How we store in cache

How to cut cost in cache storing

Size of cache

Book

Lecture – 20

Handling Cache Misses

Steps Taken on an Instruction Cache Miss:

1. Send the original PC value (PC-4) to the memory.
2. Instruct main memory to perform a read and wait for the memory to complete it’s access.
3. Write the cache entry, putting the data from memory in the data portion of the entry, writing the upper bits of the address into the tag field, and turning the valid bit on.
4. Restart the instruction execution at the first step, which will refetch the instruction, this time finding it in cache.

Handling Writes

Write Through:

- Writes always update both the cache and the memory, ensuring that data is always consistent between two.
- Memory write operation will take longer time and slow down the processor.
- Let CPI is 1.0 without cache miss and about 10% of the instruction is store and every write require 100 clock cycle, then CPI = 1.0 + 100 * 10% = 11.
- Solution to this problem is **write buffer**.

Handling Writes

Write buffer:

- Write buffer stores data while it is waiting to be written to memory.
- After writing the data into the cache and into the write buffer, the processor can continue execution.
- When a write to main memory completes, the entry in the write buffer is freed.
- The processor must stall for the write buffer to become empty.
- If the rate of generating write by the processor is larger than the rate at which the memory can accept, then it will create problem.

Handling Writes

Write Back:

- It handles write by updating values only to the block in the cache, then writing the modified block to the lower level of the hierarchy when the block is replaced.
- It improves performance.
- Implementation is difficult.

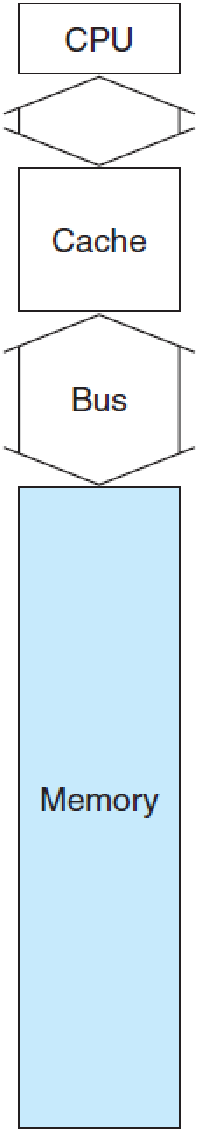
Designing Memory System to Support Cache

- We can reduce the miss penalty by Increasing the memory bandwidth.

- The clock rate of the bus is usually much slower than the processor, by as much as a factor of 10. The speed of this affects the miss penalty.

- Assume the following memory access time:
  - 1 memory bus clock cycle to send the address.
  - 15 memory bus clock cycles for each DRAM access initiated.
  - 1 memory bus clock cycle to send a word of data.

One-Word-Wide Memory Organization



Suppose,

Cache block = 4 words.

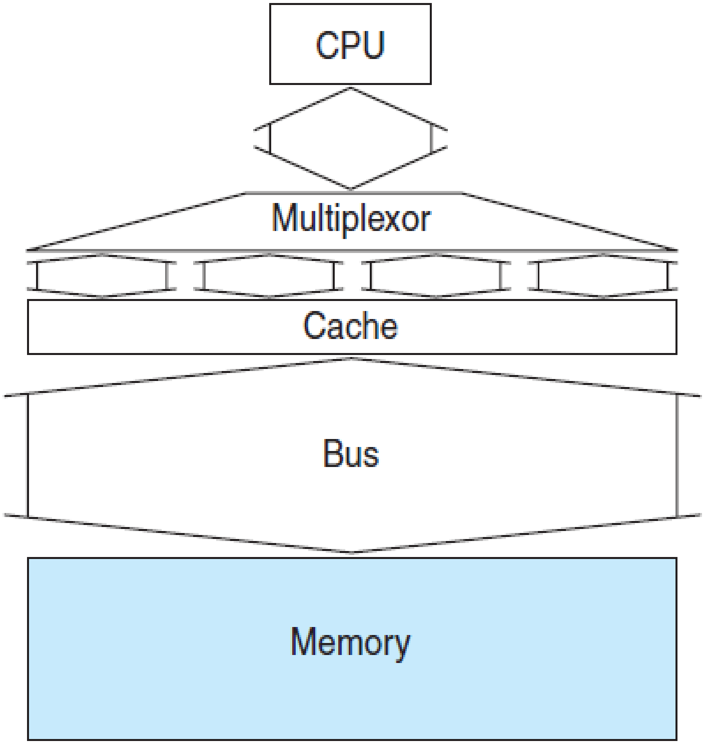
Memory width = 1 word.

Then,

Miss penalty = 1+4*15+4*1 = 65 memory bus clock cycles.

The number of bytes transferred per bus clock cycle for a single miss = 4*4/65 = 0.25

Wide Memory Organization



Suppose,

Cache block = 4 words.

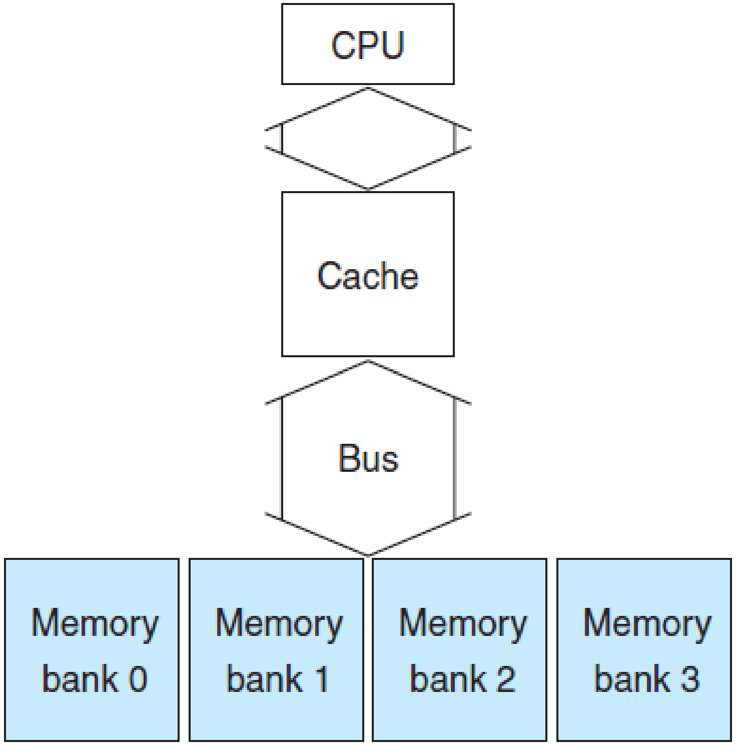
Memory width = 2 words.

Then,

Miss penalty = 1+2*15+2*1 = 33 memory bus clock cycles.

The number of bytes transferred per bus clock cycle for a single miss = 4*4/33 = 0.48

Interleaved Memory Organization



Suppose,

Cache block = 4 words.

Memory width = 4 words.

Then,

Miss penalty = 1+1*15+4*1 = 20 memory bus clock cycles.

The number of bytes transferred per bus clock cycle for a single miss = 4*4/20 = 0.80

✓ It also improves write miss.

Reducing Cache Misses

Two ways:

1. Fully associative cache
2. Set associative cache

Fully Associative Cache

- A memory block can be placed in any location in the cache.
- To find a given block all the entries in the cache must be searched.
- A comparator is associated with each cache entry increasing the hardware cost.
- This placement is practical only for caches with small number of blocks.

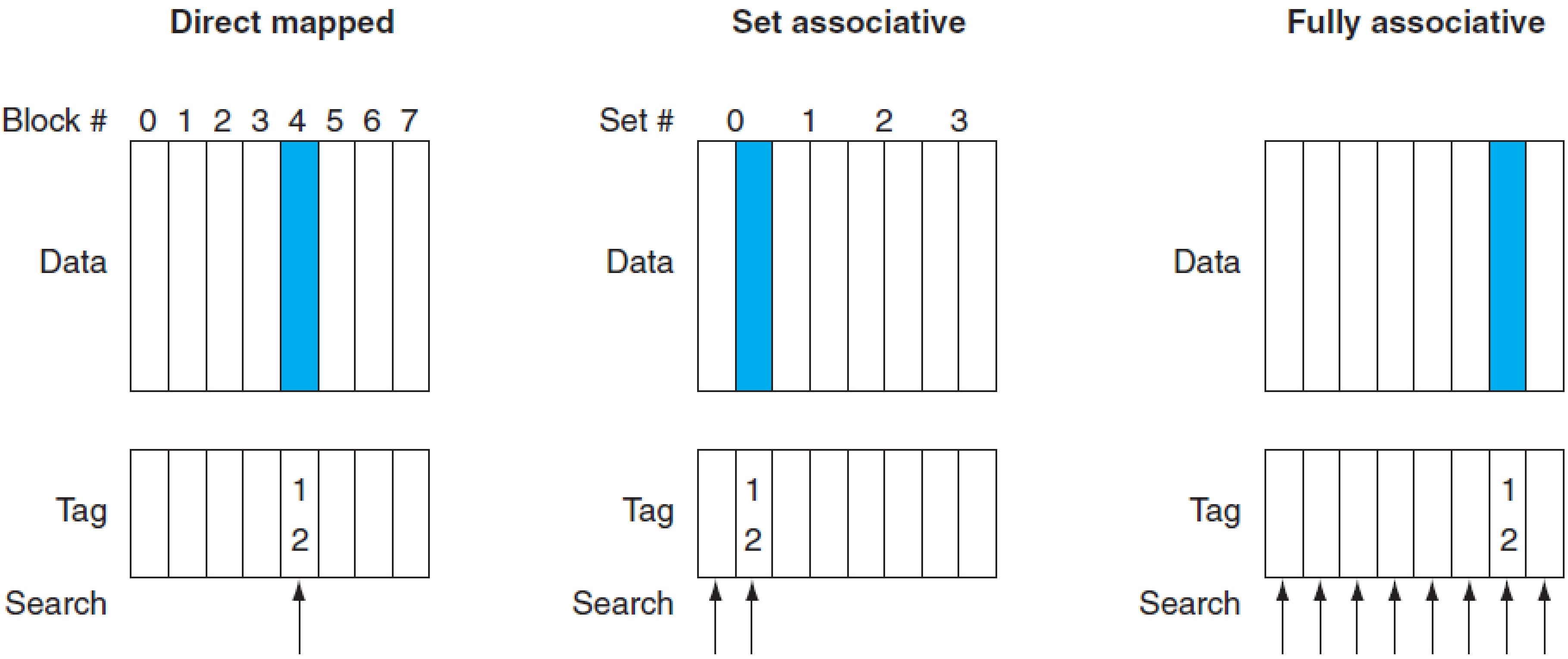
Set Associative Cache

- There are a fixed number of locations where each block can be placed.
- A set-associative cache with n locations for a block is called an n-way set associative cache.
- A n-way set associative cache consists of a number of sets, each of which consists of n blocks.
- Each block in the memory maps to a unique set in the cache given by the index field and a block can be placed in any elements of the set.

Set Associative Cache

- In combines both direct mapping and fully associative placement. A block is directly mapped into a set, and then all the blocks in the set are searched for a match.
- Set number = (Block number) modulo (Number of sets in the cache).
- The advantage of increasing the degree of associativity is that it usually decreases the miss rate.
- The main disadvantage is it increases the hit time.

Memory Block Mapping in the Cache



Cache Configuration

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								

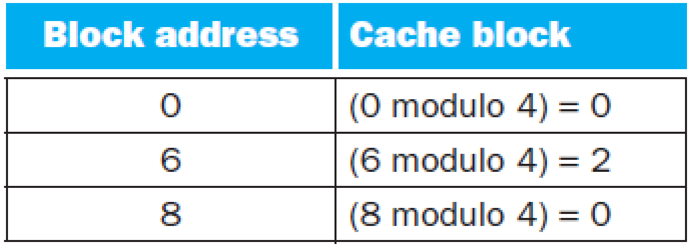
Eight-way set associative (fully associative)

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

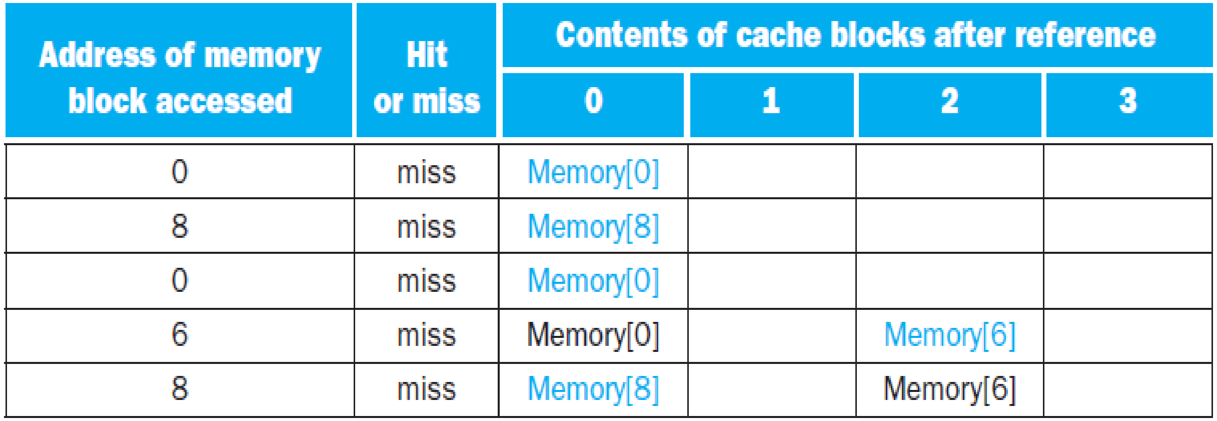
Memory Accesses in Cache

- Cache size = four one-word blocks.
- Block addresses = 0, 8, 0, 6, 8

Direct Mapped Cache:



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

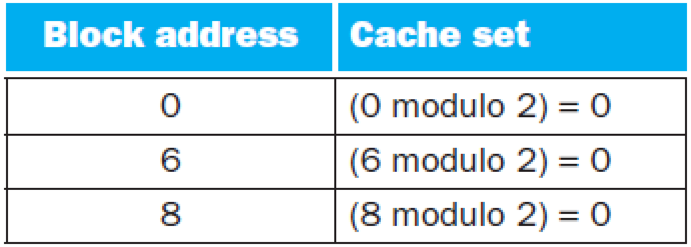


Address of memory block accessed	Hit or miss	Contents of cache blocks after reference			
		0	1	2	3
0	miss	Memory[0]			
8	miss	Memory[8]			
0	miss	Memory[0]			
6	miss	Memory[0]		Memory[6]	
8	miss	Memory[8]		Memory[6]	

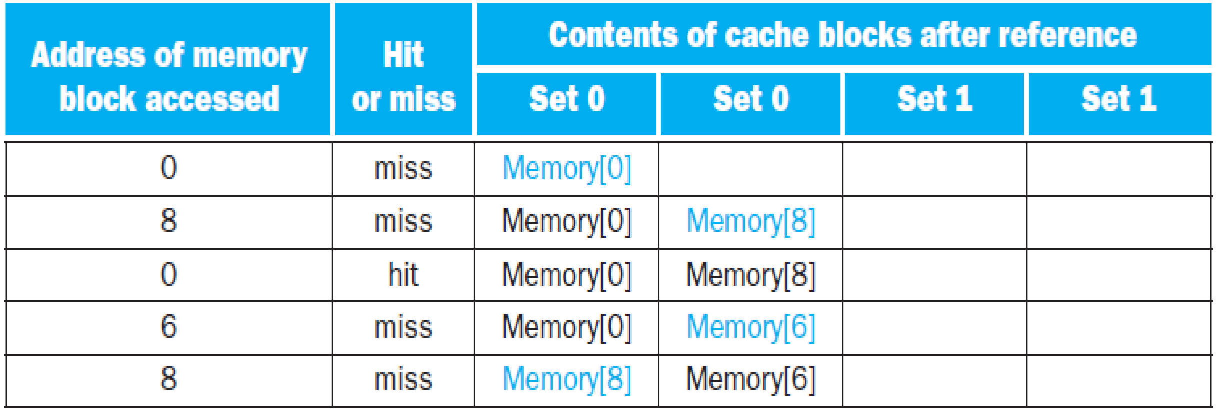
Five misses for five accesses.

Memory Accesses in Cache

Two Way Set Associative:



Block address	Cache set
0	(0 modulo 2) = 0
6	(6 modulo 2) = 0
8	(8 modulo 2) = 0

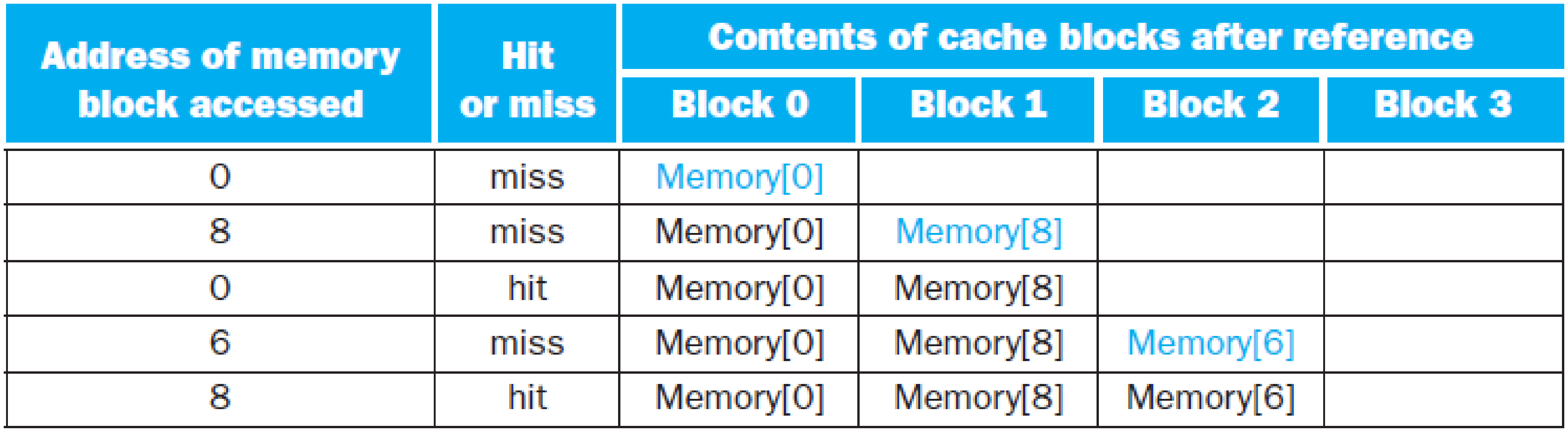


Address of memory block accessed	Hit or miss	Contents of cache blocks after reference			
		Set 0	Set 0	Set 1	Set 1
0	miss	Memory[0]			
8	miss	Memory[0]	Memory[8]		
0	hit	Memory[0]	Memory[8]		
6	miss	Memory[0]	Memory[6]		
8	miss	Memory[8]	Memory[6]		

Four misses on five accesses

Memory Accesses in Cache

Fully Associative Cache:



Address of memory block accessed	Hit or miss	Contents of cache blocks after reference			
		Block 0	Block 1	Block 2	Block 3
0	miss	Memory[0]			
8	miss	Memory[0]	Memory[8]		
0	hit	Memory[0]	Memory[8]		
6	miss	Memory[0]	Memory[8]	Memory[6]	
8	hit	Memory[0]	Memory[8]	Memory[6]	

Three misses on five memory accesses

Memory Accesses in Cache

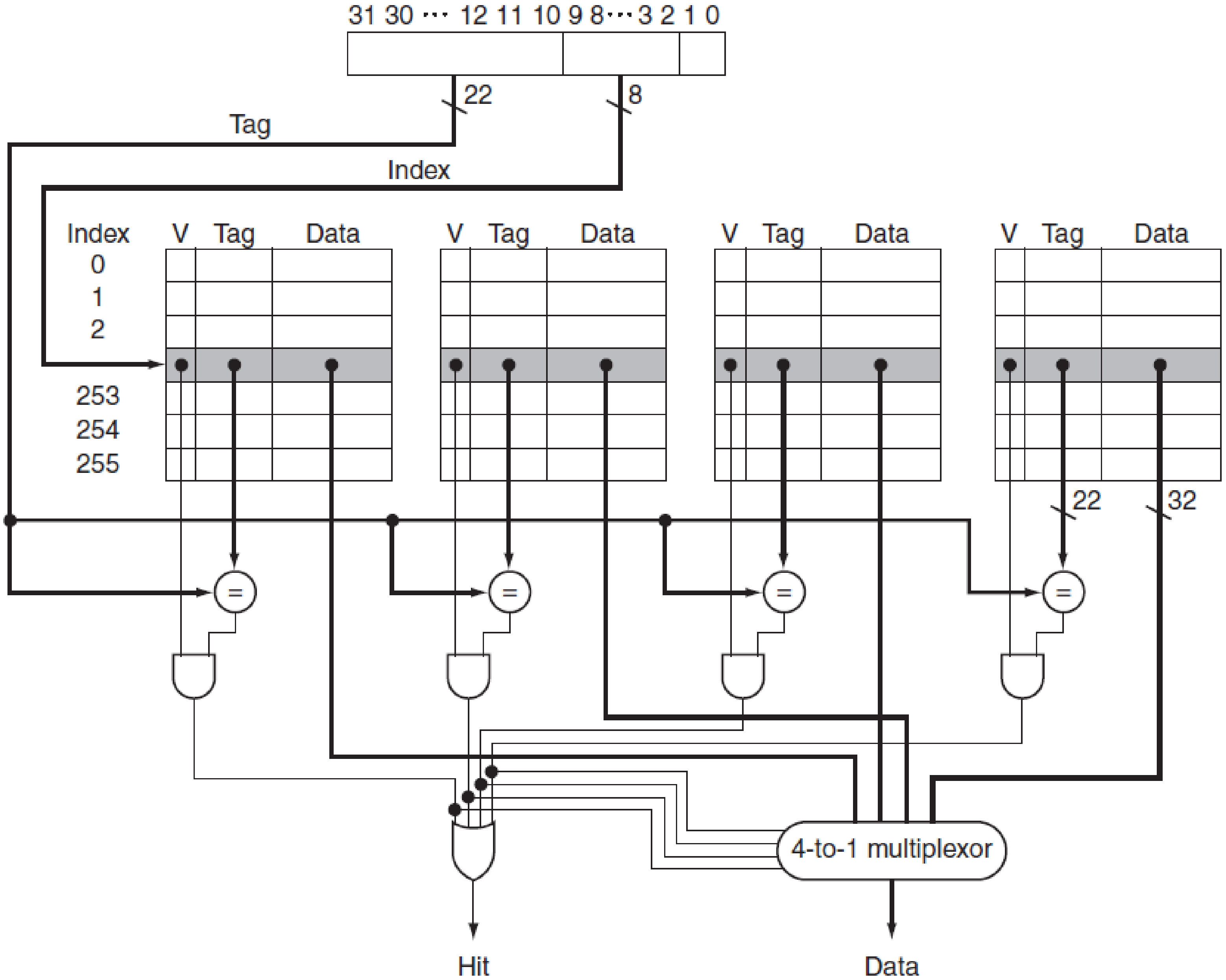
- Here, it shows that fully associative is best.

- If we had eight blocks in the cache, there would be no replacements in the two-way set-associative cache, and it would have the same number of misses as the fully associative cache.

- If we had 16 blocks, all three caches would have the same number of misses.

- So, the cache size and associativity are not independent in determining cache performance.

Implementation of Four-way Set Associative Cache



Size of Tag

- Cache size = 4K blocks
- Block size = 4 words
- Address = 32 bits

Direct mapped cache:

- Number of sets = 4K
- Index = 12 bits
- Number of tags=32-12-4= 16 bits

Size of Tag

Two way Set Associative:

- Number of sets = 4K/2 = 2K
- Index = 11 bits.
- Tag = 32-11-4 =17 bits

Four way Set Associative:

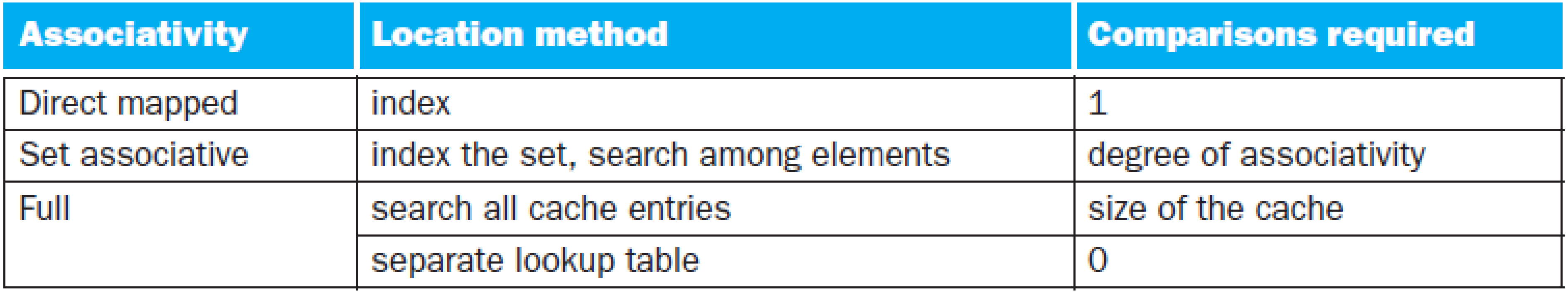
- Number of sets = 1K
- Index = 10 bits
- Tag = 32-10-4 =18 bits

Size of Tag

Fully Associative:

- Number of sets = 1
- Index = 0 bit
- Tag = 32-0-4 = 28 bits

Locating a Block



Associativity	Location method	Comparisons required
Direct mapped	index	1
Set associative	index the set, search among elements	degree of associativity
Full	search all cache entries	size of the cache
	separate lookup table	0

Block Placement

1. Random Selection
2. Least Recently Used (LRU)

Three Sources of Misses

1. **Compulsory Misses:**

   Caused by the first access to a block that has never been in the cache.

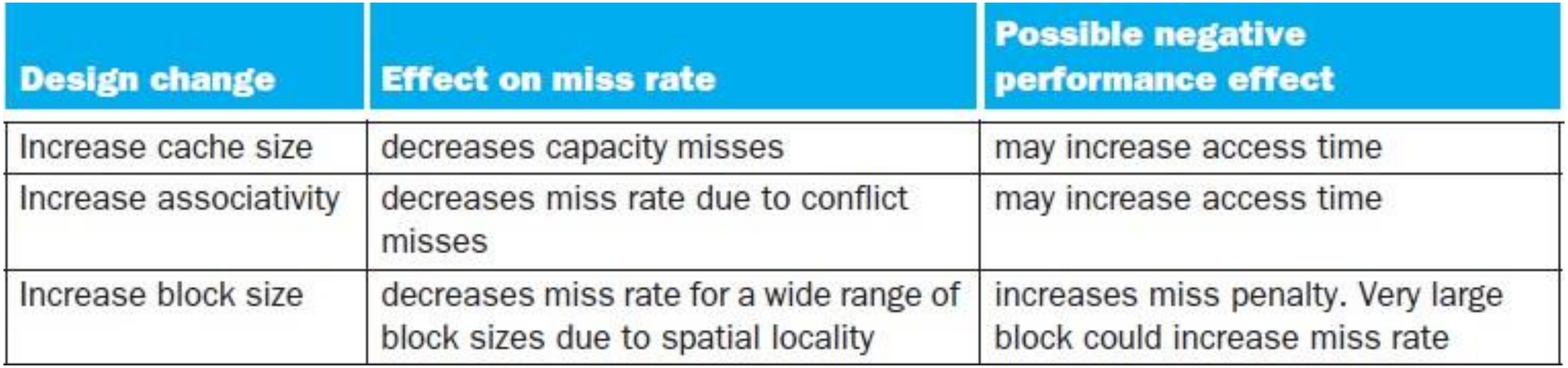
2. **Capacity Misses:**

   Caused when the cache cannot contain all the blocks needed during execution of a program.

3. **Conflict Misses/ Collision Misses:**

   When multiple blocks compete for the same location (direct mapping) or same set (set associative).

Solutions



Design change	Effect on miss rate	Possible negative performance effect
Increase cache size	decreases capacity misses	may increase access time
Increase associativity	decreases miss rate due to conflict misses	may increase access time
Increase block size	decreases miss rate for a wide range of block sizes due to spatial locality	increases miss penalty. Very large block could increase miss rate