Lecture 22
Set-Associative Cache

School of Computer Science and Engineering
Soongsil University

5. Large and Fast: Exploiting Memory Hierarchy

5.1 Introduction

5.2 Memory Technologies

5.3 The Basics of Caches

5.4 Measuring and Improving Cache Performance

5.5 Dependable Memory Hierarchy

5.6 Virtual Machines

5.7 Virtual Memory

5.8 A Common Framework for Memory Hierarchies

5.9 Using a Finite-State Machine to Control a Simple Cache

5.10 Parallelism and Memory Hierarchies: Cache Coherence

5.11 Parallelism and Memory Hierarchy: RAID

5.12 Advanced Material: Implementing Cache Controllers

5.13 Real Stuff: The ARM Cortex-A8 and Intel Core i7 Memory Hierarchies

5.4 Measuring and Improving Cache Performance

- **Two techniques for improving cache performance**
  1. Using associativity
  2. Multilevel caching
- **CPU time with cache**
  - CPU time = (CPU execution clock cycles

    + memory-stall clock cycles) × clock cycle time

    ,where CPU execution time includes cache hit time
  - Main cause of memory-stall is cache miss

Memory Stalls

- Memory-stall clock cycles

  = read-stall clock cycles + write-stall clock cycles
- Read-stall cycles

  = (reads/program) x read miss rate

  x read miss penalty
- Write-stall cycles

  = (writes/program) x write miss rate

  x write miss penalty + write buffer stalls

Simplification

- **Assumptions**
  - read miss penalty = write miss penalty
  - Read miss rate = write miss rate
  - negligible write buffer stalls
- **Memory-stall clock cycles**

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

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

$$= \text{IC} \times \frac{\text{Memory accesses}}{\text{Instruction}} \times \text{Miss rate} \times \text{Miss penalty}$$

In a Harvard Architecture RISC

- **Memory-stall clock cycles**

$$
\begin{aligned}
&= \frac{\text{Memory accesses}}{\text{Program}} \times \text{Miss rate} \times \text{Miss penalty} \\
&= \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Memory accesses}}{\text{Instruction}} \times \frac{\text{Misses}}{\text{Memory access}} \times \text{Miss penalty} \\
&= \text{IC} \times (1 \times \text{I - cache miss rate} \\
&\qquad + \text{Frequency of memory reference instructions} \times \text{D - cache miss rate)} \\
&\quad \times \text{Miss penalty}
\end{aligned}
$$

Example: Calculating Cache Performance

- **How much faster with a perfect cache?**
  - Instruction cache miss rate = 2%, Data cache miss rate = 4%
  - CPI = 2 without any memory stalls, Miss penalty = 100 cycles
  - Frequency of all loads and stores = 36%

[Answer]

- Instruction count = I
- Instruction miss cycles = I × 2% × 100 = 2.00 × I
- Data miss cycles = I × 36% × 4% × 100 = 1.44 × I
- Total stall cycles = 2.00 × I + 1.44 × I = 3.44 × I
- Thus, CPI with memory stalls = 2 + 3.44 = 5.44
- Performance with perfect cache is better by 5.44 / 2 = **2.72**

When CPI = 1, speedup = (1+3.44)/1 = **4.44**

Reducing Cache Misses by More Flexible Placement of Blocks

- **3 placement schemes**
  1. Direct mapping
  2. Set-associative mapping
  3. (Fully) associative mapping
- **Direct mapped cache**
  - Direct mapping to a single location in the upper level
- **Fully associative cache**
  - A memory block can be placed anywhere in the cache.
  - Special hardware for parallel comparison
  - CAM (content addressable memory) = associative memory

n-Way Set Associative Cache

- Each block can be placed in one of n locations.
- Set size = n (blocks)

  Number of sets = cache size / n
- Each block in the memory maps to a unique set in the cache given by the index field.

  → A block is direct mapped into a set.
- A block can be placed in any element of that set.

  → All the blocks in the set are searched for a match.
- Increased degree of associativity (=n)

  → decreased miss rate

  increased hit time

Comparison of Three Block Placement Policies

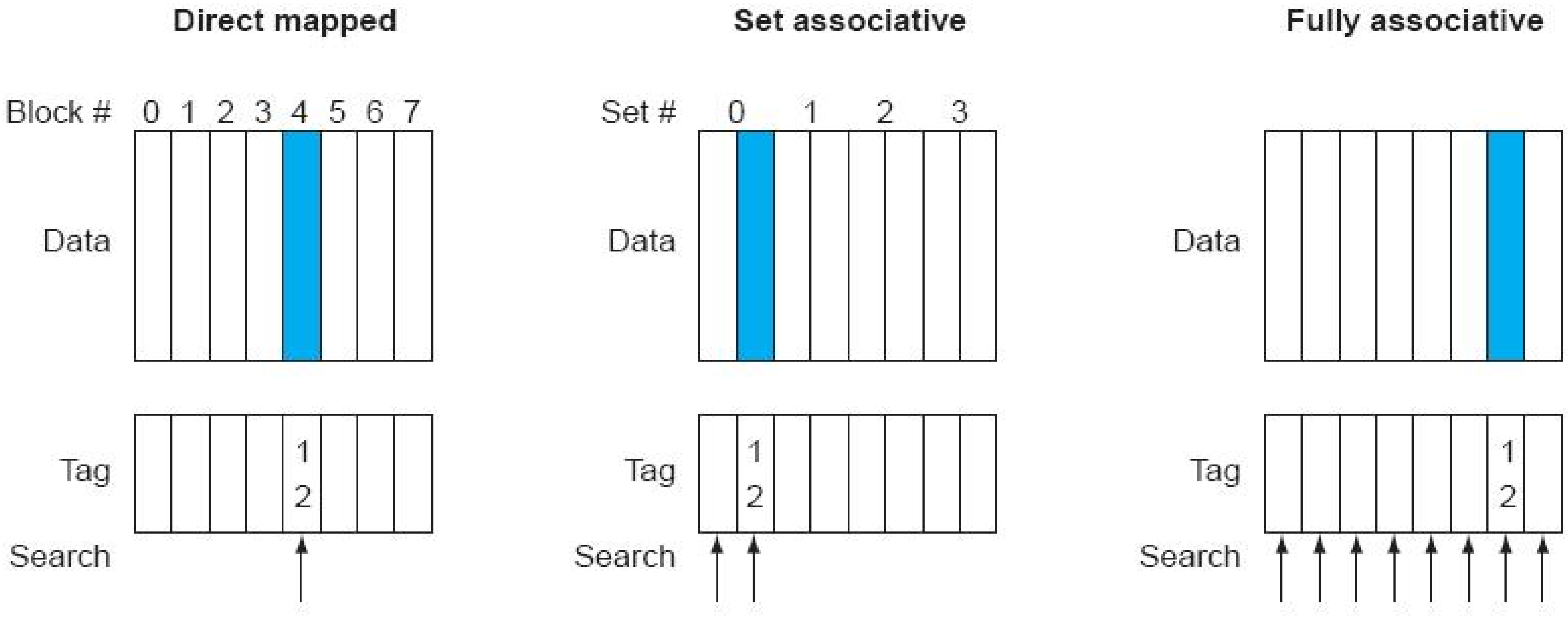
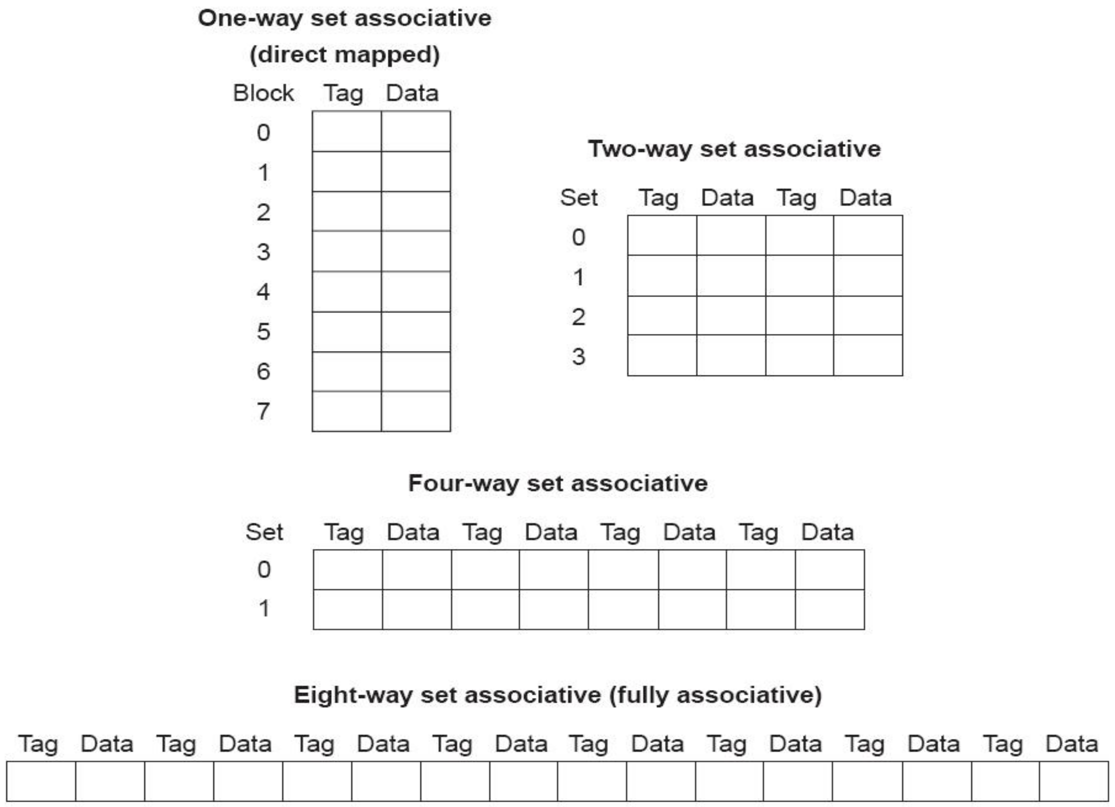


Figure 5.14

8-Block Cache Example

- **When cache size = m blocks**
  - 1-way set associative mapping = direct mapping
  - m-way set associative mapping = fully associative mapping



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								

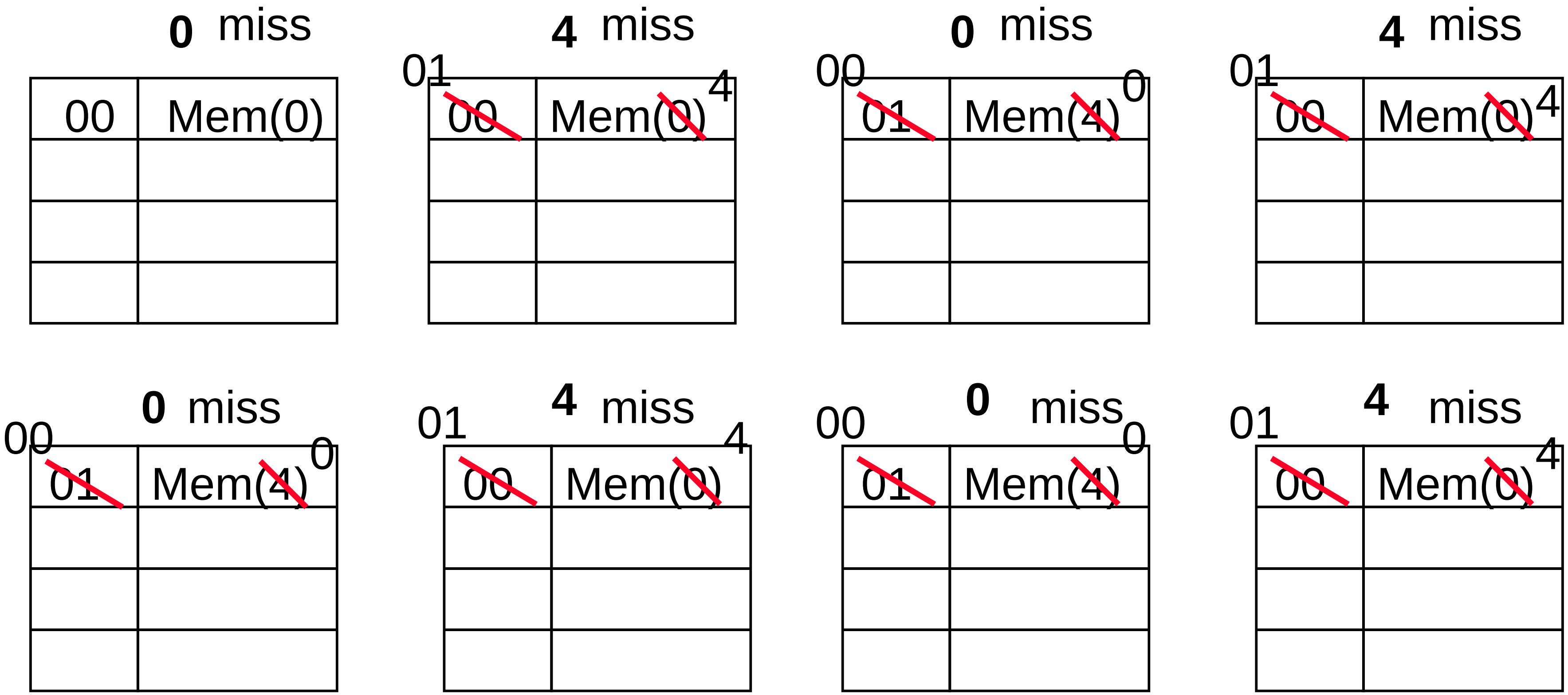
Eight-way set associative (fully associative)

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

Figure 5.15

Conflict Misses

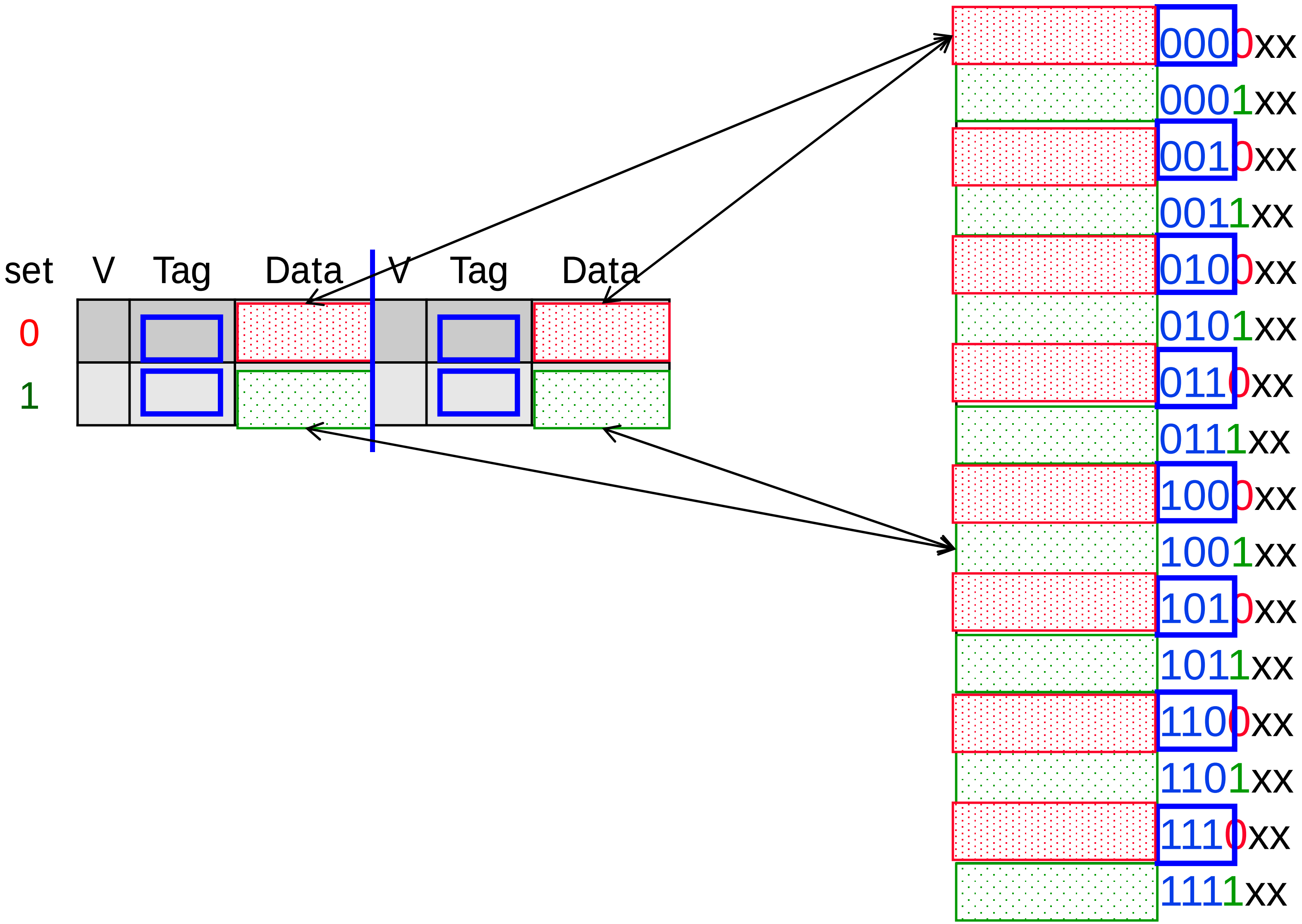
- **Reference string: 0 4 0 4 0 4 0 4**



● 8 requests, 8 misses

- Ping pong effect due to conflict misses
  - Two memory locations that map into the same cache block

2-Way Set-Associative Cache



Removing Conflict Misses

- **Reference string: 0 4 0 4 0 4 0 4**

0 miss

set	V	Tag	Data	V	Tag	Data
0	1	000	Mem(0)			
1						

4 miss

set	V	Tag	Data	V	Tag	Data
0	1	000	Mem(0)	1	010	Mem(4)
1						

0 hit

set	V	Tag	Data	V	Tag	Data
0	1	000	Mem(0)	1	010	Mem(4)
1						

4 hit

set	V	Tag	Data	V	Tag	Data
0	1	000	Mem(0)	1	010	Mem(4)
1						

● 8 requests, 2 misses

- Solves the ping pong effect in a direct mapped cache
  - Two memory locations that map into the same cache set can co-exist

Example: Misses and Associativity in Caches

- Three small caches, each consisting of four one-word blocks
    - Direct mapped
    - Two-way set associative
    - Fully associative
- Replacement policy = LRU
- Access sequence = (0, 8, 0, 6, 8)
- **Calculate the number of misses for each cache.**

[Answer-1/3]

1. Direct mapped cache : 5 misses

Block address	Cache block
0	(0 mod 4) = 0
4	(4 mod 4) = 0
6	(6 mod 4) = 2
8	(8 mod 4) = 0

Block addr.	H/M	Contents of cache blocks			
		0	1	2	3
0	Miss	M[0]			
8	Miss	M[8]			
0	Miss	M[0]			
6	Miss	M[0]		M[6]	
8	Miss	M[8]		M[6]	

[Answer-2/3]

2. 2-way set associative cache : 4 misses

Block address	Cache block
0	(0 mod 2) = 0
4	(4 mod 2) = 0
6	(6 mod 2) = 0
8	(8 mod 2) = 0

Block addr.	H/M	Contents of cache blocks			
		Set 0		Set 1	
0	Miss	M[0]			
8	Miss	M[0]	M[8]		
0	Hit	M[0]	M[8]		
6	Miss	M[0]	M[6]		
8	Miss	M[8]	M[6]		

[Answer-3/3]

3. Fully associative cache : 3 misses

Block addr.	H/M	Contents of cache blocks			
0	Miss	M[0]			
8	Miss	M[0]	M[8]		
0	Hit	M[0]	M[8]		
6	Miss	M[0]	M[8]	M[6]	
8	Hit	M[0]	M[8]	M[6]	

Large Degree of Associativity

- Decreased miss rate

  but, becomes smaller when cache size becomes larger

- Data cache miss rate of Intrinsity FastMATH
  - 10 SPEC2000 programs

Associativity	Data miss rate
1	10.3%
2	8.6%
4	8.3%
8	8.1%

Figure 5.16

Performance

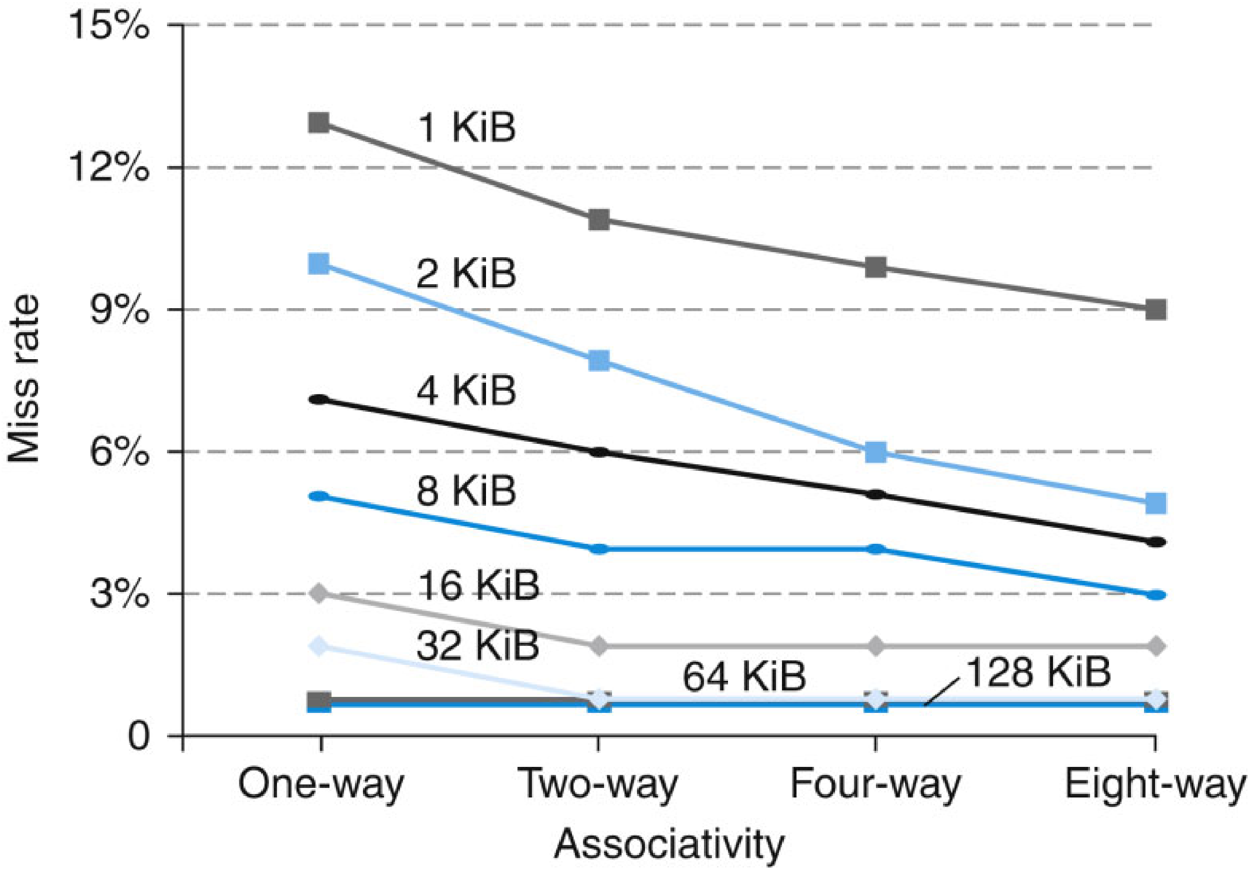
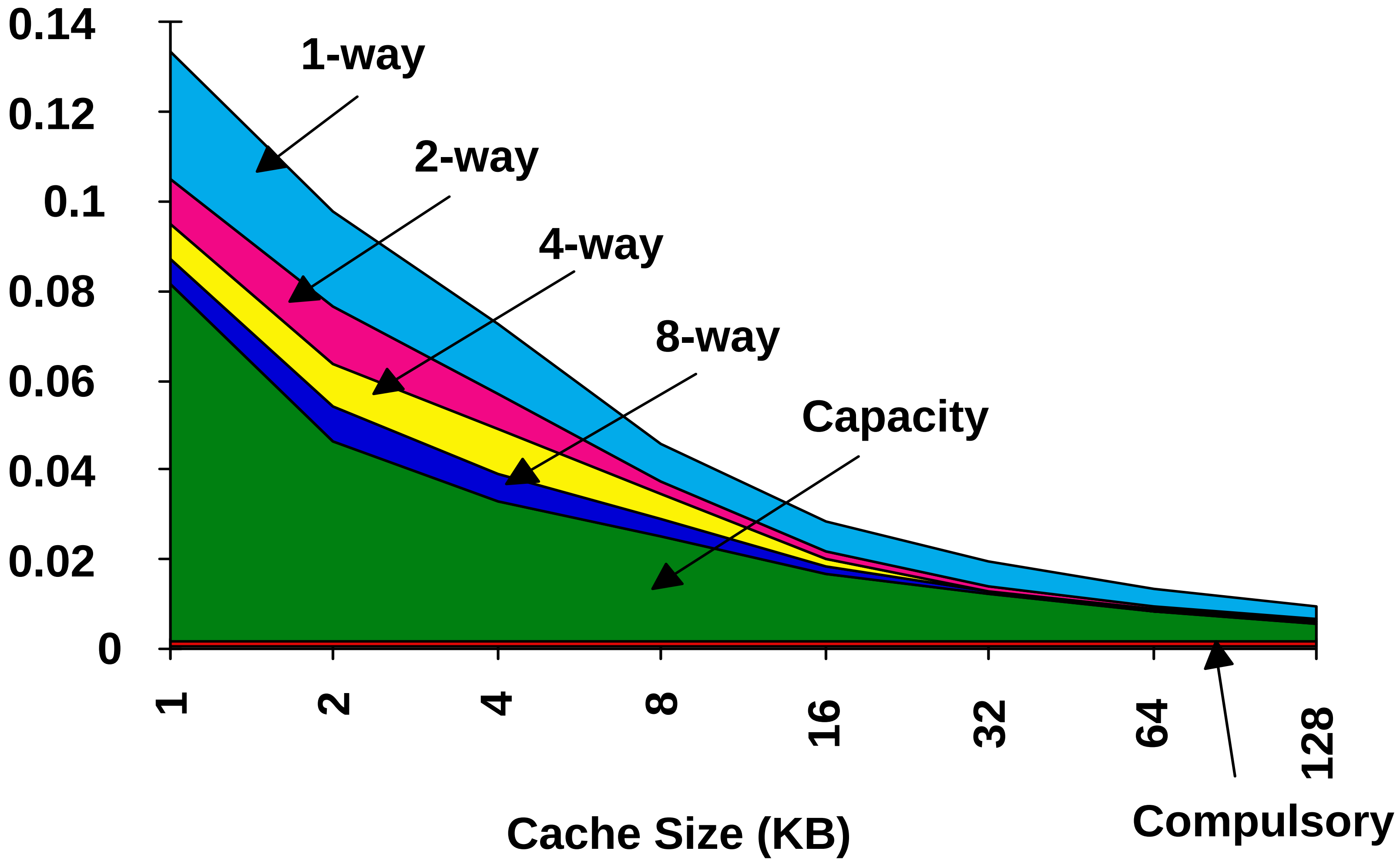


Figure 5.36

Miss Rate for SPEC92 Benchmarks



Locating a Block in the Cache

- **Set-associative mapping**
  - The three portions of an address: Figure 5.17

tag	index	Block offset

  - Index ... used to select the set
  - Tags ... searched in parallel to determine hit or miss
- **Increasing the associativity by a factor of two**
  - Doubling the number of blocks per set

    -> doubling the number of comparators
  - Halving the number of sets

    -> decrease the size of the index by 1 bit

    increase the size of the tag by 1 bit

4-Way Set-Associative Cache

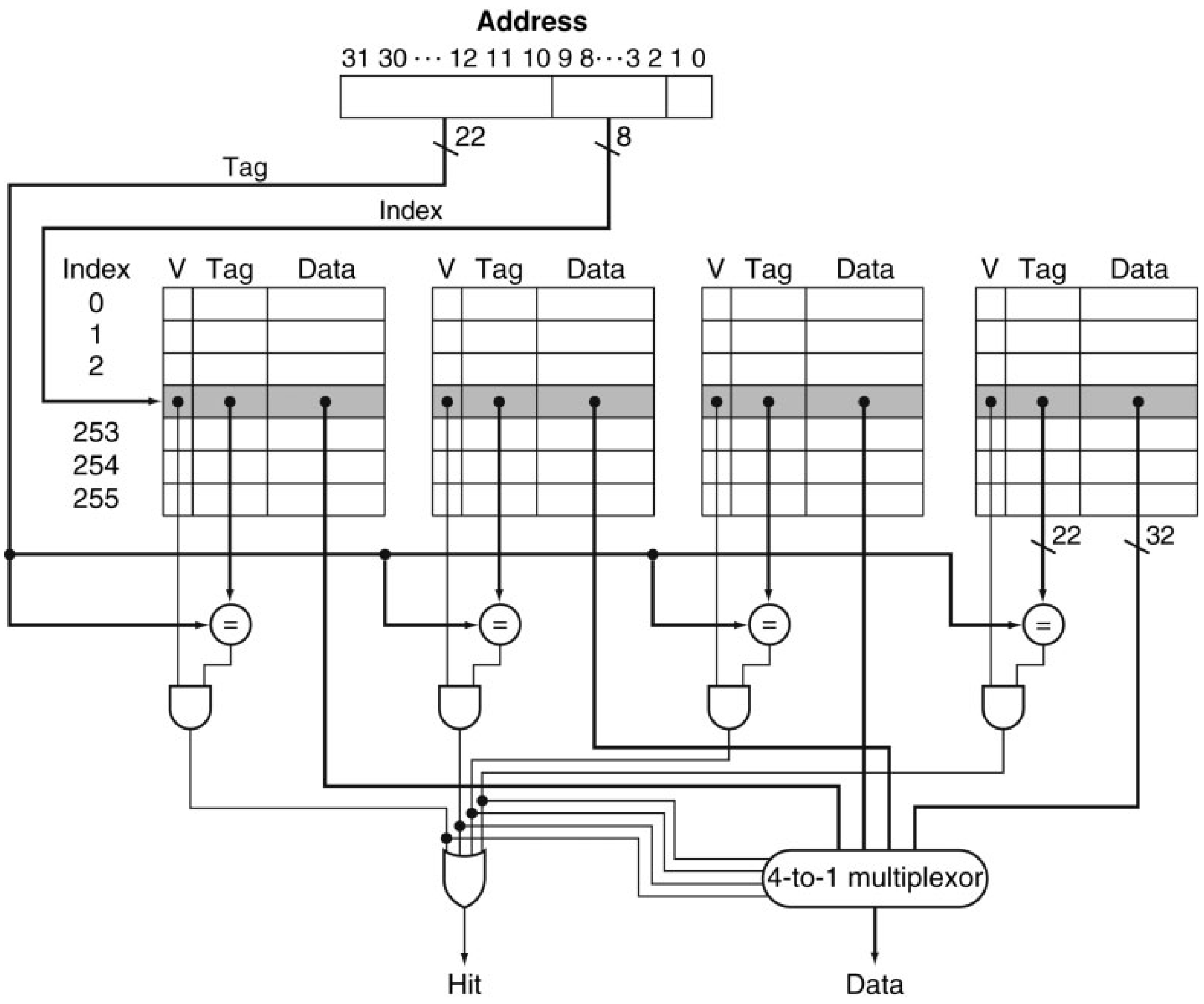
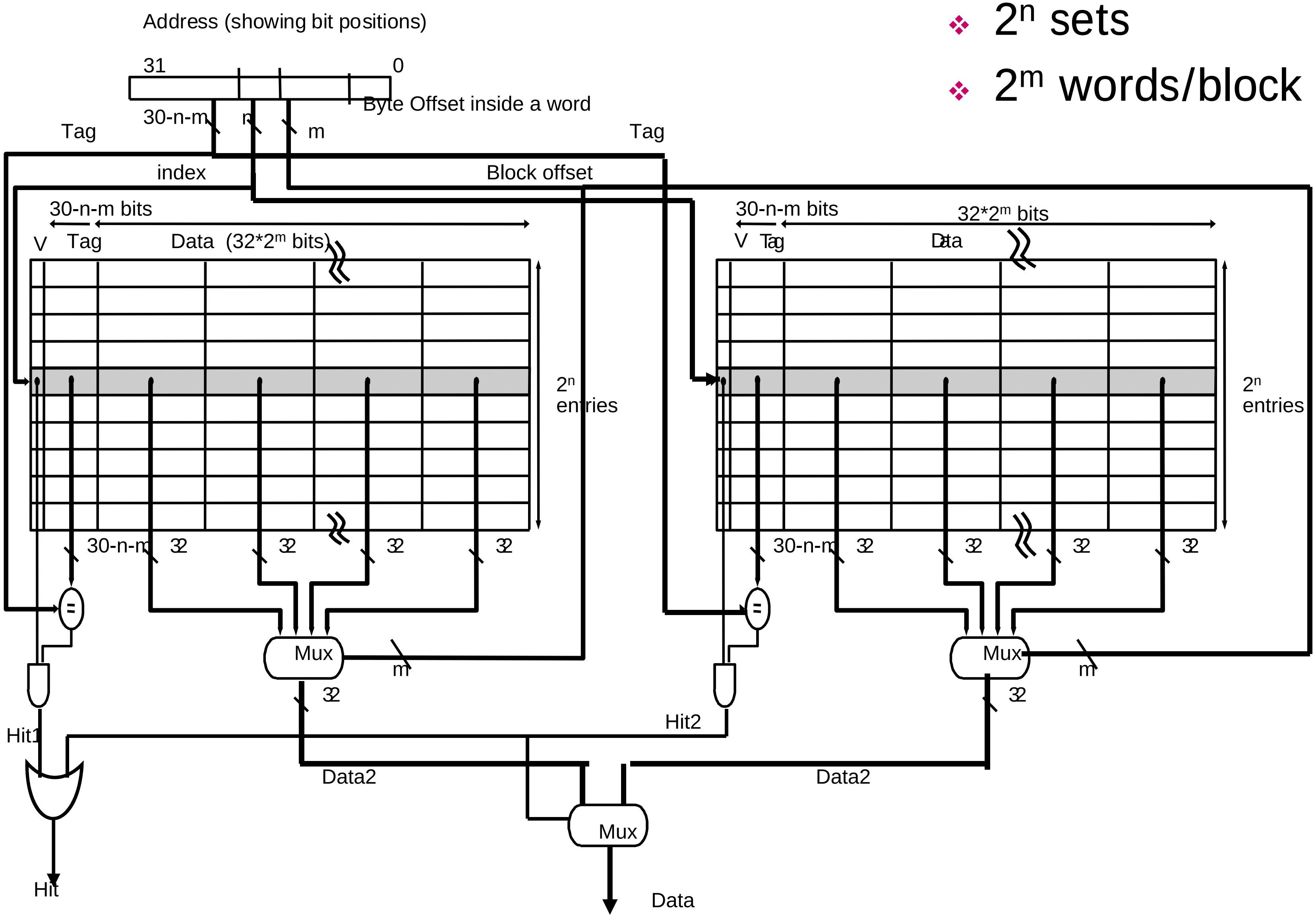


Figure 5.18

Set-Associative Cache with 2^m-Word Block

- 2^n sets
- 2^m words/block


Address (showing bit positions)
31
0
Byte Offset inside a word
30-n-m
n
m
Tag
Tag
index
Block offset
30-n-m bits
V
Tag
Data (32*2m bits)
2n entries
30-n-m
32
32
32
32
Mux
m
32
30-n-m bits
32*2m bits
V
Tag
Data
2n entries
30-n-m
32
32
32
32
Mux
m
32
Hit1
Hit2
Data2
Data2
Mux
Hit
Data