Computer Architecture

Computer Science & Engineering

Chương 5

Tổ chức và Cấu trúc bộ nhớ

Các loại Bộ nhớ (Công nghệ)

- RAM tĩnh (SRAM)
  - 0.5ns – 2.5ns, $2000 – $5000 per GB
- RAM động (DRAM)
  - 50ns – 70ns, $20 – $75 per GB
- Đĩa từ (Magnetic disk)
  - 5ms – 20ms, $0.20 – $2 per GB
- Bộ nhớ lý tưởng
  - Thời gian truy xuất theo SRAM
  - Dung lượng & Giá thành/GB theo đĩa

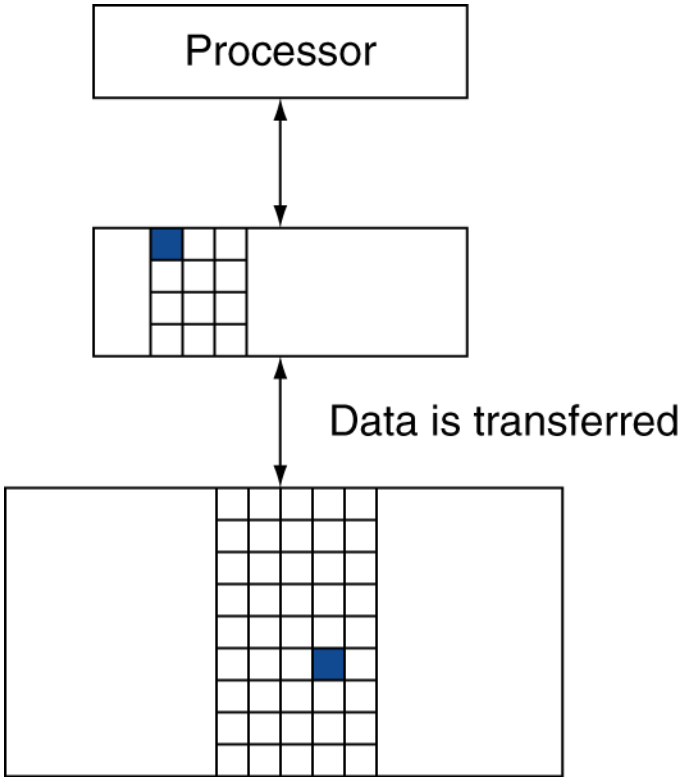
Tính cục bộ (Locality)

- Chương trình truy cập một vùng nhỏ không gian bộ nhớ
- Cục bộ về thời gian (Temporal Locality)
  - Những phần tử vừa được tham chiếu có xu hướng được tham chiếu lại trong tương lai gần
  - Ví dụ: các lệnh trong 1 vòng lặp, các biến quy nạp
- Cục bộ về không gian (Spatial Locality)
  - Những phần tử ở gần những phần tử vừa được tham chiếu có xu hướng được tham chiếu lại trong tương lai gần → Ví dụ: truy cập lệnh trong 1 basic block, dữ liệu mảng

Tận dụng lợi thế về cục bộ

- Tổ chức phân tầng bộ nhớ
- Lưu trữ mọi thứ trên đĩa
- Chỉ nạp vào bộ nhớ Chính (DRAM) 1 phần đang sử dụng từ đĩa
- Chỉ nạp vào bộ nhớ CACHE (SRAM) 1 phần đang truy cập ở bộ nhớ chính
  - Bộ nhớ Cache là bộ nhớ mà CPU truy cập trực tiếp

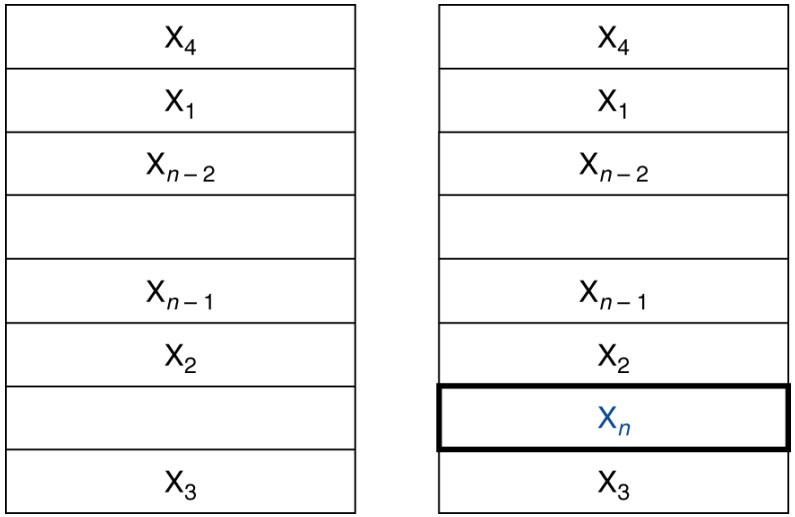
Các lớp tổ chức của bộ nhớ



- Khối (Block=aka line): Đơn vị sao chép
  - Có thể gồm nhiều từ (words)
- Nếu dữ liệu truy cập hiện diện
  - Trúng(hit): đúng dữ liệu cần truy xuất
    - Tỷ lệ trúng (hit rate): hits/accesses
- Nếu dữ liệu truy cập không hiện diện
  - Trật (miss): khối chứa dữ liệu cần được nạp từ lớp thấp hơn
    - Thời gian: giá phải trả để giải quyết
    - Tỷ lệ sai (miss rate): misses/accesses = (1 – hit ratio)

Bộ nhớ đệm (Cache)

- Bộ nhớ Cache
  - Trong cấu trúc lớp của tổ chức hệ thống bộ nhớ, Cache là lớp trực tiếp với CPU
- Giả sử truy cập X_1, …, X_{n-1}, X_n

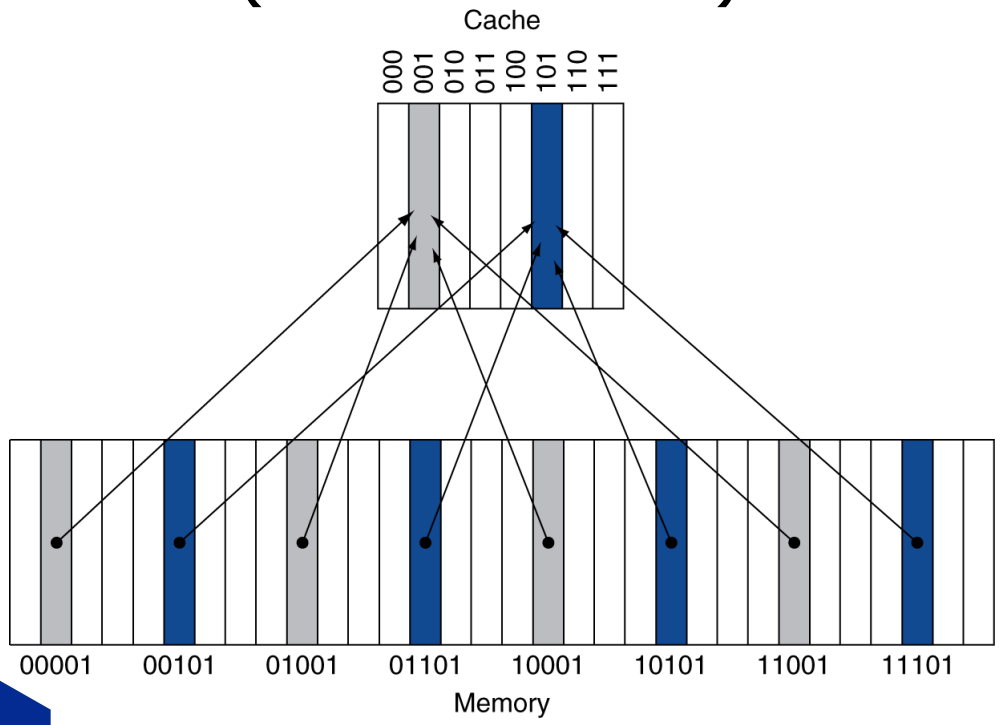


a. Before the reference to X_n b. After the reference to X_n

- Làm sao biết được dữ liệu cần truy cập có trong Cache?
- Ở đâu?

Ánh xạ trực tiếp

- Vị trí xác định qua địa chỉ
- Ánh xạ trực tiếp: Chỉ có 1 lực chọn
  - (Block address) modulo (#Blocks in cache)



- Chỉ số khối (#Blocks) là lũy thừa của 2
- Sử dụng các bit thấp của địa chỉ

Nhãn (Tags) & Bit hợp lệ

- Làm sao có thể biết được một khối nào đó tồn tại trong cache?
  - Chứa cả địa chỉ khối và dữ liệu
  - Thực tế, chỉ cần những bit cao
  - Gọi là nhãn (tag)
- Nếu dữ liệu không hiện diện thì
  - Valid bit: 1 = hiện diện, 0 = không hiện diện
  - Khởi động ban đầu là không hiện diện (0)

Ví dụ Cache

- 8-blocks, 1 word/block, ánh xạ trực tiếp
- Trạng thái ban đầu

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

Ví dụ (tt.)

Word addr	Binary addr	Hit/miss	Cache block
22	10 110	Miss	110

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	**Y**	**10**	**Mem[10110]**
111	N		

Ví dụ (tt.)

Word addr	Binary addr	Hit/miss	Cache block
26	11 010	Miss	010

Index	V	Tag	Data
000	N		
001	N		
010	Y	11	Mem[11010]
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Ví dụ (tt.)

Word addr	Binary addr	Hit/miss	Cache block
22	10 110	Hit	110
26	11 010	Hit	010

Index	V	Tag	Data
000	N		
001	N		
010	Y	11	Mem[11010]
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Ví dụ (tt.)

Word addr	Binary addr	Hit/miss	Cache block
16	10 000	Miss	000
3	00 011	Miss	011
16	10 000	Hit	000

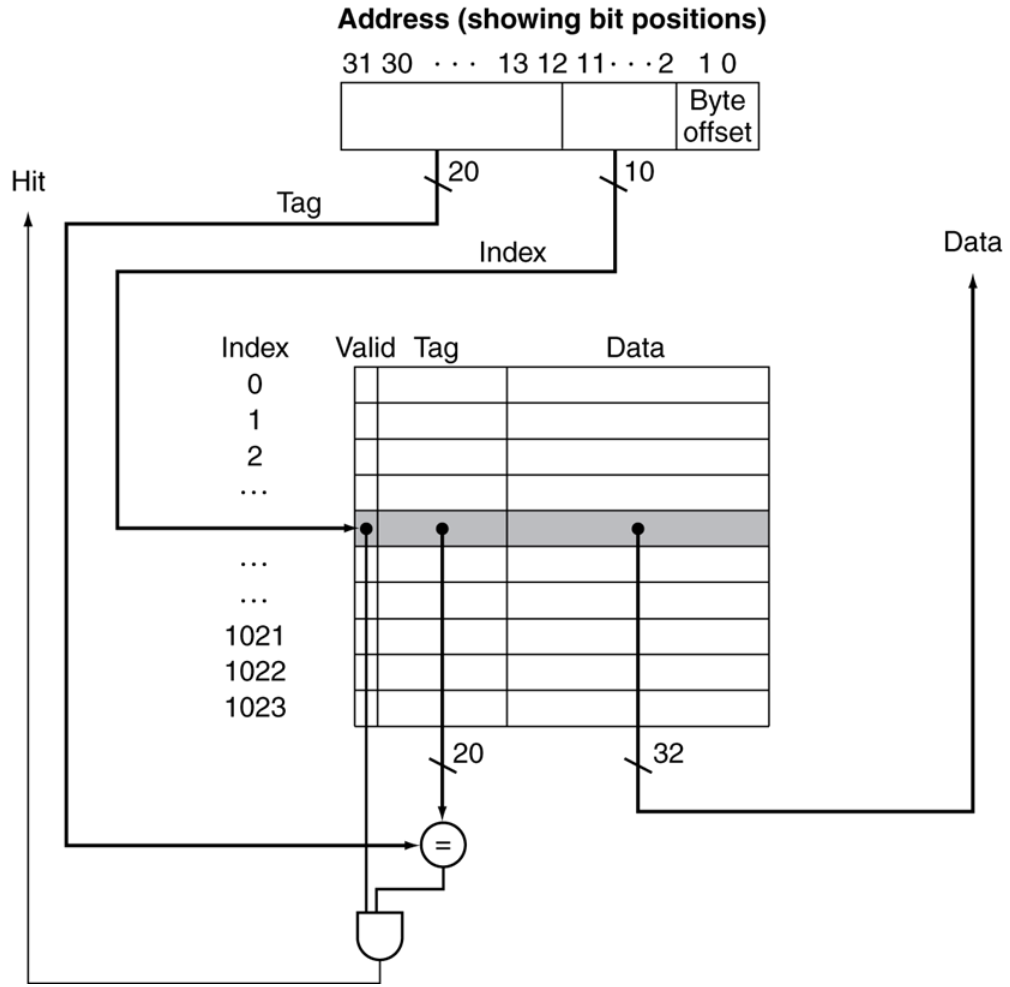
Index	V	Tag	Data
000	**Y**	**10**	**Mem[10000]**
001	N		
010	Y	11	Mem[11010]
011	**Y**	**00**	**Mem[00011]**
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Ví dụ (tt.)

Word addr	Binary addr	Hit/miss	Cache block
18	10 010	Miss	010

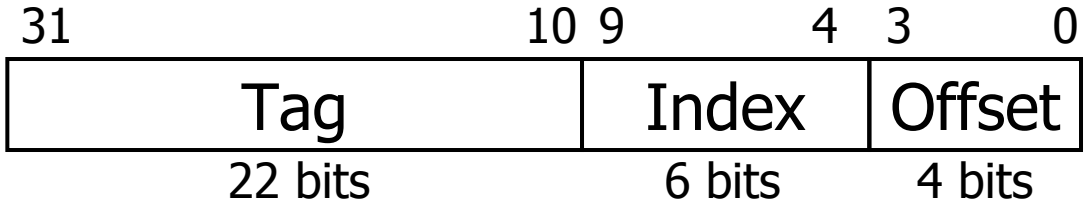
Index	V	Tag	Data
000	Y	10	Mem[10000]
001	N		
010	**Y**	**10**	**Mem[10010]**
011	Y	00	Mem[00011]
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Chia nhỏ không gian địa chỉ



Ví dụ: Khối có kích thước lớn

- 64 blocks, 16 bytes/block
  - Địa chỉ 1200 sẽ ánh xạ vào khối nào?
- Địa chỉ Block = $\lfloor 1200/16 \rfloor = 75$
- Chỉ số Block = 75 modulo 64 = 11



Nhận xét về kích thước khối

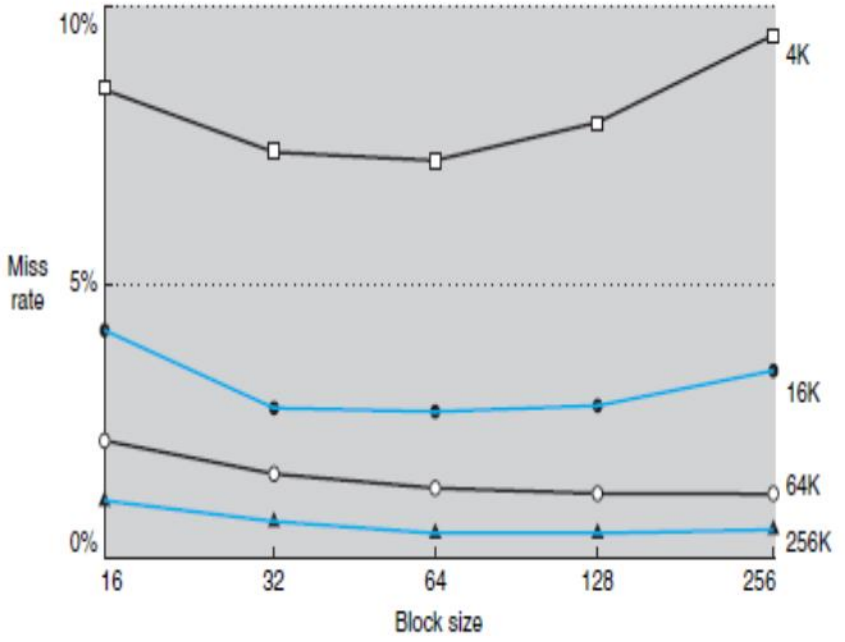


FIGURE 5.8 Miss rate versus block size. Note that the miss rate actually goes up if the block size is too large relative to the cache size. Each line represents a cache of different size. (This figure is independent of associativity, discussed soon.) Unfortunately, SPEC CPU2000 traces would take too long if block size were included, so this data is based on SPEC92.

Kích thước khối lớn: giảm "tỷ lệ trật"

- Do cục bộ không gian

Với Cache có kích thước cố định

- Kích thước khối lớn ⇒ ít khối trong Cache
  - nhiều cạnh tranh ⇒ tăng tỷ lệ trượt
- Kích thước khối lớn ⇒ ô nhiễm

Phí tổn với kích thước khối lớn

- không tận dụng được việc giảm tỷ lệ trượt

Xử lý Cache Misses

- CPU sẽ xử lý bình thường theo lộ trình, nếu thông tin có trong cache (cache hit)
- Nếu thông tin không có trong cache (mis)
  - Lộ trình bị "khựng lại" (Stall the CPU pipeline)
  - Nạp 1 khối từ lớp dưới
  - Nếu đó là lệnh (Instruction cache miss)
    - Khởi động lại bước nạp lệnh (instruction fetch)
  - Nếu là truy cập dữ liệu (Data cache miss)
    - Hoàn tất việc truy cập

Xử lý Cache Misses (tt.)

- Các bước khi có Cache miss (lệnh)
  - (PC-4) → Mem
  - Đ/khiển Mem thực hiện đọc & đợi cho đến khi quá trình đọc kết thúc
  - Ghi lên Cache: (1) Dữ liệu; (2) Tag = phần cao địa chỉ của lệnh; (3) V-bit gán lên 1
  - Khởi động lại việc thực hiện lệnh: đọc lệnh (đã tồn tại trong cache)

"Write-Through"

- Khi ghi dữ liệu lên bộ nhớ, nếu tồn tại trong cache → cập nhật khối dữ liệu trong cache
  - Tuy nhiên có thể xuất hiện bất đồng nhất dữ liệu trong cache và bộ nhớ
- Write through: đồng thời cập nhật luôn bộ nhớ
- Thời gian ghi sẽ dài hơn
  - Ví dụ: nếu CPI = 1, 10% số lệnh là lệnh store (ghi bộ nhớ) và (100 chu kỳ/lệnh ghi bộ nhớ)
    - CPI (thực tế) = 1 + 0.1×100 = 11
- Giải pháp: Ghi ra vùng đệm (buffer)
  - Dưới dạng hàng đợi ghi ra bô nhớ
  - CPU tiếp tục ngay (khựng lại khi buffer đầy → đợi)

Write-Back

- Phương án khác: giải quyết vấn đề bất đồng nhất dữ liệu khi "data-write hit"
  - Theo dõi sự thay đổi, cập nhật khối cache (dirty block)
- Nếu khối cache thay đổi quá nhiều (dirty block)
  - Cập nhật bộ nhớ
  - Có thể ghi ra buffer để khối mới thay thế được đọc trước

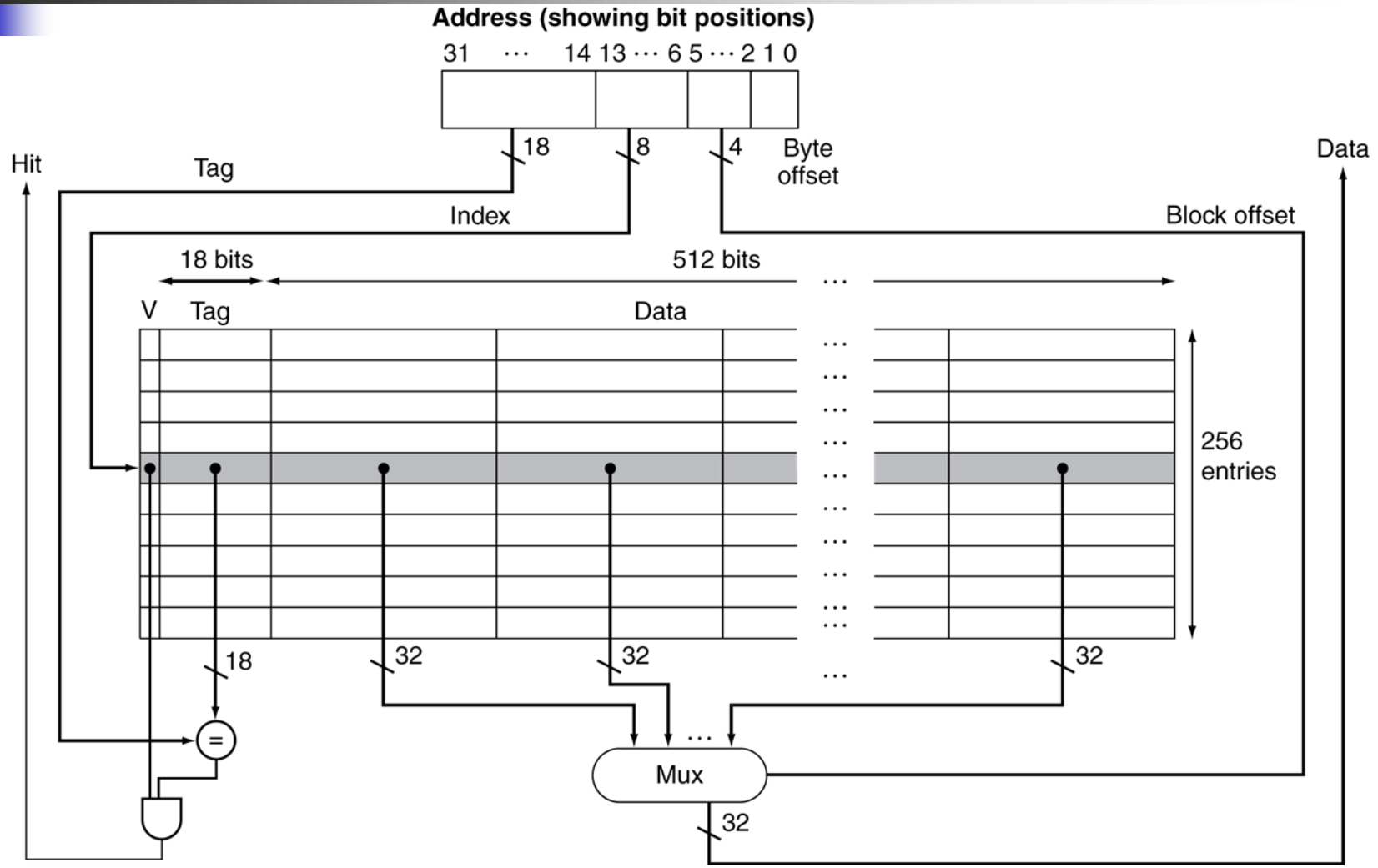
Write Allocation

- Điều gì xảy ra khi có "write miss"?
- Trong trường hợp "write-through"
  - Xác định khối on mis: Nạp từ bộ nhớ, cập nhật
  - Không cần xác định: Không nạp, tìm cách cập nhật thẳng lên bộ nhớ
- Trong trường hợp "write-back"
  - Thường là nạp khối từ bộ nhớ

Ví dụ: Intrinsity FastMATH

- Bộ xử lý nhúng có kiến trúc giống MIPS
  - Cơ chế ống (12-bước hay công đoạn)
  - Mỗi chu kỳ đều đọc lệnh & truy cập dữ liệu
  - Thực hiện cache đơn giản (peak speed)
- Phân chia: cache lệnh & cache dữ liệu
  - 16KB/cache: 256 blocks × 16 words/block
  - Cache dữ liệu: write-through or write-back
- SPEC2000 cho số liệu đo được miss rates
  - I-cache: 0.4% (lệnh)
  - D-cache: 11.4% (dữ liệu)
  - Weighted average: 3.2% (trung bình)

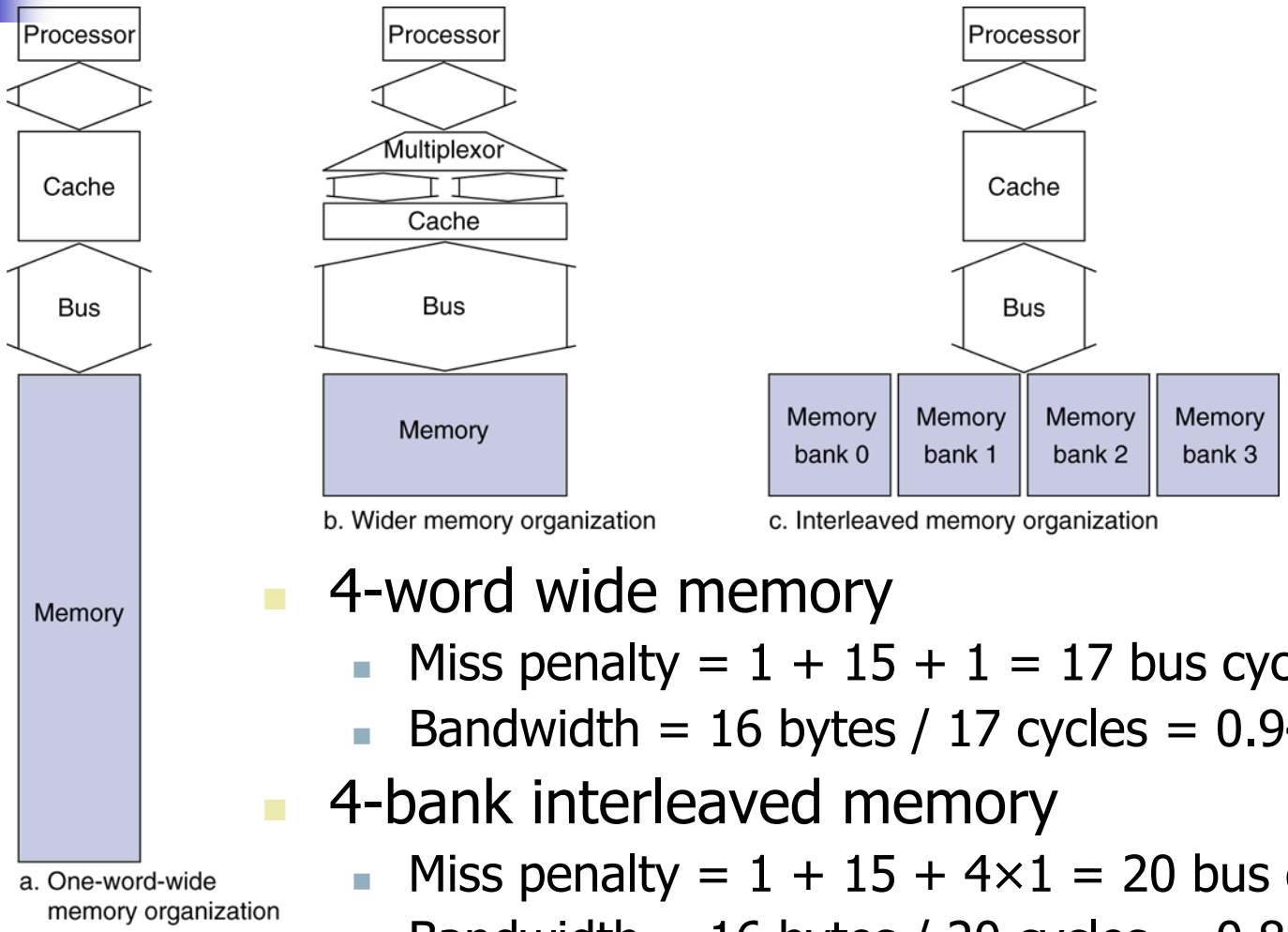
Ví dụ: Intrinsity FastMATH (tt.)



Thiết kế Bộ nhớ hỗ trợ cache

- Sử dụng DRAMs làm bộ nhớ chính
  - Thông tin theo số bit cố định (e.g., 1 word=32bit)
  - Kết nối với tuyến bus cũng có số bit cố định
    - Bus clock thường chậm hơn CPU clock
- Ví dụ đọc 1 block cache
  - 1 chu kỳ bus xác định tuyến địa chỉ truy xuất
  - 15 chu kỳ bus cho 1 lần truy xuất DRAM
  - 1 chu kỳ bus để vận chuyển thông tin
- Nếu khối có 4 từ (words), 1-word-wide DRAM
  - Miss penalty = 1 + 4×15 + 4×1 = 65 bus cycles
  - Bandwidth = 16 bytes / 65 cycles = 0.25 B/cycle

Tăng băng thông Bộ nhớ



a. One-word-wide memory organization

b. Wider memory organization

c. Interleaved memory organization

- 4-word wide memory
  - Miss penalty = 1 + 15 + 1 = 17 bus cycles
  - Bandwidth = 16 bytes / 17 cycles = 0.94 B/cycle
- 4-bank interleaved memory
  - Miss penalty = 1 + 15 + 4×1 = 20 bus cycles
  - Bandwidth = 16 bytes / 20 cycles = 0.8 B/cycle

Đo hiệu suất Cache

- Các thành phần cấu thành thời gian thực thi của CPU
  - Số chu kỳ thực thi chương trình
    - Bao gồm cả thời gian truy cập cache (hit)
  - Chu kỳ "khựng" bộ nhớ
    - Chủ yếu do không có trong cache (miss)
- Giả thuyết đơn giản là:

$$\text{Memory stall cycles} = \frac{\text{Memory accesses}}{\text{Program}} \times \text{Miss rate} \times \text{Miss penalty}$$

$$= \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Misses}}{\text{Instruction}} \times \text{Miss penalty}$$

Ví dụ: Hiệu suất Cache

- Giả sử
  - I-cache miss rate = 2% (truy xuất lệnh)
  - D-cache miss rate = 4% (truy xuất Dữ liệu)
  - Miss penalty = 100 cycles (Phí tổn theo t/gian)
  - Base CPI (ideal cache) = 2
  - Lệnh Load & stores chiếm 36% của c/trình
- Số chu kỳ "trượt" /lệnh sẽ là
  - I-cache: 0.02 × 100 = 2
  - D-cache: 0.36 × 0.04 × 100 = 1.44
- CPI (thực tế) = 2 + 2 + 1.44 = 5.44
  - CPU với bộ nhớ lý tưởng: 2.72 lần (5.44/2) nhanh hơn

Thời gian truy cập trung bình

- Thời gian truy cập trong trường hợp thông tin tồn tại trong cache cũng rất quan trọng
- Thời gian truy cập bộ nhớ trung bình (AMAT): AMAT = Hit time + Miss rate × Miss penalty
- Ví dụ:
  - CPU với 1ns clock, hit time = 1 cycle, miss penalty = 20 cycles, I-cache miss rate = 5%
  - AMAT = 1 + 0.05 × 20 = 2ns
    - 2 chu kỳ cho mỗi lệnh

Kết luận

- Khi hiệu suất CPU tăng
  - Miss penalty becomes more significant
- Giảm CPI
  - Phần lớn thời gian sẽ tiêu tốn do đợi truy xuất bộ nhớ
- Tăng tần số xung Clock
  - Khựng do truy cập bộ nhớ → tăng chu kỳ CPU
- Không thể bỏ qua hành vi cache khi đánh giá hiệu suất hệ thống

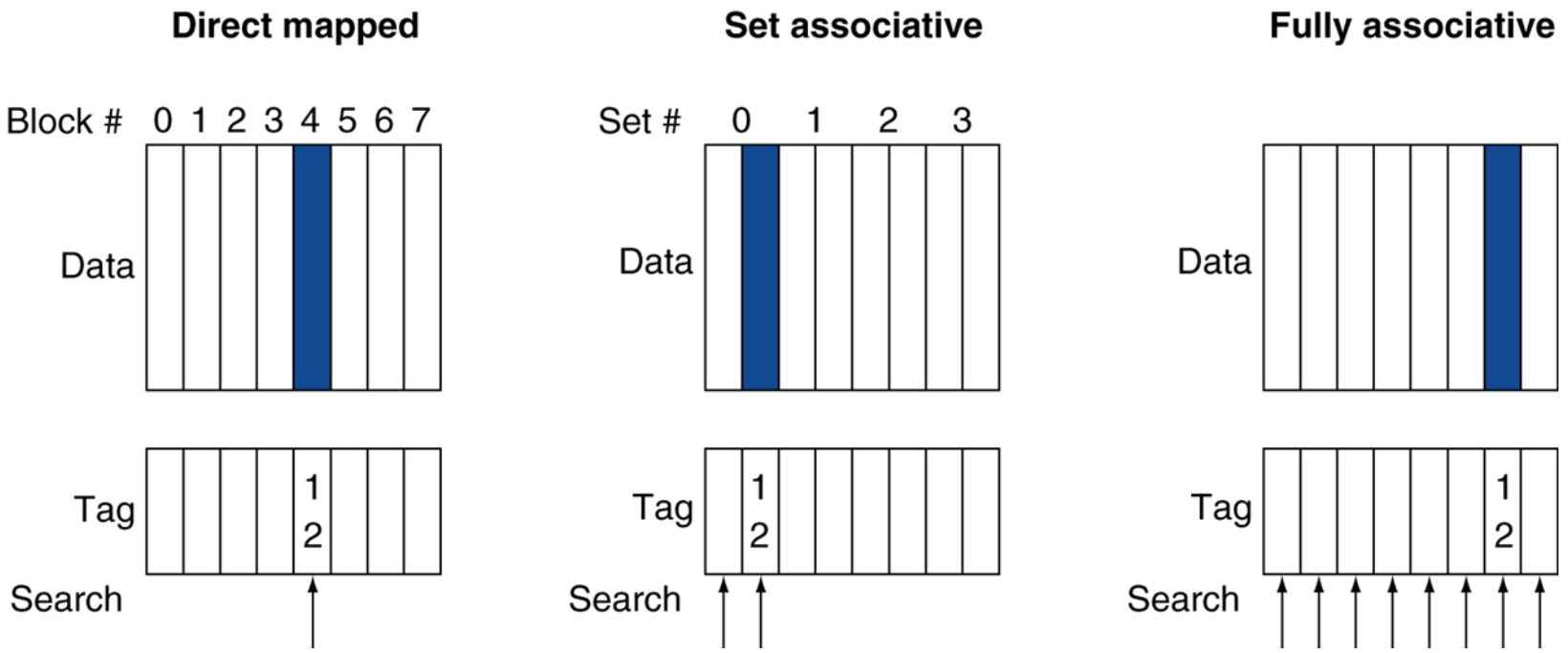
Test ôn lại

Cần tất cả bao nhiêu bits nhớ khi xây dựng bộ nhớ cache với 16 KB dữ liệu, mỗi khối (block) gồm 4 từ (words). Giả sử dùng cache ánh xạ trực tiếp và tuyến địa chỉ là 32 bits ?

Bộ nhớ Caches quan hệ

- Associative cache: Đa ánh xạ (≠ ánh xạ trực tiếp
  - Cho phép 1 khối bộ nhớ ánh xạ vào bất cứ phần tử nào của cache
  - Yêu phải dò tìm tất cả trong cache để truy cập 1 khối nào đó
  - Bộ so sánh cho mỗi phần tử cache (giá thành sẽ cao)
- *Tập quan hệ n*-chiều (*n-ways*)
  - Mỗi tập chứa *n* phần tử
  - Chỉ số khối xác định tập (set)
    - (Block number) modulo (#Sets in cache)
  - Dò tìm các phần tử trong tập để truy cập khối
  - *n* bộ so sánh (giá thành sẽ thấp hơn)

Ví dụ: Cache quan hệ



Các tập quan hệ (*n-ways*)

- cache có 8 phần tử

One-way set associative
(direct mapped)

Block	Tag	Data
0		
1		
2		
3		
4		
5		
6		
7		

Two-way set associative

Set	Tag	Data	Tag	Data
0				
1				
2				
3				

Four-way set associative

Set	Tag	Data	Tag	Data	Tag	Data	Tag	Data
0								
1								

Eight-way set associative (fully associative)

Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data

Ví dụ cụ thể

- Giả sử có 4-block caches
  - (1) ánh xạ trực tiếp; (2) ánh xạ quan hệ tệp 2 (2-way), (3) quan hệ toàn phần
  - Chuỗi các khối truy cập: 0, 8, 0, 6, 8
- (1) Ánh xạ trực tiếp (1-way)

Block address	Cache index	Hit/miss	Cache content after access			
			0	1	2	3
0	0	miss	**Mem[0]**			
8	0	miss	**Mem[8]**			
0	0	miss	**Mem[0]**			
6	2	miss	Mem[0]		**Mem[6]**	
8	0	miss	**Mem[8]**		Mem[6]	

Ví dụ: (tt.)

- 2-way set associative

Block address	Cache index	Hit/miss	Cache content after access			
			Set 0		Set 1	
0	0	miss	**Mem[0]**			
8	0	miss	Mem[0]	**Mem[8]**		
0	0	hit	**Mem[0]**	Mem[8]		
6	0	miss	Mem[0]	**Mem[6]**		
8	0	miss	**Mem[8]**	Mem[6]		

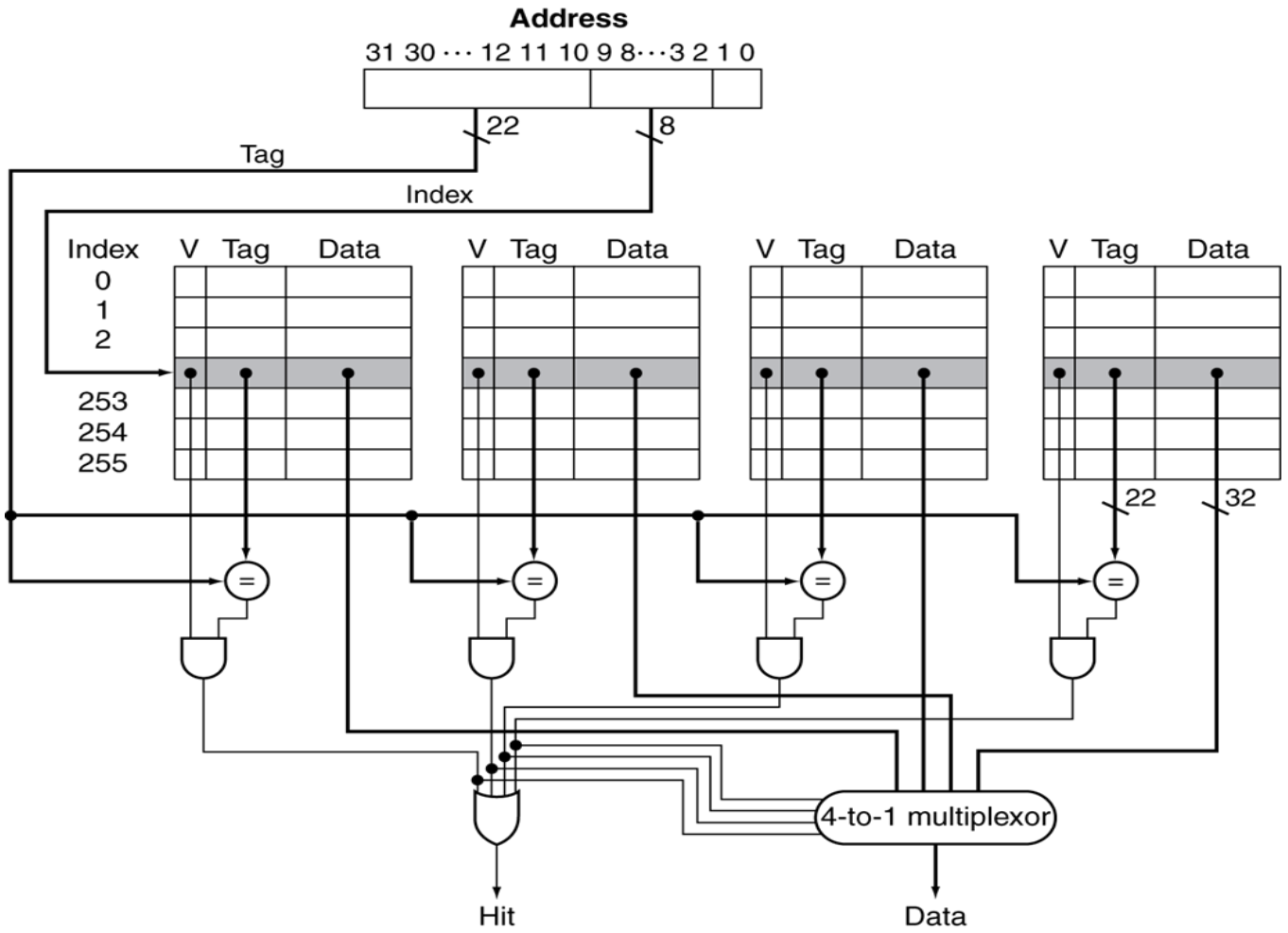
- Fully associative

Block address		Hit/miss	Cache content after access			
0		miss	**Mem[0]**			
8		miss	Mem[0]	**Mem[8]**		
0		hit	**Mem[0]**	Mem[8]		
6		miss	Mem[0]	Mem[8]	**Mem[6]**	
8		hit	Mem[0]	**Mem[8]**	Mem[6]	

Tác dụng của Cache quan hệ

- Tăng "quan hệ" → giảm miss rate
  - But with diminishing returns
- Mô phỏng 1 hệ thống 64KB cache dữ liệu, 16-word/khối với SPEC2000
  - 1-way: 10.3%
  - 2-way: 8.6%
  - 4-way: 8.3%
  - 8-way: 8.1%

Tổ chức hiện thực “ Set Associative Cache”



Chính sách thay thế

- Ánh xạ trực tiếp: không có lựa chọn ≠
- Ánh xạ quan hệ tệp *n*-chiều
  - Chọn phần tử (v=0), nếu có
  - Nếu không, chọn giữa các phần tử trong tệp
- Chọn "ít dùng nhất" (LRU)
  - Chọn phần tử nào ít dùng nhất
    - Đơn giản với 2-way, phức tạp hơn với 4-way, phức tạp cho những tệp > 4
- Chọn ngẫu nhiên
  - Cho kết quả giống LRU đối với tệp quan hệ lớn.

Cache đa cấp (multilevel)

- Cache sơ cấp (cấp 1) gắn trực tiếp với CPU
  - Dung lượng nhỏ nhưng nhanh
- Cache cấp 2: giải quyết khi thông tin không có ở cấp 1
  - Dung lượng lớn hơn, chậm hơn, nhưng vẫn nhanh hơn bộ nhớ chính
- Bộ nhớ chính giải quyết khi thông tin không có ở cấp 2
- Một số hệ thống: cấp 3

Ví dụ: Cache đa cấp

- Giả sử
  - CPU có CPI = 1, clock rate = 4GHz
  - Miss rate/instruction = 2%
  - Thời gian truy suất bộ nhớ chính = 100ns
- Nếu chỉ có cache sơ cấp (cấp 1)
  - Miss penalty = 100ns/0.25ns = 400 cycles
  - Effective CPI = 1 + 0.02 × 400 = 9

Ví dụ: Cache đa cấp (tt.)

- Giả sử thêm cache cấp 2
  - Thời gian truy xuất = 5ns
  - Miss rate toàn cục (to main memory) = 0.5%
- Primary miss trong trường hợp L-2 hit
  - Penalty = 5ns/0.25ns = 20 cycles
- Primary miss trong trường hợp L-2 miss
  - Extra penalty = 0.5% của 400 cycles
- CPI = 1 + 0.02 × 20 + 0.005 × 400 = 3.4
- Tỷ số hiệu năng = 9/3.4 = 2.6

Nhận xét về cache đa cấp

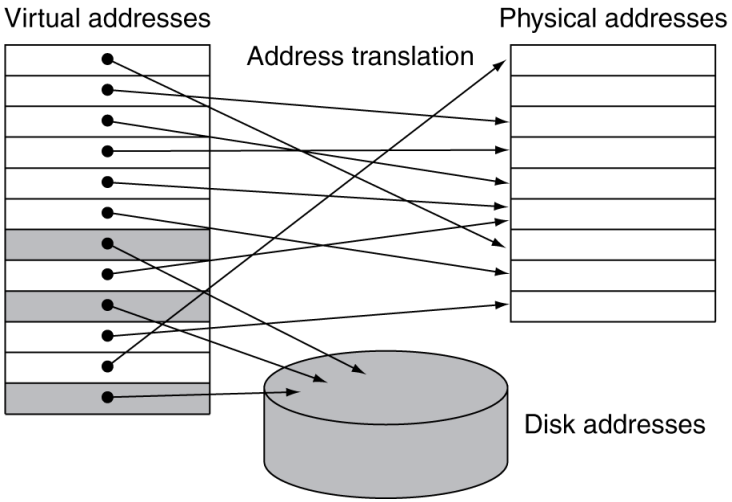
- Cache sơ cấp (Primary cache)
  - Tập trung vào giảm thời gian hit
- Cache cấp 2 (L-2 cache)
  - Tập trung vào giảm tỷ lệ mis, tránh truy xuất bộ nhớ chính
  - Hit time không tác dụng nhiều ở cấp này
- Kết quả
  - L-1 cache thường nhỏ ơn Cache cấp 2
  - L-1 block size nhỏ hơn L-2 block size

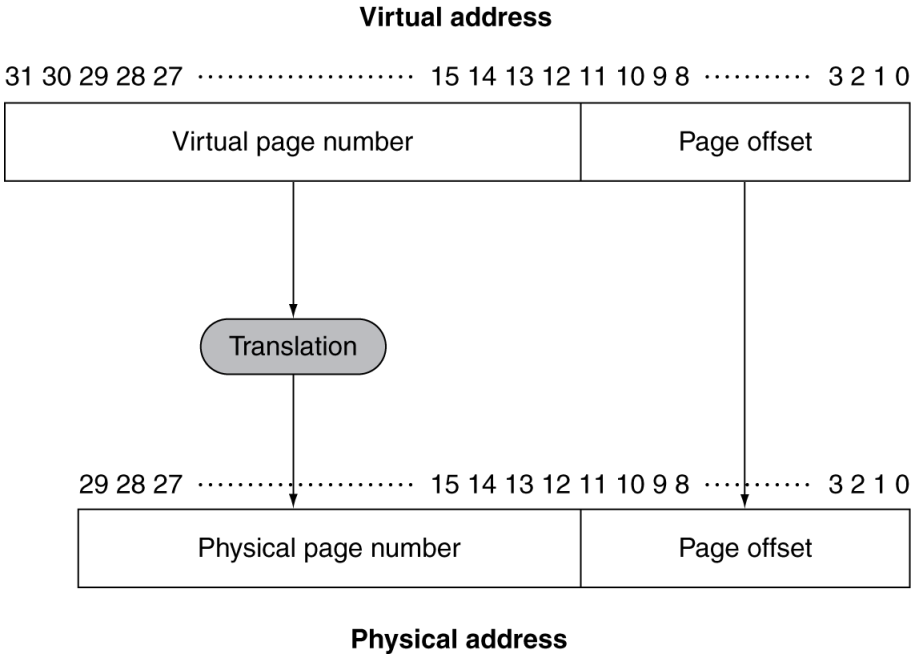
Bộ nhớ ảo (Virtual Memory)

- Bộ nhớ chính được sử dụng như "cache" của bộ nhớ đại trà (Đĩa từ)
  - Quản lý với sự kết hợp phần cứng CPU và hệ điều hành (OS)
- Bộ nhớ chính được sử dụng chung cho nhiều chương trình đồng thời
  - Mỗi chương trình chiếm 1 không gian bộ nhớ riêng & chỉ những phần được dùng thường xuyên
  - Không gian của mỗi chương trình được bảo vệ, trách sự xâm lấn giữa chúng
- CPU và OS chuyển đổi từ địa chỉ ảo sang địa chỉ vật lý
  - Khối "bộ nhớ ảo" được gọi là trang
  - Khi 1 khối ảo không tồn tại trong bộ nhớ → lỗi trang

Chuyển đổi địa chỉ

- Trang có dung lượng cố định (e.g., 4K)





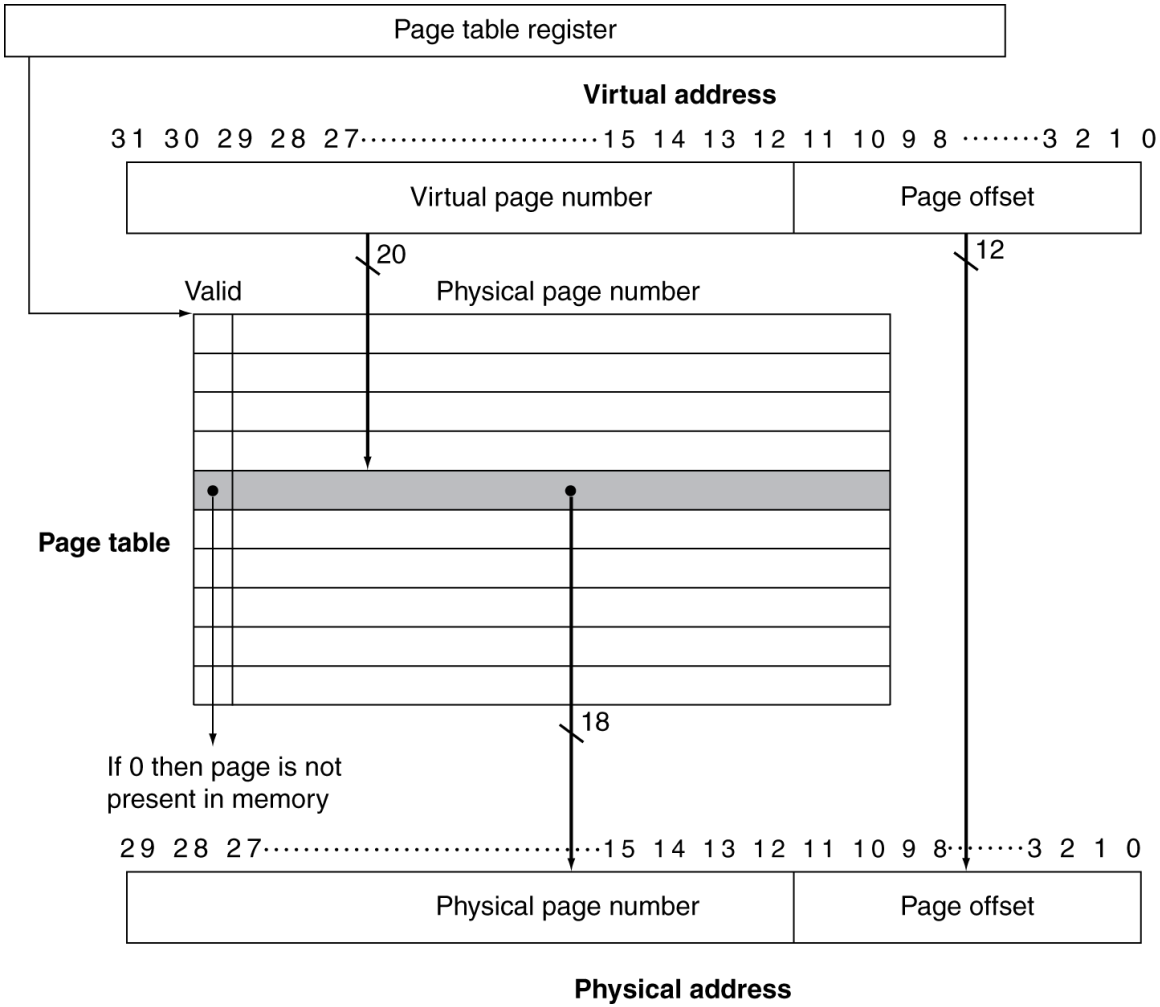
Lỗi trang (Page fault)

- Khi xuất hiện lỗi trang, trang yêu cầu được nạp từ đĩa vào bộ nhớ
  - Thời gian: hàng triệu chu kỳ clock
  - OS sẽ xử lý
- Tiêu chí: tối thiểu số lỗi trang
  - Sắp xếp theo quan hệ toàn phần
  - Sử dụng các giải thuật thông minh

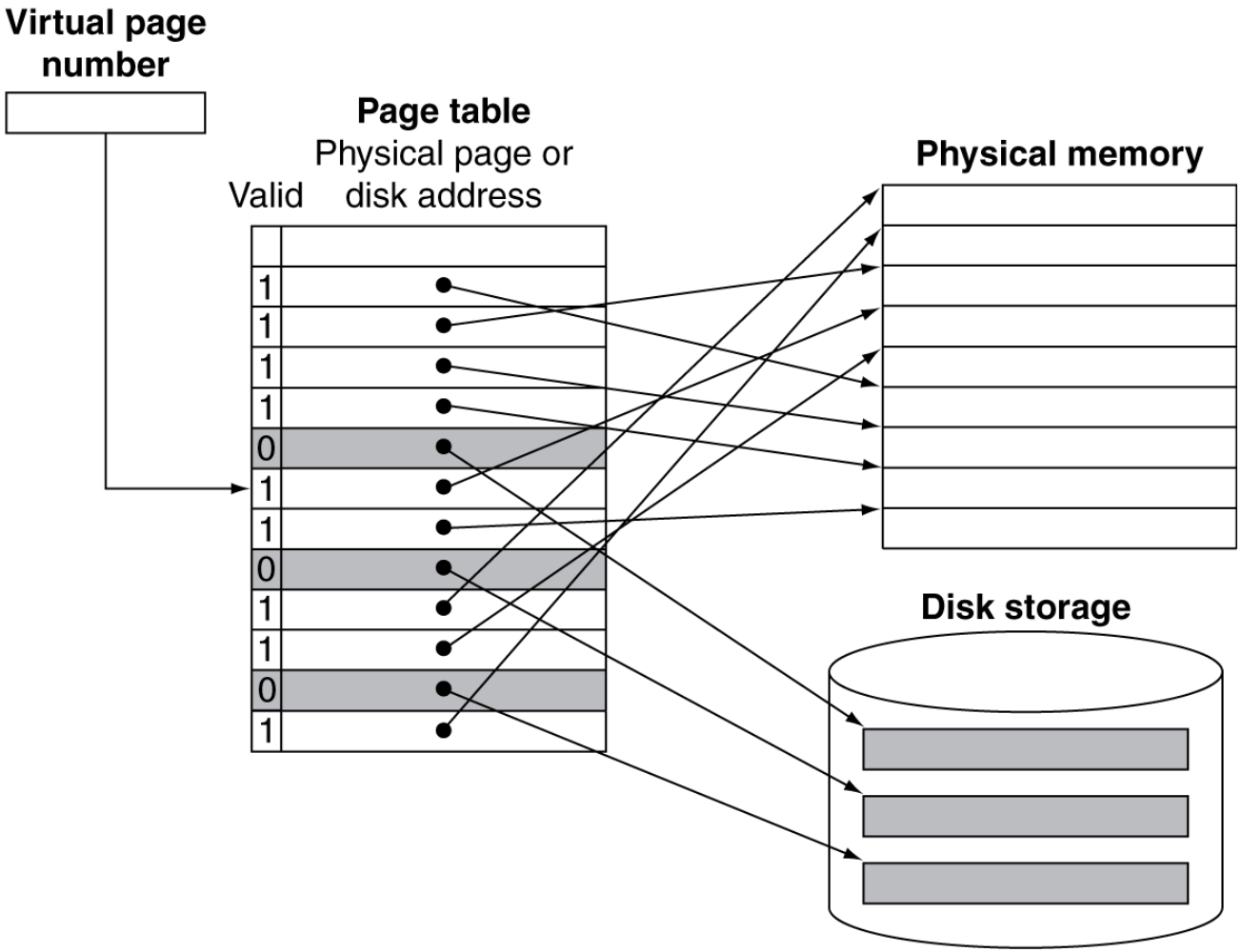
Bảng phân trang (Page Tables)

- Lưu trữ thông tin sắp xếp trang
  - Bảng ánh xạ trang 2 chiều, đánh chỉ số theo trang ảo
  - Thanh ghi ánh xạ trong CPU sẽ chỉ đến bảng ánh xạ trong bộ nhớ vật lý
- Nếu trang tồn tại trong bộ nhớ
  - PTE chứa chỉ số trang vật lý tương ứng
  - Cùng với bit trạng thái (đã tham chiếu, dirty,...)
- Nếu trang không tồn tại trong bộ nhớ
  - PTE tham chiếu đến vùng swap trên đĩa

Chuyển đổi với bảng phân trang



Ánh xạ trang (pages) lên đĩa



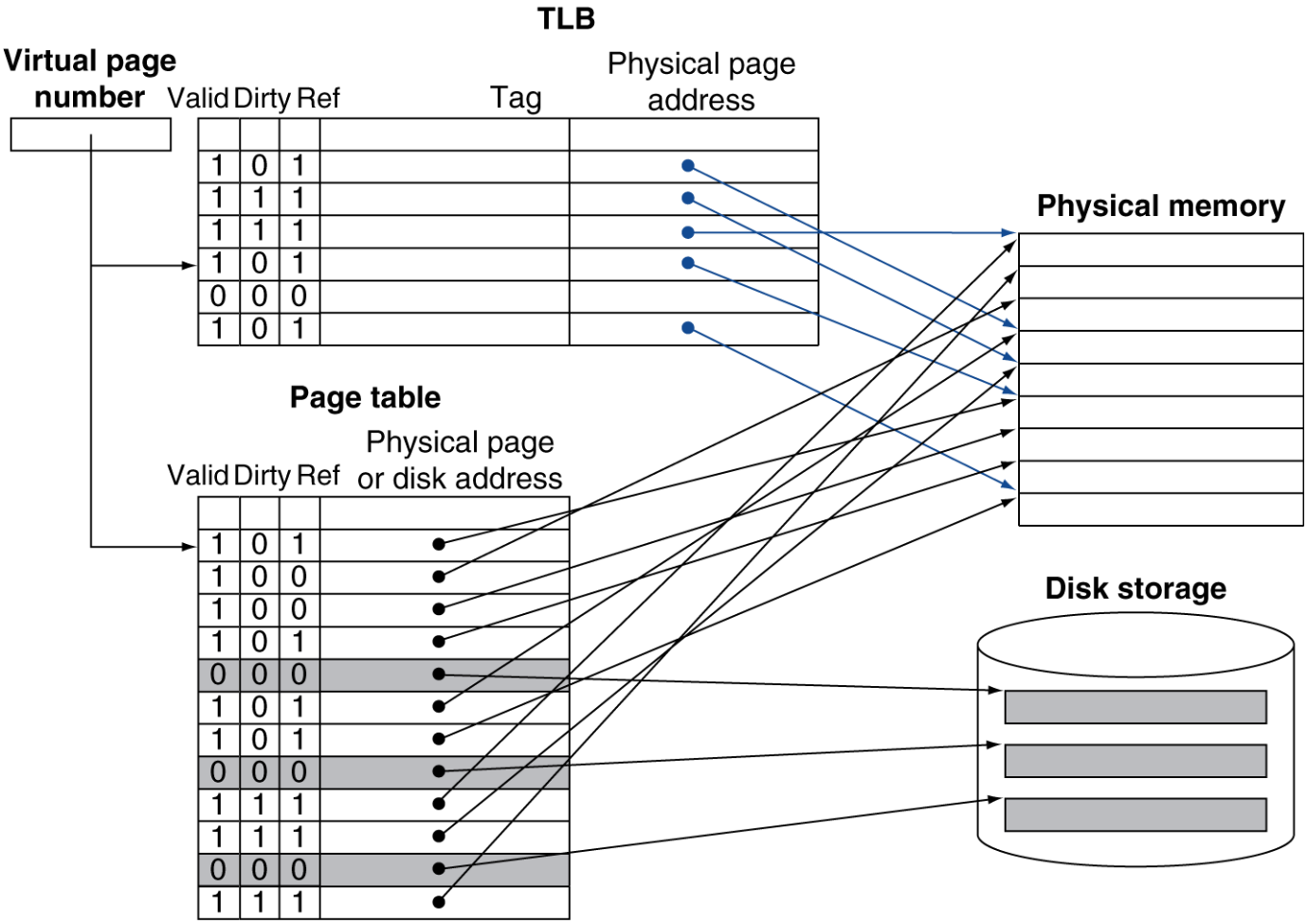
Thay thế & cập nhật

- Để giảm lỗi trang, việc thay thế thường chọn trang ít sử dụng nhất (LRU)
  - Bit tham chiếu trong bảng phân trang PTE gán lên 1 mỗi khi trang được tham chiếu
  - Hệ điều hành sẽ định kỳ xóa về 0
  - Trang có bit tham chiếu bằng 0: chưa được dùng
- Cập nhật trở lại đĩa: thời gian lên tới hàng triệu chu kỳ (phụ thuộc vào loại đĩa)
  - Cập nhật nguyên khối, không cập nhật từng từ
  - "Write through" không thực tế
  - Sử dụng "write-back"
  - "Dirty bit" trong bảng PTE = 1, khi trang thay đổi nội dung

Chuyển đổi nhanh với TLB

- TLB = Translation-lookaside buffer
- Việc chuyển đổi địa chỉ: truy xuất xảy ra 2 lần truy cập bộ nhớ
  - Truy cập để lấy được địa chỉ vật lý tương ứng PTE
  - Sau đó mới truy cập để lấy thông tin bộ nhớ
- Tuy nhiên truy cập bảng trang: tính cục bộ
  - Sử dụng bộ nhớ cache nhanh PTE trong CPU
  - Translation Look-aside Buffer (TLB)
  - Thường thì: 16–512 PTEs, 0.5–1 chu kỳ với hit, 10–100 chu kỳ với miss, 0.01%–1% miss rate
  - Misses có thể giải quyết bằng phần cứng hoặc mềm

Chuyển đổi nhanh với TLB (tt.)



Xử lý TLB Misses

- Nếu trang yêu cầu có trong bộ nhớ chính
  - Nạp PTE từ bộ nhớ chính & thử lại
  - Có thể thực hiện bằng phần cứng
    - Trở nên phức tạp với các cấu trúc bảng ánh xạ trang phức tạp
  - Hoặc bằng phần mềm
    - Xử dụng cơ chế ngoại lệ đặc biệt, xử lý chuyên dụng
- Nếu trang yêu cầu không có trong bộ nhớ chính
  - OS sẽ nạp trang yêu cầu từ đĩa và cập nhật bảng ánh xạ trang
  - Sau đó khởi động lại lệnh bị lỗi trang

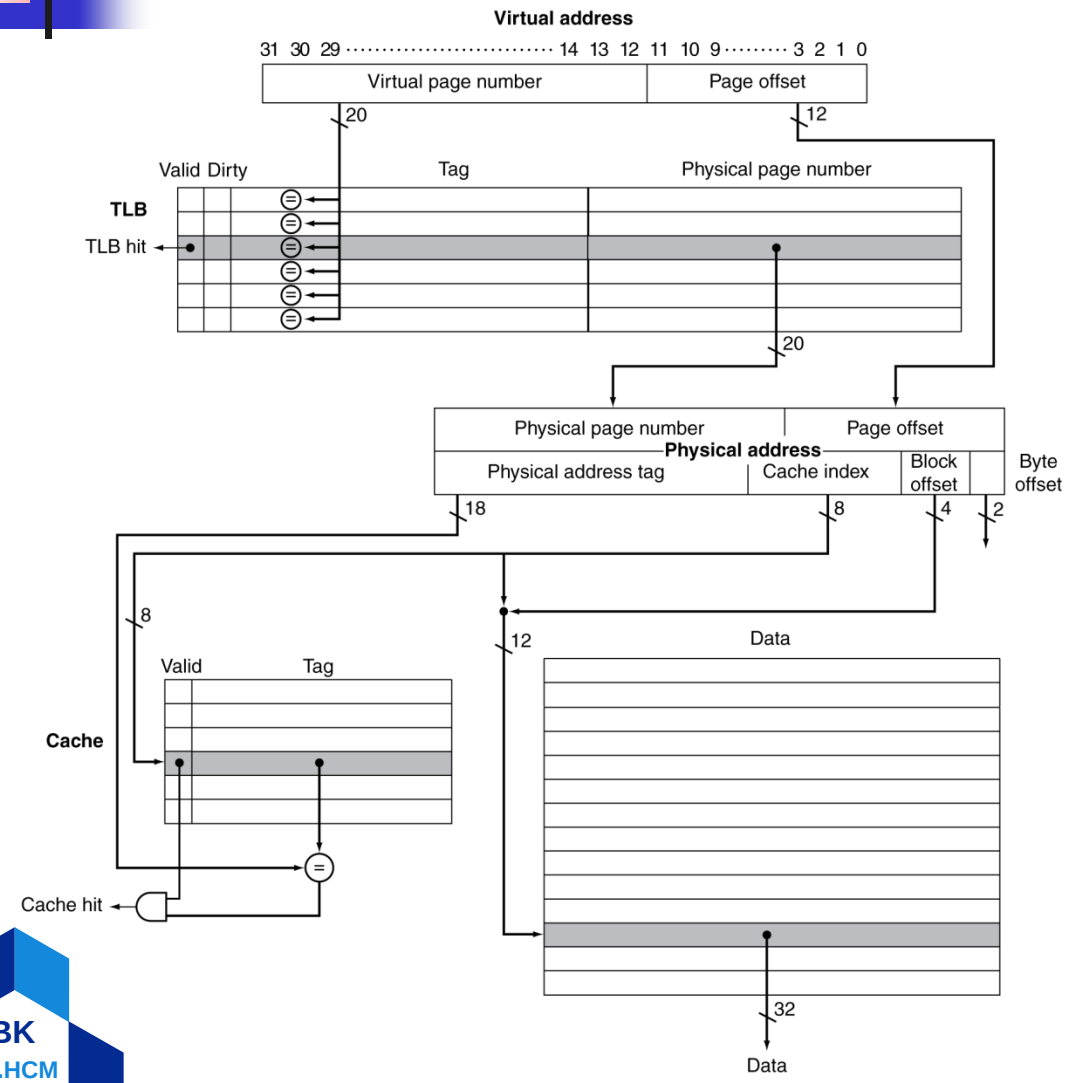
Bộ xử lý (handler) TLB Miss

- TLB miss xuất hiện → 2 trường hợp
  - Trang hiện hữu, nhưng PTE không có trong TLB
  - Trang không tồn tại trong bộ nhớ chính
- TLB mis cần nhận biết trước khi thanh ghi đích được cập nhật
  - Xuất hiện ngoại lệ
- Bộ xử lý sẽ sao chép PTE từ bộ nhớ vào TLB
  - Sau đó lệnh được khởi đọng lại
  - Nếu trang không có, lỗi trang xảy ra

Khi có lỗi trang

- Sử dụng địa chỉ bị lỗi để tìm phần tử PTE trang tương ứng
- Xác định vị trí trên đĩa
- Chọn trang trong bảng ánh xạ để thay thế
  - Nếu bị sửa đổi nhiều: cập nhật lên đĩa
- Đọc trang yêu cầu từ đĩa & Cập nhật bảng ánh xạ trang
- Khởi động quá trình chậy lại
  - Khởi động lại lệnh gây lỗi trang

Giao tiếp TLB & Cache



- If cache tag uses physical address
  - Need to translate before cache lookup
- Alternative: use virtual address tag
  - Complications due to aliasing
    - Different virtual addresses for shared physical address

Thực hiện bảo vệ bộ nhớ

- Các chương trình chạy đồng thời chia sẻ chung không gian địa chỉ ảo
  - Nhưng cần được bảo vệ, tránh truy cập lẫn nhau
  - Cần sự tham gia của hệ điều hành
- Phần cứng hỗ trợ hệ điều hành
  - Chế độ đặc quyền (Privileged supervisor mode)
  - Lệnh đặc quyền
  - Bảng ánh xạ trang và các thông tin trạng thái khác chỉ được truy cập bằng chế độ đặc quyền
  - Ngoại lệ "gọi hệ thống" (System call exception) (ví dụ: syscall in MIPS)

Cấu trúc phân tầng bộ nhớ

- Nguyên tắc chung được áp dụng cho tất cả các tầng (lớp) trong cầu trúc phân tầng bộ nhớ
  - Sử dụng thuật ngữ "cache"
- Các hoạt động tại mỗi tầng
  - Sắp đặt khối khối (Block placement)
  - Tìm kiếm khối (Finding a block)
  - Thay thế khối trong tường hợp miss
  - Chính sách cập nhật (Write policy)

Sắp đặt khối

- Xác định bởi hàm ánh xạ quan hệ
  - Ánh xạ trực tiếp (1-way associative)
    - Chỉ có 1 phương án duy nhất 1:1
  - Ánh xạ tệp *n* (n-way associative)
    - Có n cách ánh xạ (1:n)
  - Ánh xạ toàn phần (Fully associative)
    - Bất cứ trang nào
- Mối quan hệ càng cao:→ càng giảm lỗi trang
  - Gia tăng phức tạp, giá thành & thời gian truy xuất

Tìm kiếm khối

Cách ánh xạ (associativity)	Phương pháp định vị (Location method)	So sánh nhãn (Tag comparisons)
Trực tiếp	Chỉ số (index)	1
Tệp quan hệ n (n-way)	Tệp chỉ số (Set index), sau đó tìm từng thành phần trong tệp	n
Quan hệ toàn phần (Fully Associative)	Tìm toàn bộ (Search all entries)	Ánh xạ tương ứng
	Full lookup table	0

- Caches phần cứng
  - Giảm thiểu so sánh → giảm giá thành
- Bộ nhớ ảo
  - Full table lookup makes full associativity feasible
  - Benefit in reduced miss rate

Thay thế khối (Replacement)

- Lựa chọn trang thay thế khi có lỗi trang
  - Trang ít sử dụng nhất (LRU)
    - Tương đối phức tạp & phí tổn phần cứng khi mối quan hệ ánh xạ cao
  - Ngẫu nhiên
    - Gần với LRU, dễ thực hiện hơn
- Bộ nhớ ảo
  - LRU xấp xỉ với hỗ trợ bằng phần cứng

Phương thức cập nhật đĩa

- "Write-through"
  - Cập nhật cả tầng trên & dưới
  - Đơn giản việc thay thế, nhưng yêu cầu có write buffer
- "Write-back"
  - Chỉ cập nhật tầng trên
  - Cập nhật tầng thấp khi có nhu cầu thay thế
  - Cần lưu trữ nhiều trạng thái
- Bộ nhớ ảo
  - Chỉ "write-back" là khả thi với thời gian ghi đĩa

Nguồn gốc của "Misses"

- Misses bắt buộc (lúc khởi động)
  - Lần đầu tiên truy cập khối
- Miss do dung lượng (Capacity)
  - Do hạn chế dung lượng cache
  - Một khối vừa thay ra lại bị truy cập ngay sau đó
- Miss do đụng độ (aka collision misses)
  - Trong trường hợp cache quan hệ không toàn phần
  - Tranh chấp các khối trong cùng 1 tệp
  - Sẽ không xảy ra đối với cache quan hệ toàn phần vớii dung lượng tổng như nhau

Tối ưu thiết kế cache

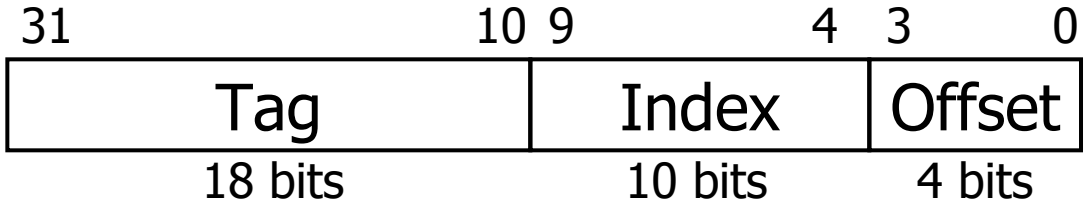
Thay đổi thiết kế	Ảnh hưởng miss rate	Hiệu ứng ngược
Tăng dung lượng cache	Giảm capacity misses	Có thể tăng thời gian truy xuất
Tăng quan hệ	Giảm conflict misses	Có thể tăng thời gian truy xuất
Tăng dung lượng khối	Giảm compulsory misses	Tăng miss penalty. For very large block size, may increase miss rate due to pollution.

Hỗ trợ tập lệnh

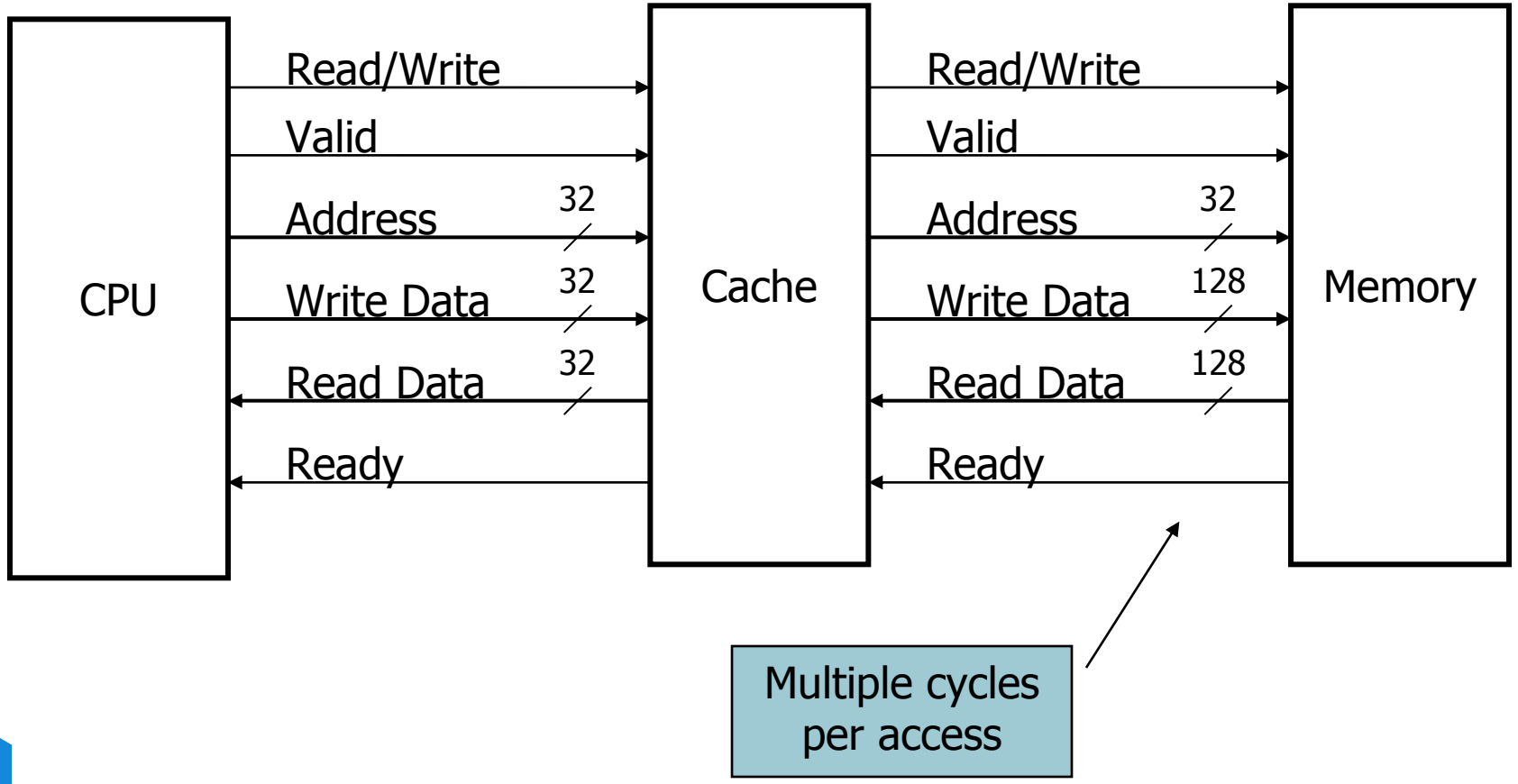
- Chế độ người dùng & hệ thống
- Các lệnh đặc dụng (privileged instructions) chỉ có ở chế độ hệ thống
  - Bẫy hệ thống khi có sự chuyển từ chế độ người dùng sang hệ thống
- Các tài nguyên vật lý chỉ truy cập được với những lệnh đặc dụng
  - Kể cả bảng ánh xạ trang, đ/khiển ngắt quãng, Thanh ghi I/O

Điều khiển Cache

- Ví dụ đặc tính cache
  - Ánh xạ trực tiếp, write-back, write allocate
  - Kích thước khối: 4 từ (words) = (16 bytes)
  - Kích thước cache: 16 KB (1024 blocks)
  - Địa chỉ 32-bit byte
  - Valid bit & dirty bit cho mỗi khối
  - Blocking cache
    - CPU waits until access is complete

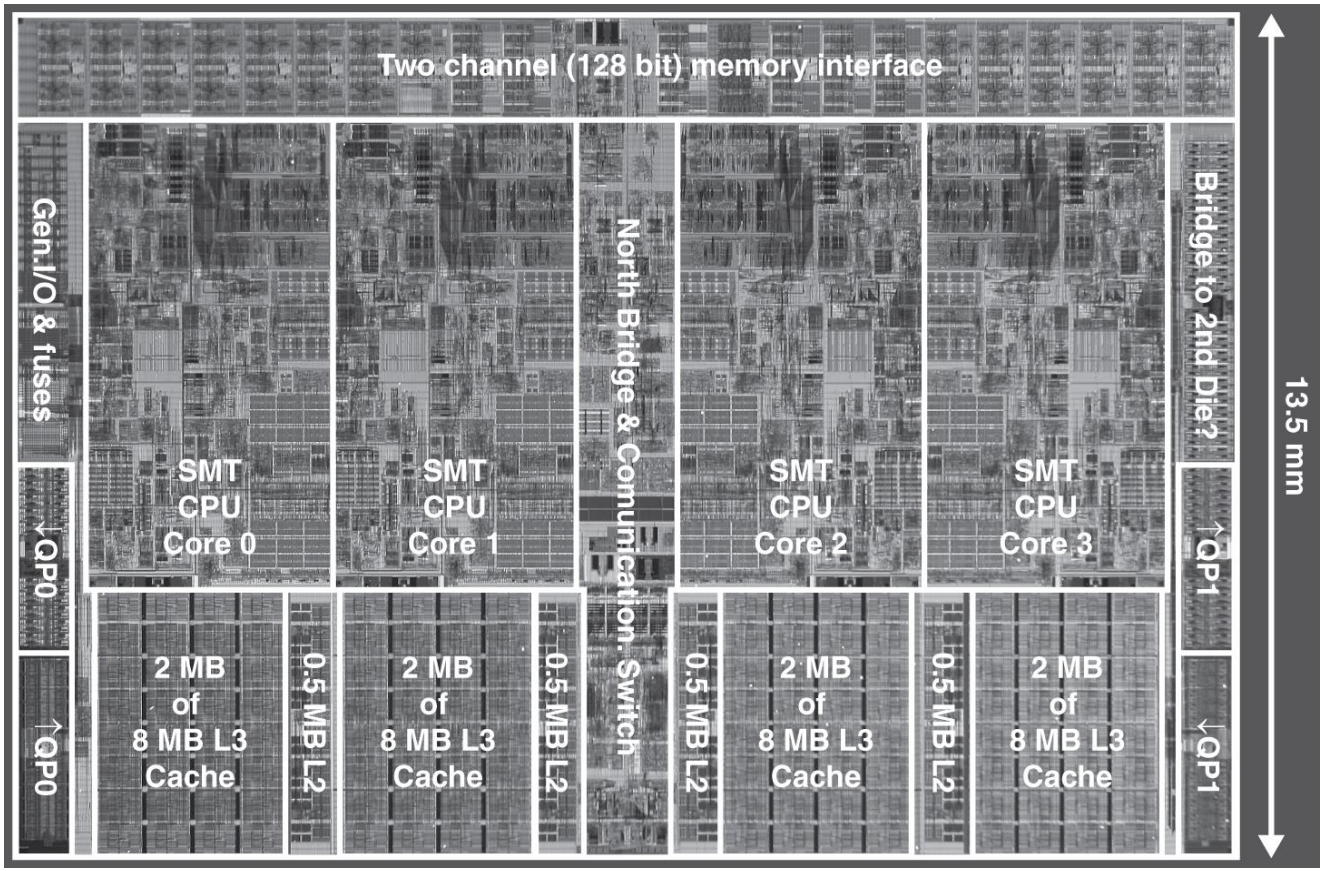


Các tín hiệu giao tiếp



Cache nhiều cấp on chip

Intel Nehalem 4-core processor



Per core: 32KB L1 I-cache, 32KB L1 D-cache, 512KB L2 cache

Tổ chức TLB cache cấp 2

	Intel Nehalem	AMD Opteron X4
Virtual addr	48 bits	48 bits
Physical addr	44 bits	48 bits
Page size	4KB, 2/4MB	4KB, 2/4MB
L1 TLB (per core)	L1 I-TLB: 128 entries for small pages, 7 per thread (2×) for large pages L1 D-TLB: 64 entries for small pages, 32 for large pages Both 4-way, LRU replacement	L1 I-TLB: 48 entries L1 D-TLB: 48 entries Both fully associative, LRU replacement
L2 TLB (per core)	Single L2 TLB: 512 entries 4-way, LRU replacement	L2 I-TLB: 512 entries L2 D-TLB: 512 entries Both 4-way, round-robin LRU
TLB misses	Handled in hardware	Handled in hardware

Tổ chức Cache 3 cấp

	Intel Nehalem	AMD Opteron X4
L1 caches (per core)	L1 I-cache: 32KB, 64-byte blocks, 4-way, approx LRU replacement, hit time n/a L1 D-cache: 32KB, 64-byte blocks, 8-way, approx LRU replacement, write-back/allocate, hit time n/a	L1 I-cache: 32KB, 64-byte blocks, 2-way, LRU replacement, hit time 3 cycles L1 D-cache: 32KB, 64-byte blocks, 2-way, LRU replacement, write-back/allocate, hit time 9 cycles
L2 unified cache (per core)	256KB, 64-byte blocks, 8-way, approx LRU replacement, write-back/allocate, hit time n/a	512KB, 64-byte blocks, 16-way, approx LRU replacement, write-back/allocate, hit time n/a
L3 unified cache (shared)	8MB, 64-byte blocks, 16-way, replacement n/a, write-back/allocate, hit time n/a	2MB, 64-byte blocks, 32-way, replace block shared by fewest cores, write-back/allocate, hit time 32 cycles

n/a: data not available

Hạn chế phí tổn Mis (Penalty)

- Trả về "từ" được yêu cầu trước tiên
  - Sau đó nạp tiếp phần còn lại của khối
- Xử lý Non-blocking miss
  - Hit under miss: allow hits to proceed
  - Mis under miss: allow multiple outstanding misses
- Nạp trước bằng phần cứng: Lệnh & Dữ liệu
- Opteron X4: bank interleaved L1 D-cache
  - Two concurrent accesses per cycle

Kết luận

- Bộ nhớ có tốc độ truy xuất nhanh → Nhỏ ;
  Bộ nhớ có chứa dung lượng lớn → Chậm
  - Mục tiêu mong muốn: nhanh và lớn ☹
  - Cơ chế Caching giải quyết vấn đề ☺
- Nguyên tắc cục bộ
  - Chương trình sử dụng 1 phần nhỏ không gian bộ nhớ
- Tổ chức bộ nhớ theo kiến trúc tầng
  - L1 cache ↔ L2 cache ↔ … ↔ DRAM (bộ nhớ) ↔ disk
- Thiết kế hệ thống bộ nhớ rất quan trọng đối với đa xử lý