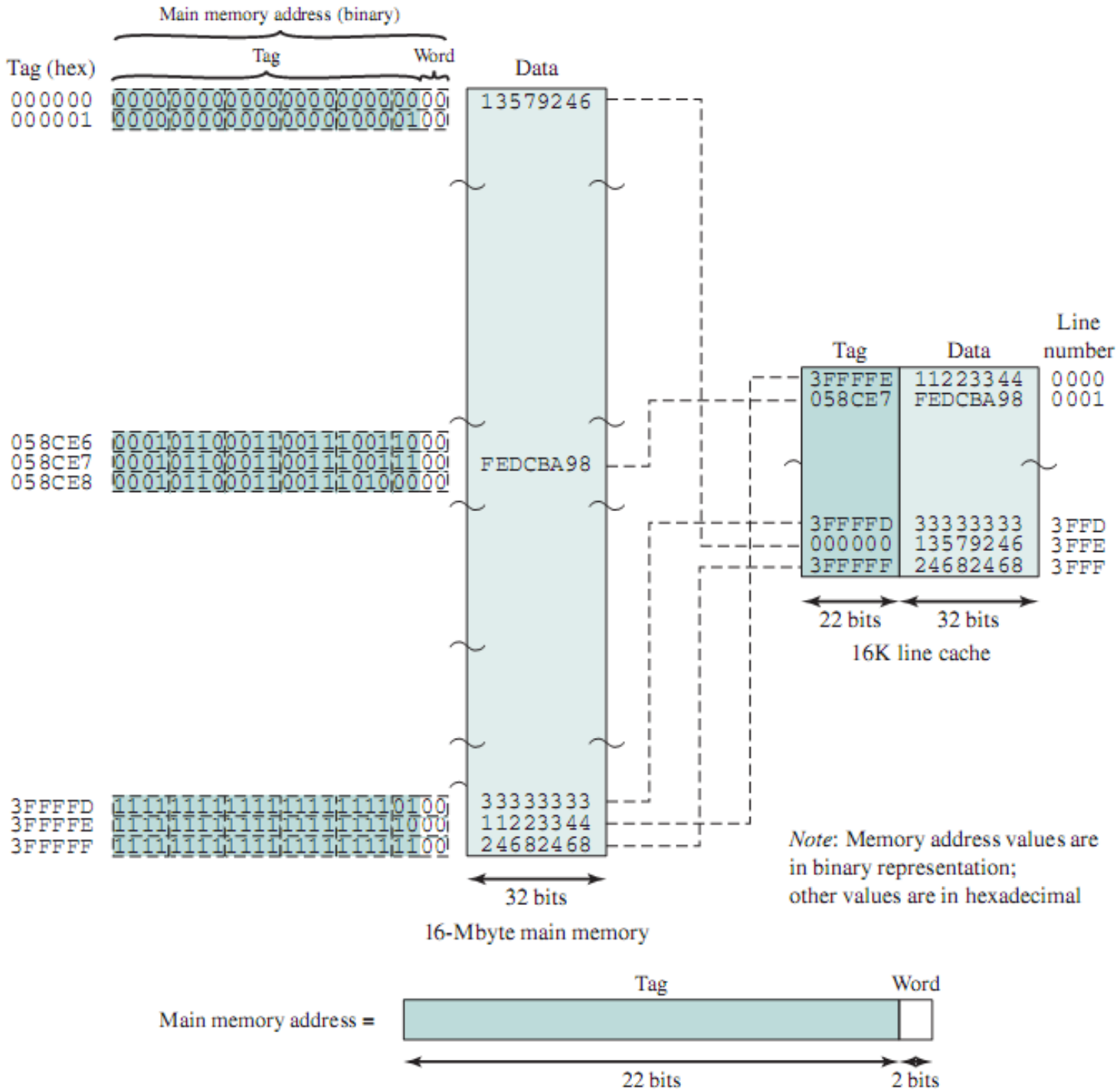


Khi truy xuất một word, logic cache tách **địa chỉ BN** thành 2 trường:

- Word (w bit): xác định một word trong block
- Tag (s bit): xác định block nào đang được ánh xạ vào line đó

So sánh **Tag** của đc này với **Tag** của các line trong cache để xác định xem block đó có đang đc ánh xạ vào cache không.

Ví dụ ánh xạ kết hợp



Tổng hợp ánh xạ kết hợp

- Chiều dài địa chỉ = (s + w) bit
- Số ô nhớ được đánh địa chỉ = 2^{s+w} word hoặc byte
- Kích thước khối = kích thước đường = 2^{w} word hoặc byte
- Số lượng khối trong bộ nhớ chính = $2^{s+ w}/2^{w} = 2^{s}$
- Số đường trong cache = không xác định
- Chiều dài trường tag = s bit

Ưu điểm: linh hoạt khi thay thế một khối và đọc một khối mới vào cache. Các thuật toán thay thế được xây dựng để tối ưu hóa tỷ lệ truy cập.
Nhược điểm: mạch phức tạp để thực hiện việc kiểm tra tất cả trường Tag của các đường trong cache một cách song song.

Ánh xạ tập kết hợp (Set Associative Mapping)

- Tận dụng các ưu điểm của cả phương pháp trên đồng thời giảm nhược điểm của chúng
- Chia cache thành một số **Tập (set)** - Mỗi Tập chứa một số đường
- Một khối sẽ được ánh xạ vào một đường bất kỳ trong một Tập nhất định

- Quan hệ

  $m = v * k$

  $i = j/v$ *(chia lấy phần dư)*

  Trong đó

  i = *số thứ tự Tập trong cache*

  j = *số thứ tự khối trong bộ nhớ chính*

  m = *số lượng đường trong cache*

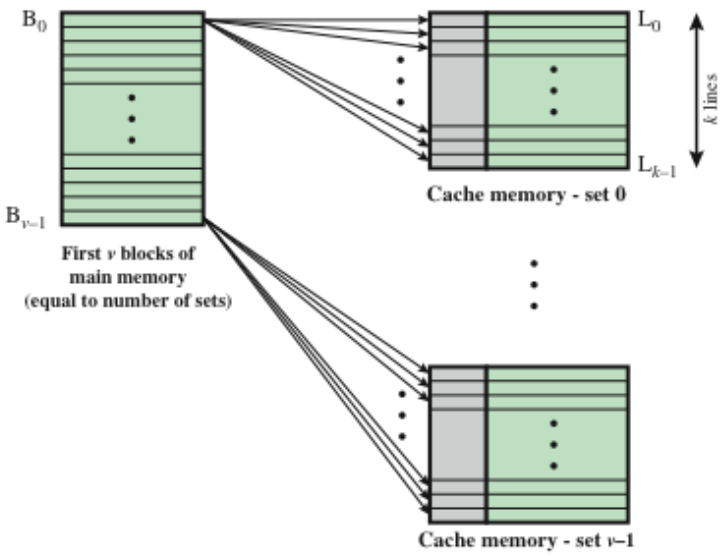
  v = *số lượng Tập có trong cache*

  k = *số lượng đường trong mỗi Tập*

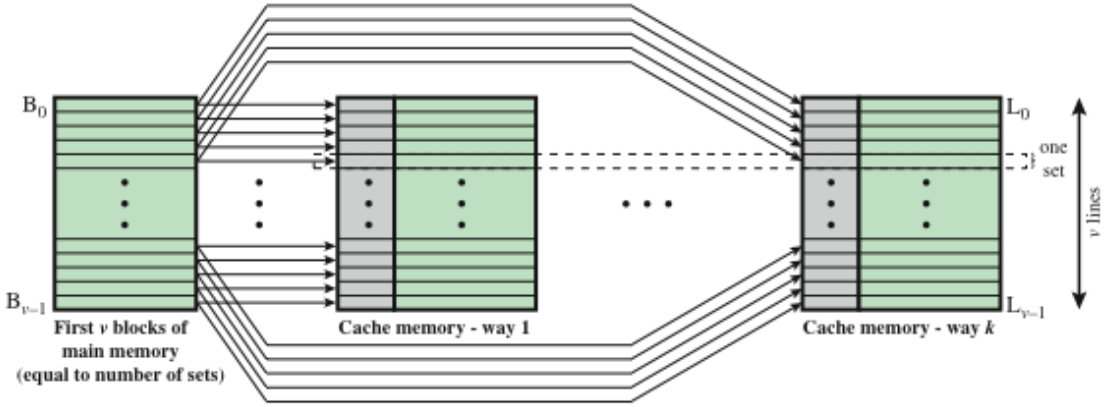
- Ví dụ: 1 Tập có 2 đường
  - Ánh xạ kết hợp 2 chiều
  - Một khối có thể nằm trong 1 trong 2 đường trong một Tập

Ánh xạ từ bộ nhớ chính đến bộ nhớ Cache:

k-Way Set Associative



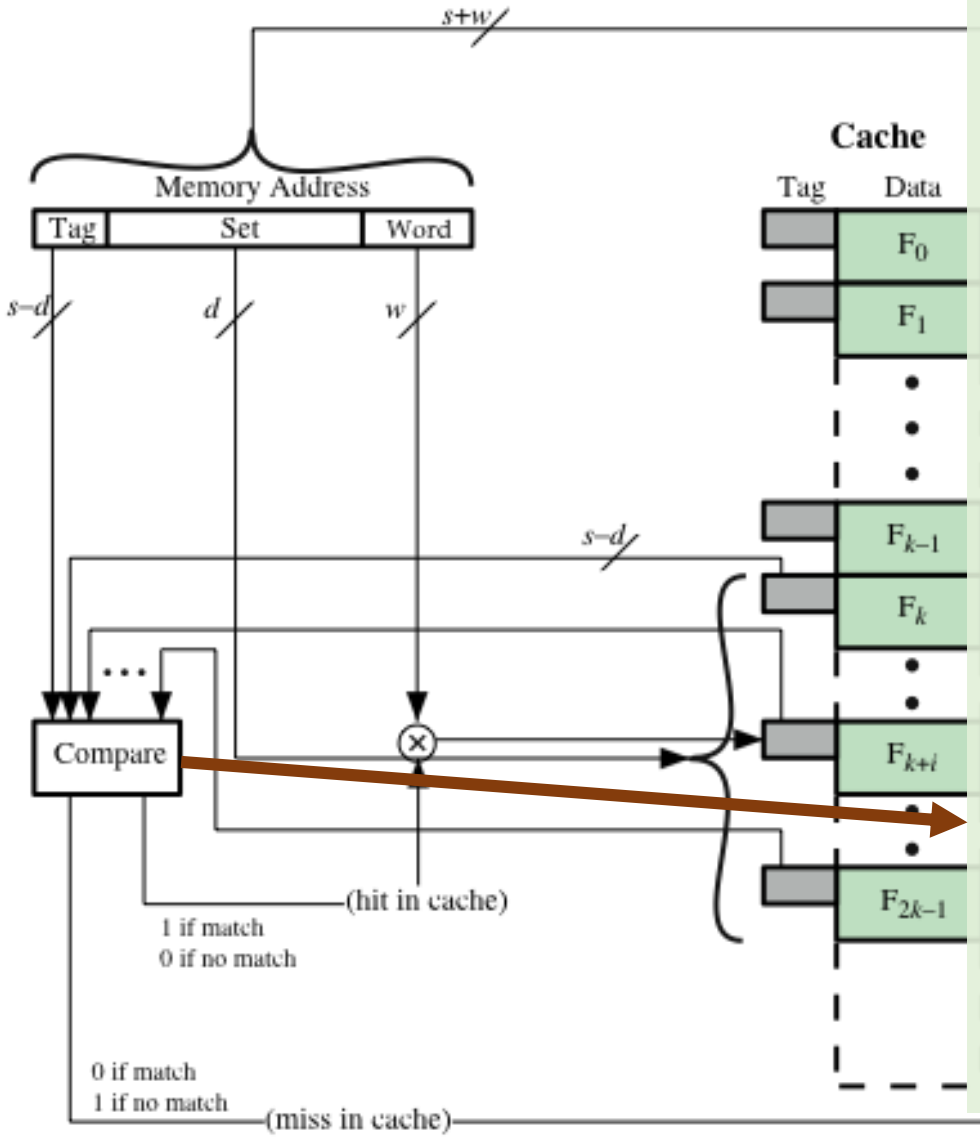
(a) v associative-mapped caches



(b) k direct-mapped caches

Figure 4.13 Mapping From Main Memory to Cache: *k*-way Set Associative

Tổ chức cache *k*-Way Set Associative



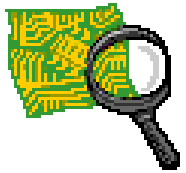
Khi truy xuất một word, logic cache tách **địa chỉ BN** thành 3 trường:

- Word (w bit): xác định một word trong block
- Set (d bit): xác định block đó được ánh xạ vào set thứ tự bao nhiêu trong cache
- Tag (s-d bit): xác định block nào đang được ánh xạ vào line nào trong Set đó

So sánh **tag** của đc này với **tag** của line trong Set để xác định xem có phải block đó đang được ánh xạ vào cache không

Tổng kết ánh xạ Set Associative

- Chiều dài địa chỉ = (s + w) bit
- Số lượng ô nhớ được đánh địa chỉ = 2^{s+w} word hoặc byte
- Kích thước khối (hoặc đường) = 2^{w} word hoặc byte
- Số khối trong BN chính = $2^{s+w}/2^{w}=2^{s}$
- Số đường trong 1 Tập= k
- Số lượng Tập = v = 2^{d}
- Số lượng đường trong cache = m= kv = k * 2^{d}
- Kích thước cache = $k * 2^{d+w}$ word hoặc byte
- Độ rộng trường Tag = (s – d) bits



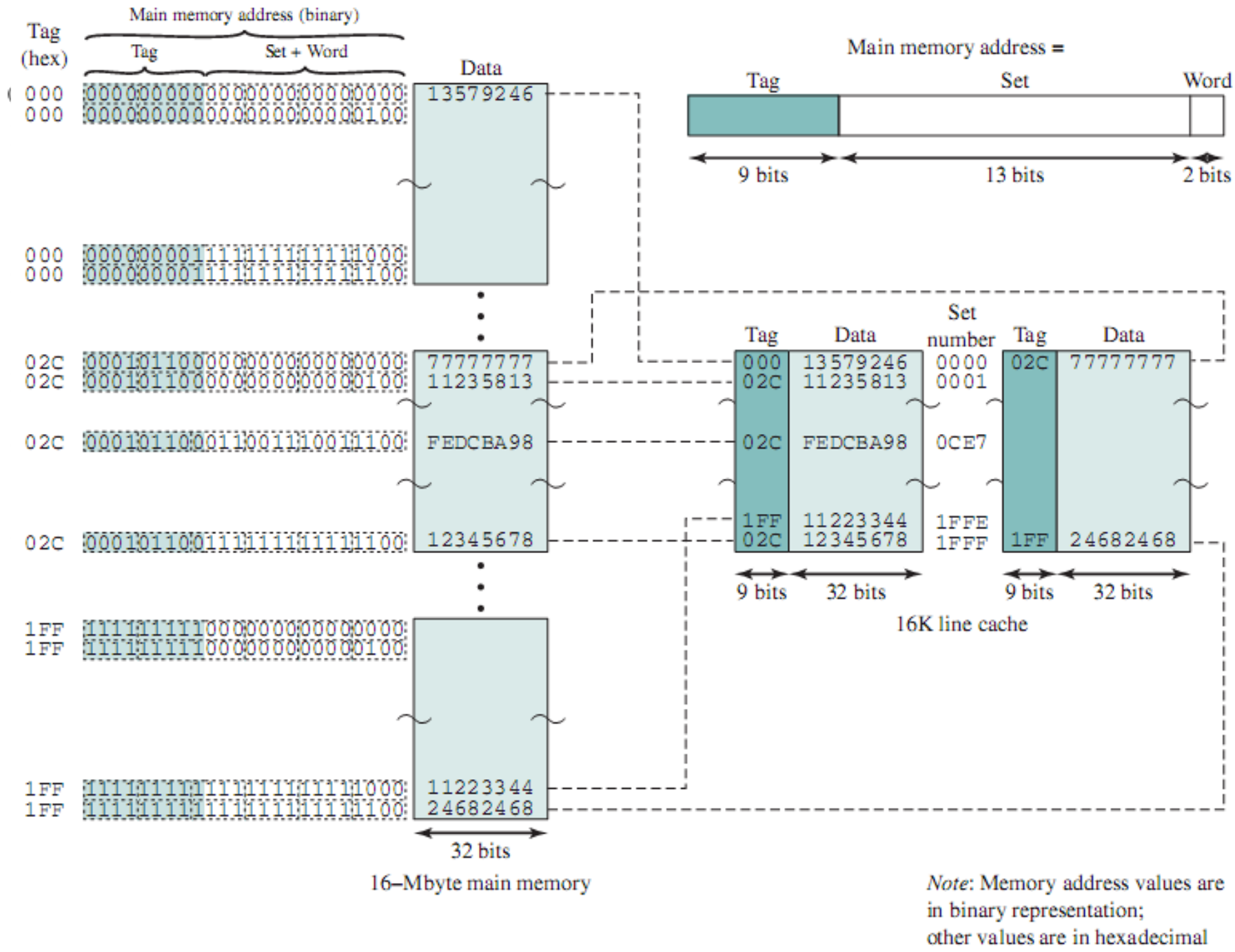
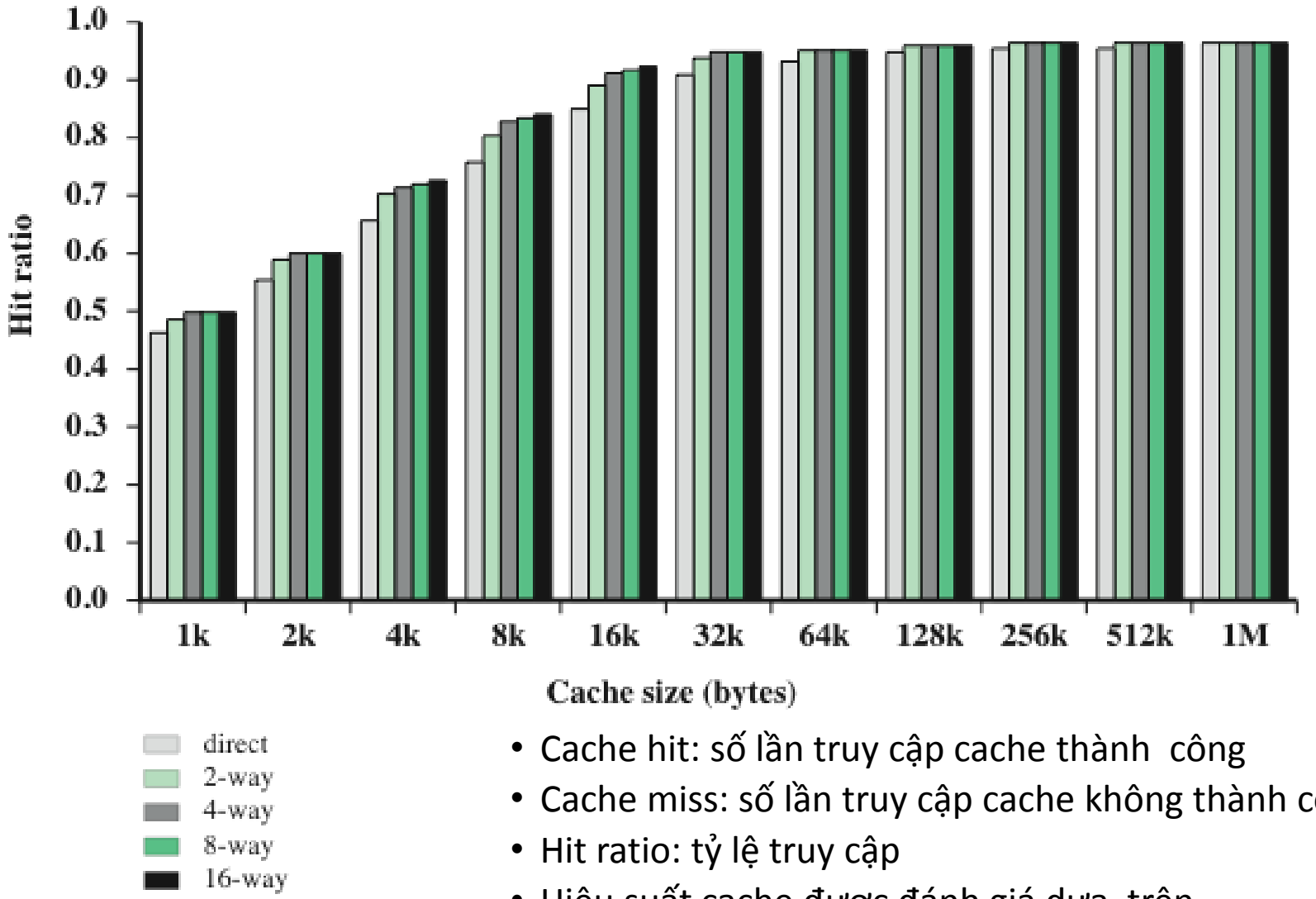


Figure 4.15 Two-Way Set-Associative Mapping Example

So sánh hiệu suất của các PP ánh xạ



- Cache hit: số lần truy cập cache thành công
- Cache miss: số lần truy cập cache không thành công
- Hit ratio: tỷ lệ truy cập
- Hiệu suất cache được đánh giá dựa trên

$$hit\ ratio = \frac{cache\ hit}{tổng\ số\ lần\ truy\ \ cập}$$

Figure 4.16 V

Bài tập ví dụ

- Xét VXL 32 bit có một cache 16-Kbyte, ánh xạ tập kết hợp 4-way. Giả sử một đường gồm 4 từ 32-bit (mỗi từ nhớ 32 bit, 1 đường có 4 từ). Xác định các trường của địa chỉ được sử dụng để ánh xạ cache. Từ nhớ có địa chỉ ABCDE8F8 được ánh xạ vào vị trí nào trong cache.
- Bộ nhớ Cache 64 đường sử dụng ánh xạ tập kết hợp 4 đường. Bộ nhớ chính có 4K khối, mỗi khối có kích thước 128 từ. Xác định định dạng địa chỉ bộ nhớ

d. Thuật toán thay thế

- Khi bộ nhớ cache đã đầy, nếu một khối mới được đưa vào cache, một trong những khối hiện có phải được thay thế
  - Đối với ax trực tiếp: một khối bất kỳ chỉ có thể ánh xạ vào 1 đường cụ thể. Nên khi cần đưa 1 khối mới vào cache buộc phải xóa dữ liệu cũ trên đường tương ứng đi.
  - Đối với các kỹ thuật kết hợp và tập két hợp, một khối có thể được ánh xạ vào 1 số đường. Vậy khi đưa 1 khối mới vào cache, ta cần xác định xem khối đó sẽ được ánh xạ vào đường nào: **thuật toán thay thế**
- Để đạt được tốc độ cao, thuật toán phải được thực hiện trong phần cứng

4 thuật toán thay thế phổ biến nhất

- Least recently used (LRU)
  - Hiệu quả nhất
  - Thay thế khối nằm trong cache lâu nhất mà không có tham chiếu đến nó
  - Do triển khai đơn giản, LRU là thuật toán thay thế phổ biến nhất
- First-in-first-out (FIFO)
  - Thay thế khối đã nằm trong cache lâu nhất
  - Dễ dàng thực hiện như một kỹ thuật vòng đệm hoặc round-robin
- Least frequently used (LFU)
  - Thay thế khối có ít tham chiếu đến nó nhất
  - Ở mỗi line thêm vào một bộ đếm, mỗi khi có tham chiếu đến line nào, bộ đếm của line đó tăng thêm 1 đơn vị
- Ngẫu nhiên
  - Có thể thay thế bất cứ khối nào
  - Các nghiên cứu đã chỉ ra: thay thế ngẫu nhiên chỉ làm giảm hiệu suất của hệ thống đi một chút so với các thuật toán thay thế ở trên

e. Chính sách ghi

Khi một khối trong cache được thay thế, có 2 trường hợp cần xem xét:

- Nếu dữ liệu trong cache không thay đổi, có thể ghi đè khối mới lên mà không cần ghi khối cũ ra trước
- Nếu ít nhất 1 thao tác ghi đã được thực hiện trên 1 word trong đường của cache thì bộ nhớ chính phải được cập nhật bằng cách ghi dữ liệu từ cache ra bộ nhớ trước khi đưa khối mới vào

Có hai vấn đề phải đối mặt:

- Nhiều thiết bị có thể có quyền truy cập vào bộ nhớ chính
- Trong trường hợp VXL đa nhân: nhiều cache tương ứng với các nhân → gây khó khăn trong việc quản lý dữ liệu

Có hai chính sách ghi: ***write though*** và ***write back***

Write Through
và Write Back

- Write through: khi VXL gửi lệnh ghi dữ liệu ra bộ nhớ, dữ liệu sẽ được ghi đồng thời ra cả cache và BN
  - Kỹ thuật đơn giản nhất
  - Tất cả các thao tác ghi được thực hiện cho bộ nhớ chính cũng như cache
  - Nhược điểm: tạo ra lưu lượng bộ nhớ đáng kể và có thể tạo ra nút cổ chai
- Write back: khi VXL gửi lệnh ghi, dữ liệu chỉ được cập nhật trên cache, sử dụng 1 bit (gọi là dirty bit) thiết lập giá trị để đánh dấu là dữ liệu đã bị thay đổi. Việc cập nhật dữ liệu lên BNC chỉ xảy ra khi thay thế khối mới vào cache
  - Giảm thao tác ghi bộ nhớ
  - Dữ liệu trên BNC không có hiệu lực. Cơ chế DMA bắt buộc phải thực hiệu qua cache
  - Nhược điểm: mạch phức tạp và khả năng có nút cổ chai

f. Kích thước Line

- Khi khối dữ liệu được lấy ra và đặt trong cache, sẽ thu được không chỉ word mong muốn mà còn 1 số word liền kề
- Khi kích thước khối tăng, ban đầu tỷ lệ truy cập sẽ tăng do nguyên tắc cục bộ → dữ liệu hữu ích (dữ liệu có khả năng lớn được truy cập trong câu lệnh tiếp theo) được đưa vào cache
- Tuy nhiên, khi kích thước khối quá lớn, tỷ lệ này giảm đi do:

  1, Các khối lớn hơn làm giảm số lượng đường trong một cache

  2, Khi kích thước khối lớn, mỗi word thêm vào lại càng xa word yêu cầu → ít có khả năng truy xuất

g. Cache nhiều cấp

- Với các hệ thống ngày nay, người ta thường sử dụng nhiều cache trong kiến trúc
- *Cache trên chip (cache on chip)* làm giảm hoạt động bus ngoài của bộ xử lý; tăng tốc thời gian xử lý và tăng hiệu năng toàn hệ thống
  - Khi lệnh hoặc dữ liệu được tìm thấy trong cache → không cần truy cập bus
  - Truy cập bộ nhớ *cache trên chip* nhanh hơn đáng kể
  - Trong giai đoạn này, bus tự do hỗ trợ các lượt truyền khác
- *Cache 2 cấp (cache 2 level)* : sử dụng một *cache on chip (cache L1)* và một cache bên ngoài (*cache L2)*
  - Cải thiện hiệu suất
  - Việc sử dụng cache nhiều cấp làm cho các vấn đề thiết kế liên quan đến cache phức tạp hơn, gồm kích thước, thuật toán thay thế, chính sách ghi

Tỉ lệ truy cập (L1 & L2) cho 8 Kbyte và 16 Kbyte L1

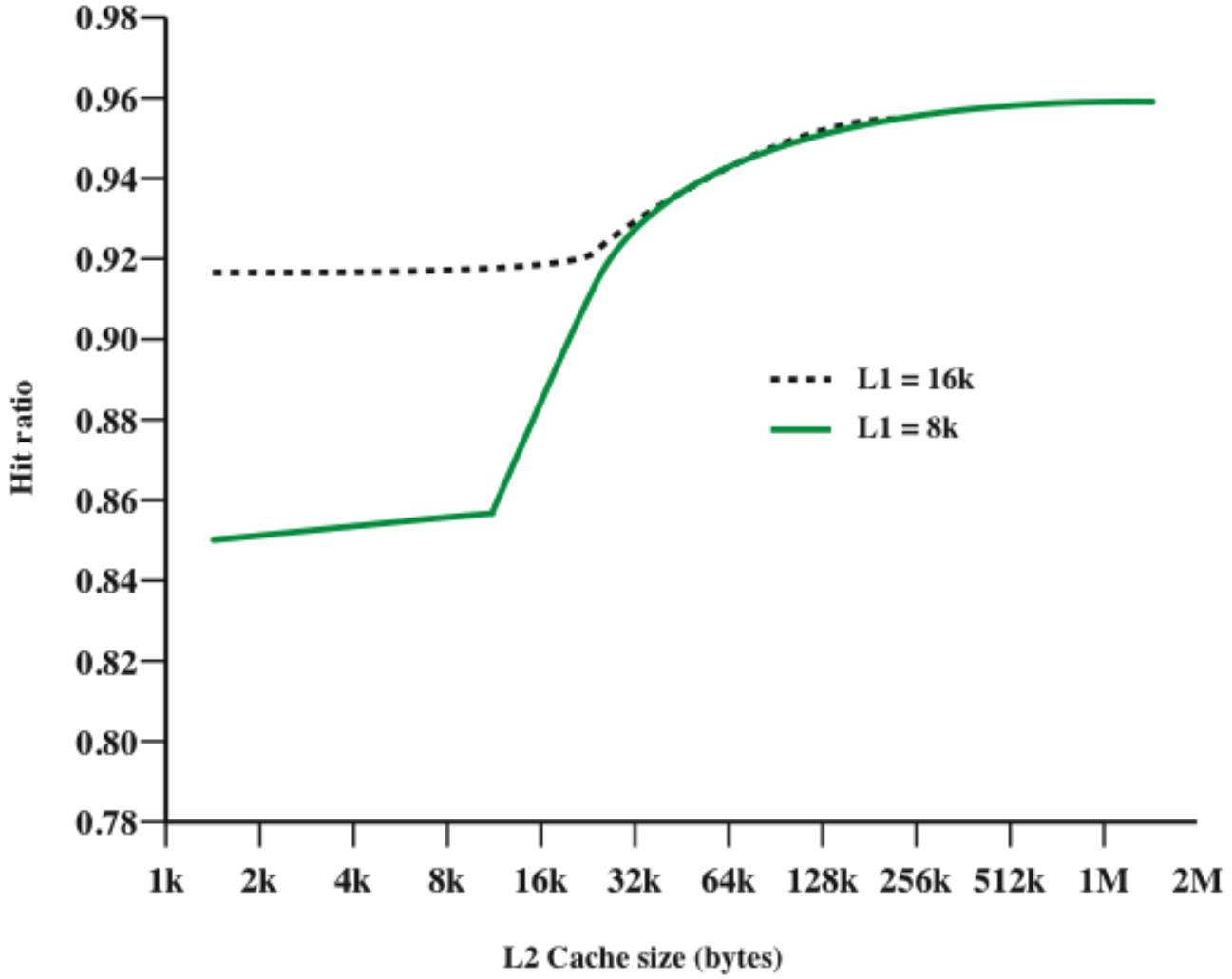


Figure 4.17 Total Hit Ratio (L1 and L2) for 8 Kbyte and 16 Kbyte L1

Cache thống nhất / cache phân chia

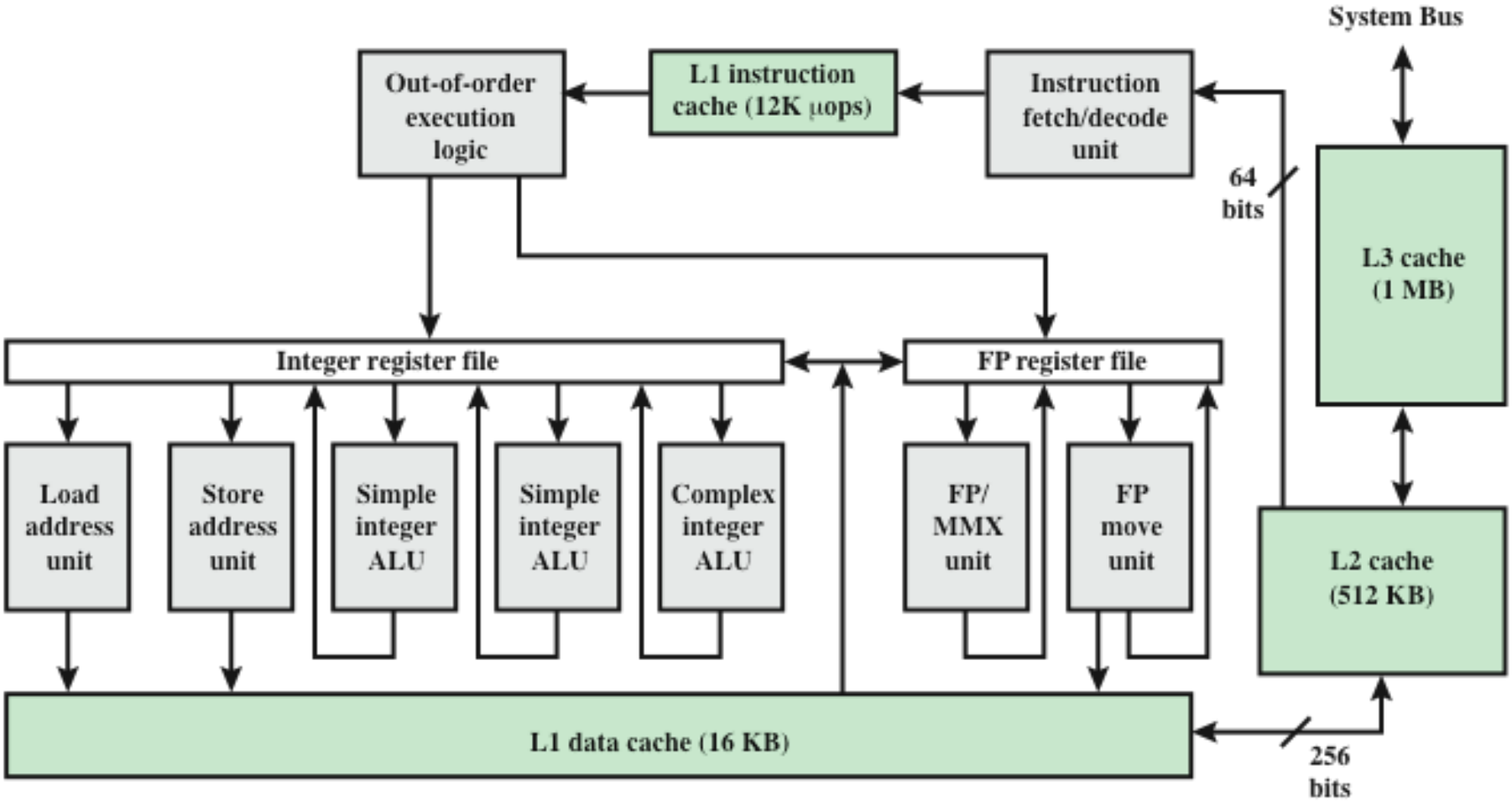
- Cache thống nhất: lệnh và dữ liệu được lưu trên cùng một cache
  - Ưu điểm: Tốc độ truy cập cao hơn
    - Tự động cân bằng giữa việc nạp dữ liệu và lệnh
    - Chỉ cần thiết kế và thực hiện một bộ nhớ cache
- Cache phân chia: lệnh và dữ liệu được lưu trên hai cache khác nhau, thường là cache L1
  - Ưu điểm:Loại bỏ sự cạnh tranh cache giữa khối tìm nạp/giải mã lệnh và khối thực hiện
- Xu hướng: cache phân chia ở L1 và cache thống nhất ở các level cao hơn

4.4.Tổ chức Cache Pentium 4

Problem	Solution	Processor on which Feature First Appears
External memory slower than the system bus.	Add external cache using faster memory technology.	386
Increased processor speed results in external bus becoming a bottleneck for cache access.	Move external cache on-chip, operating at the same speed as the processor.	486
Internal cache is rather small, due to limited space on chip	Add external L2 cache using faster technology than main memory	486
Contention occurs when both the Instruction Prefetcher and the Execution Unit simultaneously require access to the cache. In that case, the Prefetcher is stalled while the Execution Unit's data access takes place.	Create separate data and instruction caches.	Pentium
Increased processor speed results in external bus becoming a bottleneck for L2 cache access.	Create separate back-side bus that runs at higher speed than the main (front-side) external bus. The BSB is dedicated to the L2 cache.	Pentium Pro
	Move L2 cache on to the processor chip.	Pentium II
Some applications deal with massive databases and must have rapid access to large amounts of data. The on-chip caches are too small.	Add external L3 cache.	Pentium III
	Move L3 cache on-chip.	Pentium 4

Bảng 4.4 Intel Cache Evolution

Sơ đồ khối Pentium 4



Các chế độ hoạt động Cache Pentium 4

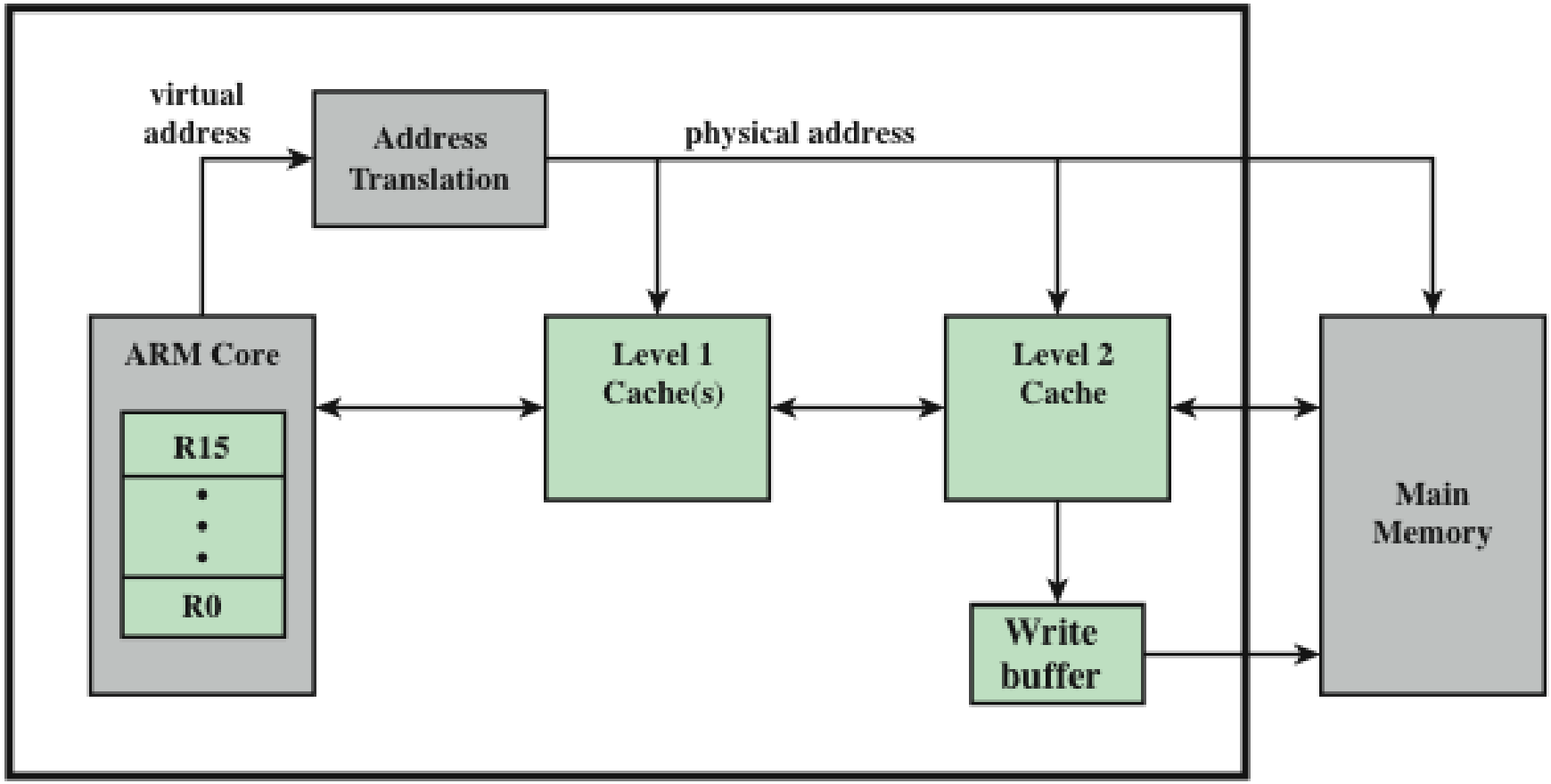
Control Bits	Operating Mode		
NW	Cache Fills	Write Throughs	Invalidates
0	Enabled	Enabled	Enabled
0	Disabled	Enabled	Enabled
1	Disabled	Disabled	Disabled

Note: CD = 0; NW = 1 là kết hợp không hợp lệ.

4.5. Tổ chức cache ARM
Đặc tính Cache ARM

Core	Cache Type	Cache Size (kB)	Cache Line Size (words)	Associativity	Location	Write Buffer Size (words)
ARM720T	Unified	8	4	4-way	Logical	8
ARM920T	Split	16/16 D/I	8	64-way	Logical	16
ARM926EJ-S	Split	4-128/4-128 D/I	8	4-way	Logical	16
ARM1022E	Split	16/16 D/I	8	64-way	Logical	16
ARM1026EJ-S	Split	4-128/4-128 D/I	8	4-way	Logical	8
Intel StrongARM	Split	16/16 D/I	4	32-way	Logical	32
Intel Xscale	Split	32/32 D/I	8	32-way	Logical	32
ARM1136-JF-S	Split	4-64/4-64 D/I	8	4-way	Physical	32

ARM Cache và tổ chức bộ đệm ghi



Tổng kết

Chương 4: Bộ nhớ Cache

- Đặc điểm của hệ thống bộ nhớ
  - Vị trí
  - Dung lượng
  - Đơn vị truyền
- Bộ nhớ phân cấp
  - How much?
  - How fast?
  - How expensive?
- Nguyên lý bộ nhớ Cache
- Các yếu tố trong thiết kế cache
  - Địa chỉ bộ nhớ cache
  - Kích thước bộ nhớ cache
  - Ánh xạ bộ nhớ
  - Thuật toán thay thế
  - Chính sách ghi
  - Kích thước line
  - Cache nhiều cấp
- Tổ chức cache Pentium 4
- Tổ chức cache ARM

Từ khóa

- Cache: bộ nhớ cache
- Cache hit: việc truy cập thành công vào bộ nhớ cache
- Cache miss: truy cập BN cache không thành công
- Hit ratio: tỷ lệ truy cập BN cache
- Cache mapping: ánh xạ BN cache
- Write policy: chính sách ghi
- Cache L1, L2, L3: BN cache level 1, 2,3
- Line: đường trong cache
- Block: khối
- Set: tập

Review questions

1. Sự khác nhau giữa truy cập tuần tự, trực tiếp và ngẫu nhiên là gì?
2. Nêu mối quan hệ giữa thời gian truy cập, giá thành và dung lượng bộ nhớ.
3. Sự khác nhau giữa ánh xạ trực tiếp, kết hợp và tập kết hợp là gì?
4. Khái niệm cache phân chia và cache thống nhất. Ứng dụng?
5. Các yếu tố chính trong thiết kế cache là gì
6. Write though và write back khác nhau như thế nào? Kỹ thuật nào giảm số lần truy cập bus hệ thống nhiều hơn?
7. Trình bày các kỹ thuật thay thế.
8. Kích thước của đường có ảnh hưởng như thế nào đến hiệu suất cache (cache hit)?
9. Dung lượng BN cache có ảnh hưởng như thế nào đến hiệu suất cache (cache hit)?