CMPE 200
Computer Architecture & Design

Lecture 4.
Memory Hierarchy (3)

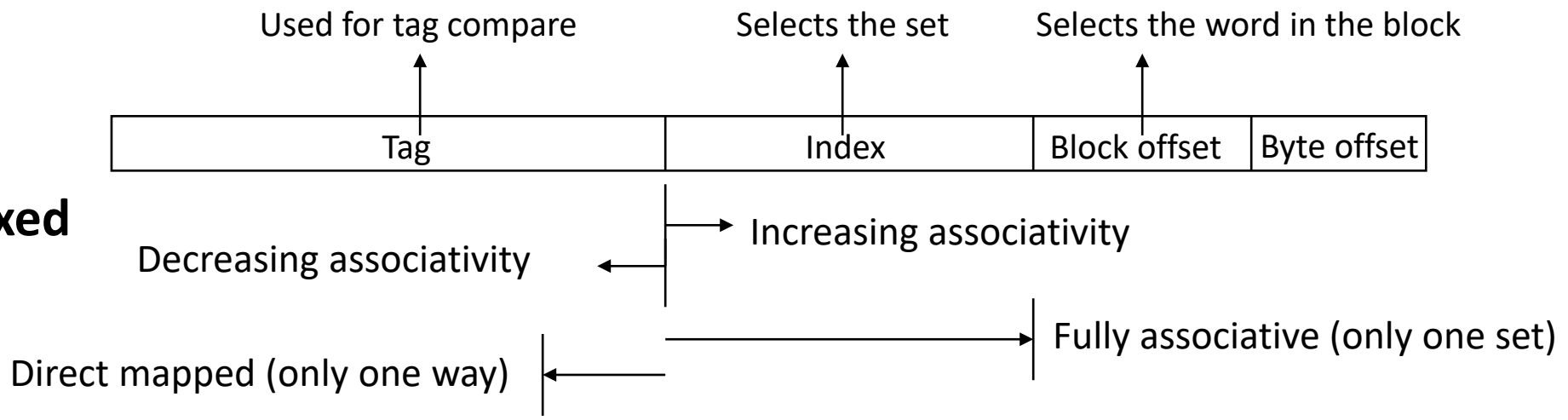
Haonan Wang

SAN JOSÉ STATE
UNIVERSITY

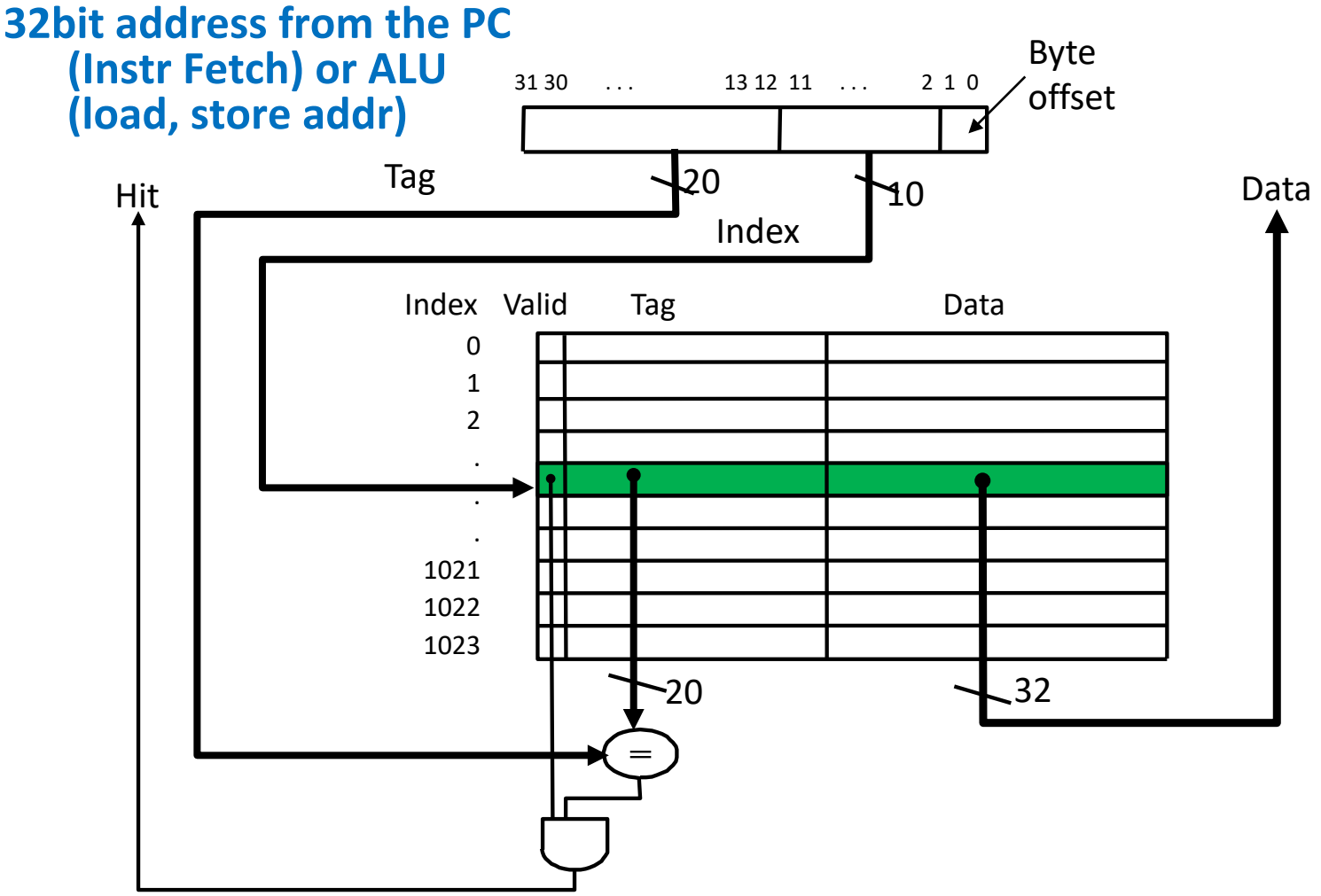
Cache Types

- **N-way Set-Associative:** Number of ways > 1 & Number of sets > 1
  - Slightly complex searching mechanism
- **Direct Mapped:** Number of ways = 1
  - Fast indexing mechanism
- **Fully-Associative:** Number of sets = 1
  - Extensive hardware resources required to search

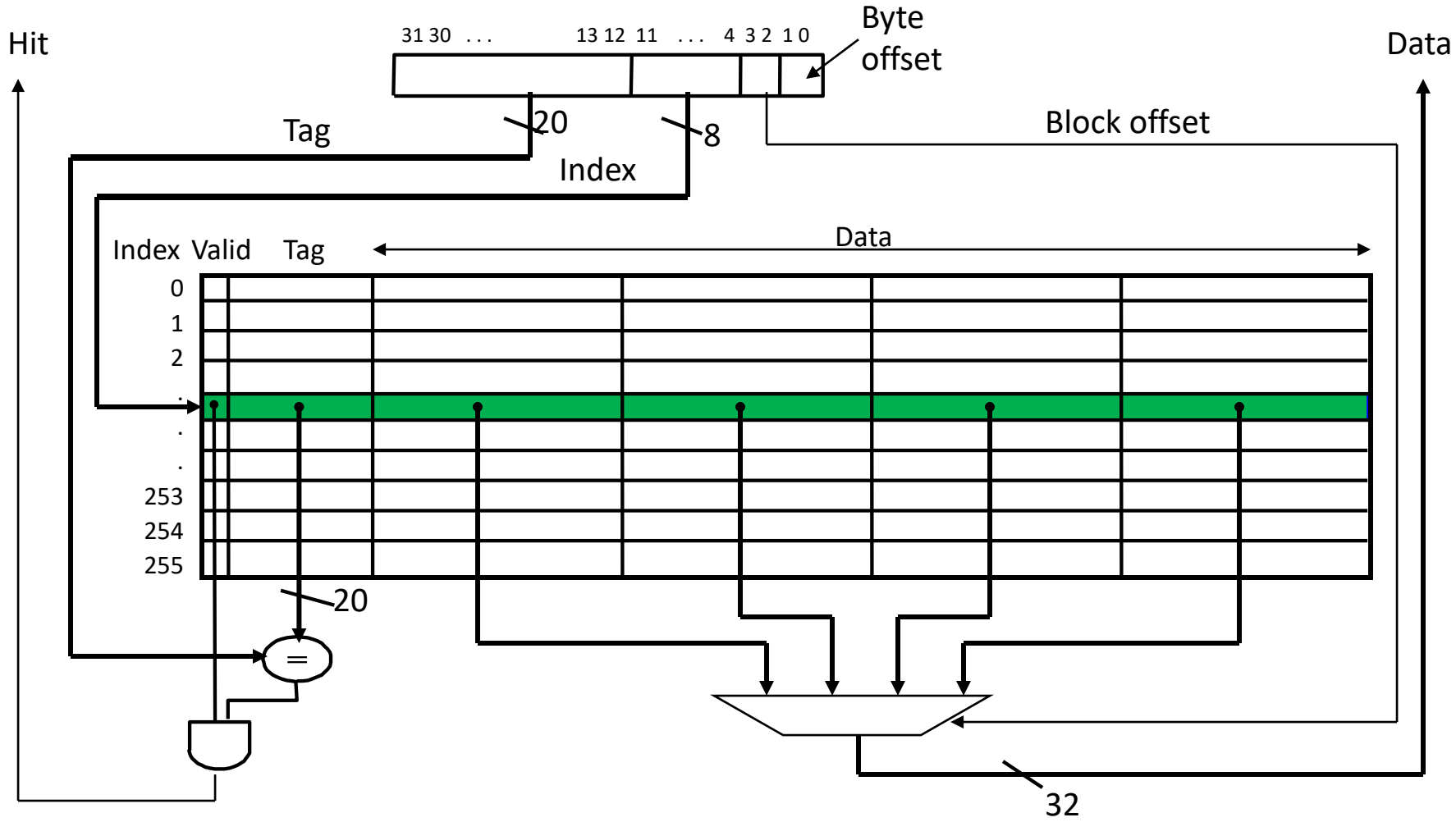
	Way 0	Way 1	…
Set 0	block 0	block 2	
Set 1	block 1	block 3	
⋮			



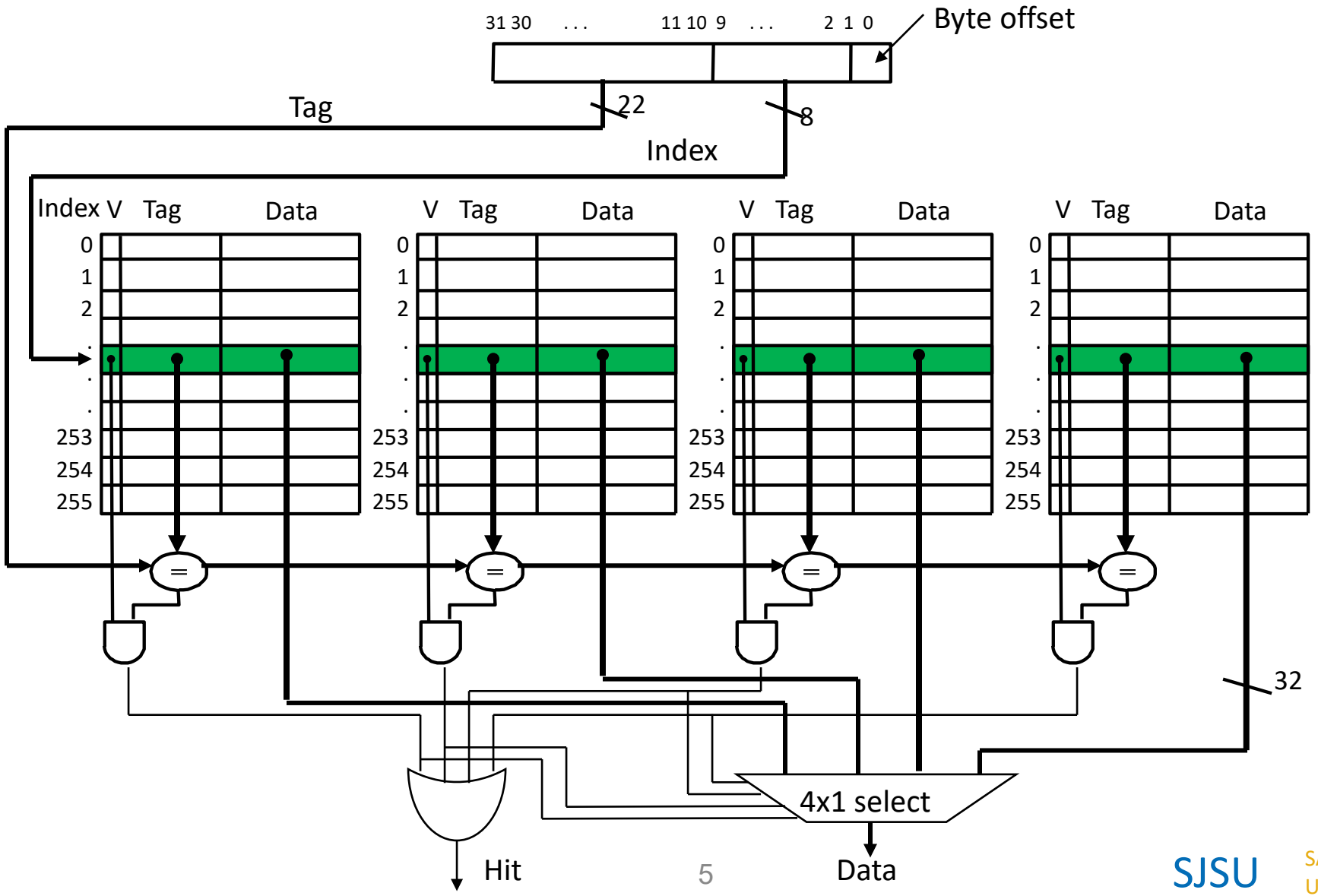
MIPS Direct Mapped Cache Example



Multiword Block Direct Mapped Cache



Four-Way Set Associative Cache



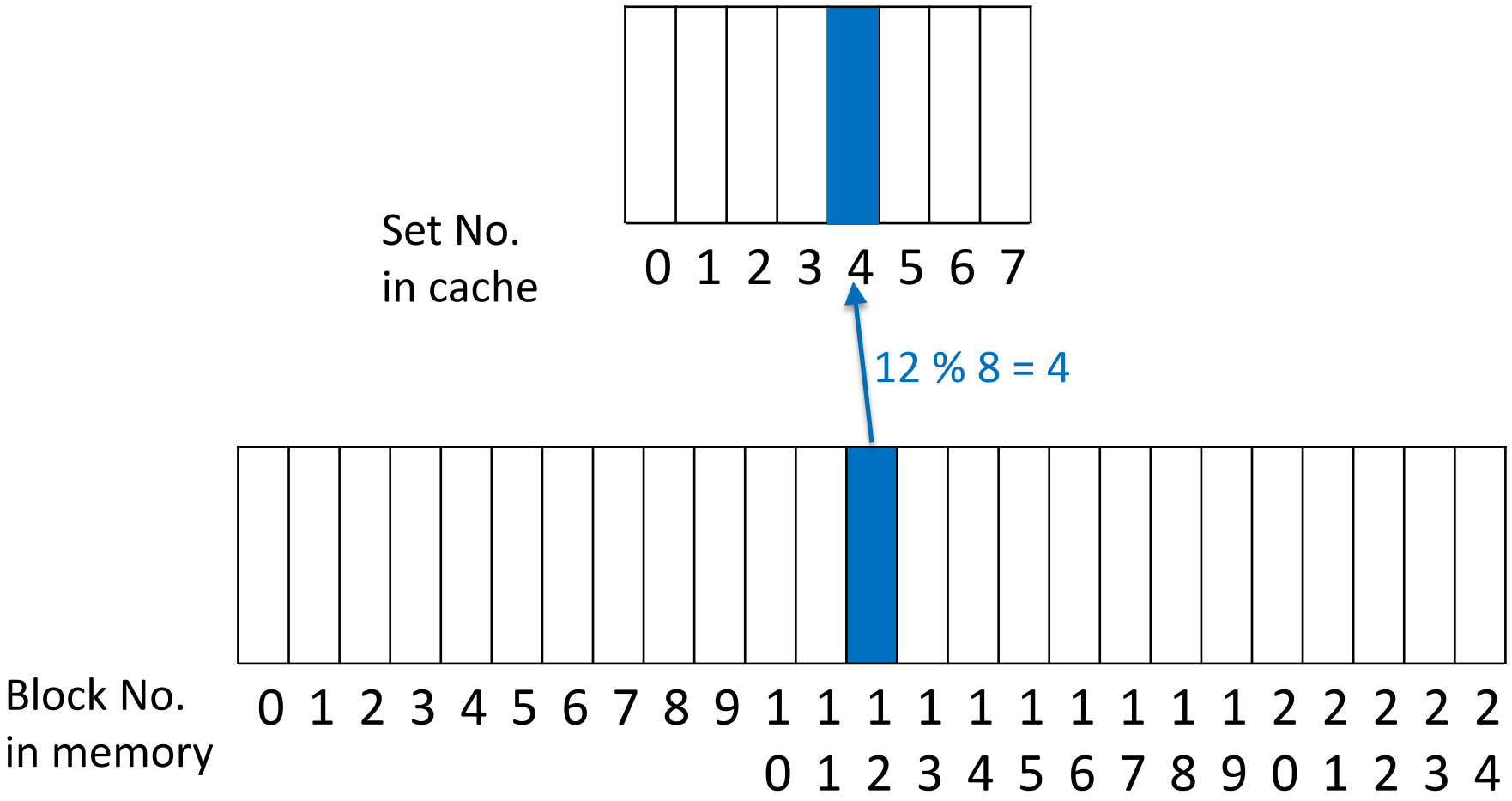
Costs of Set Associative Caches

- **Must have hardware for replacement policy**
  - E.g., to keep track of when each way's block was used

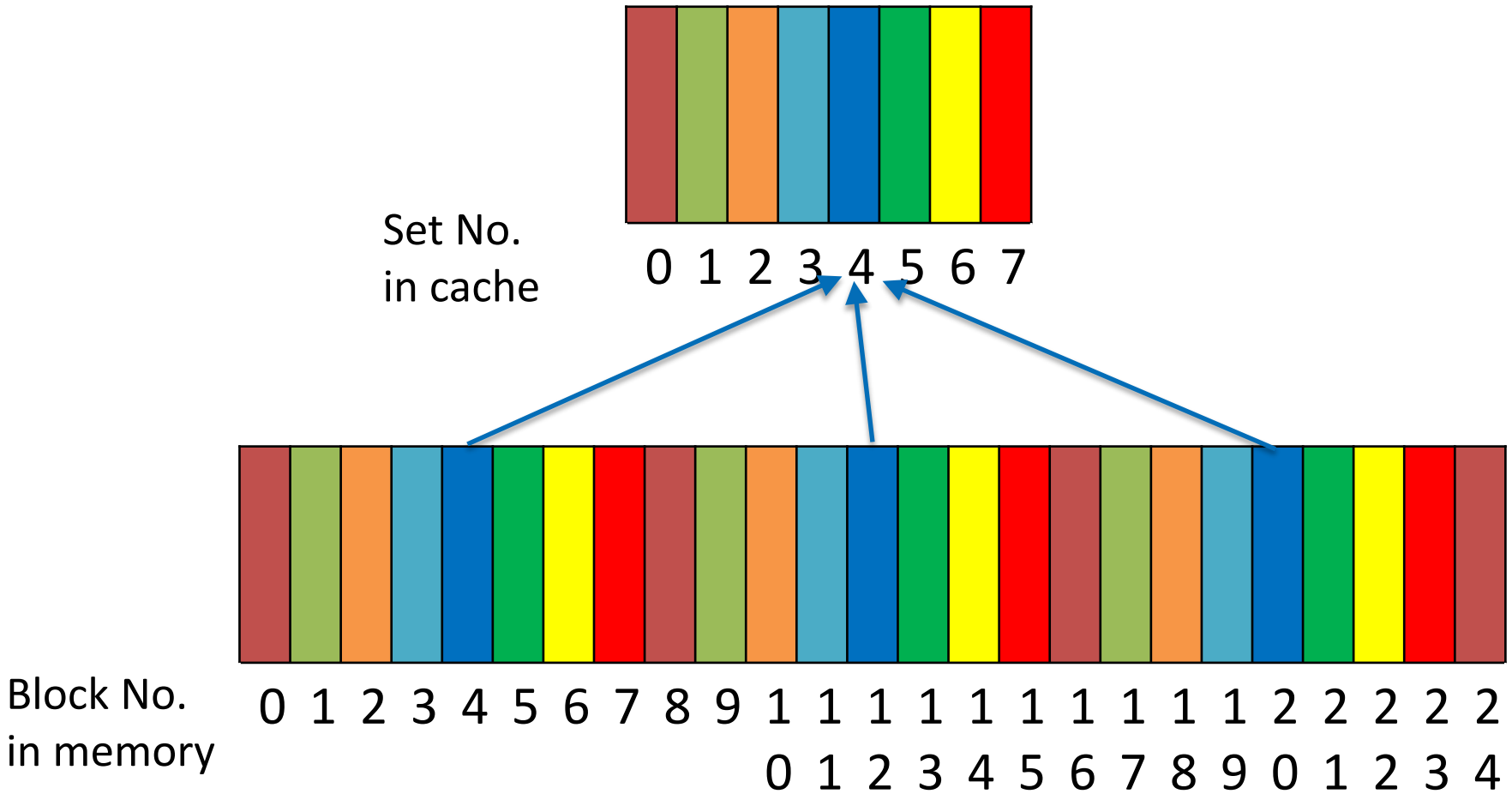
- **N-way set associative cache costs**
  - N comparators (delay and area) & MUX delay
    - Data is available after Hit/Miss decision.
    - In a direct mapped cache, the cache block is available before the Hit/Miss decision.

- **Total cache line size = valid field size + tag size + block data size + data for cache policy (e.g., time stamp, modified bit, etc.)**

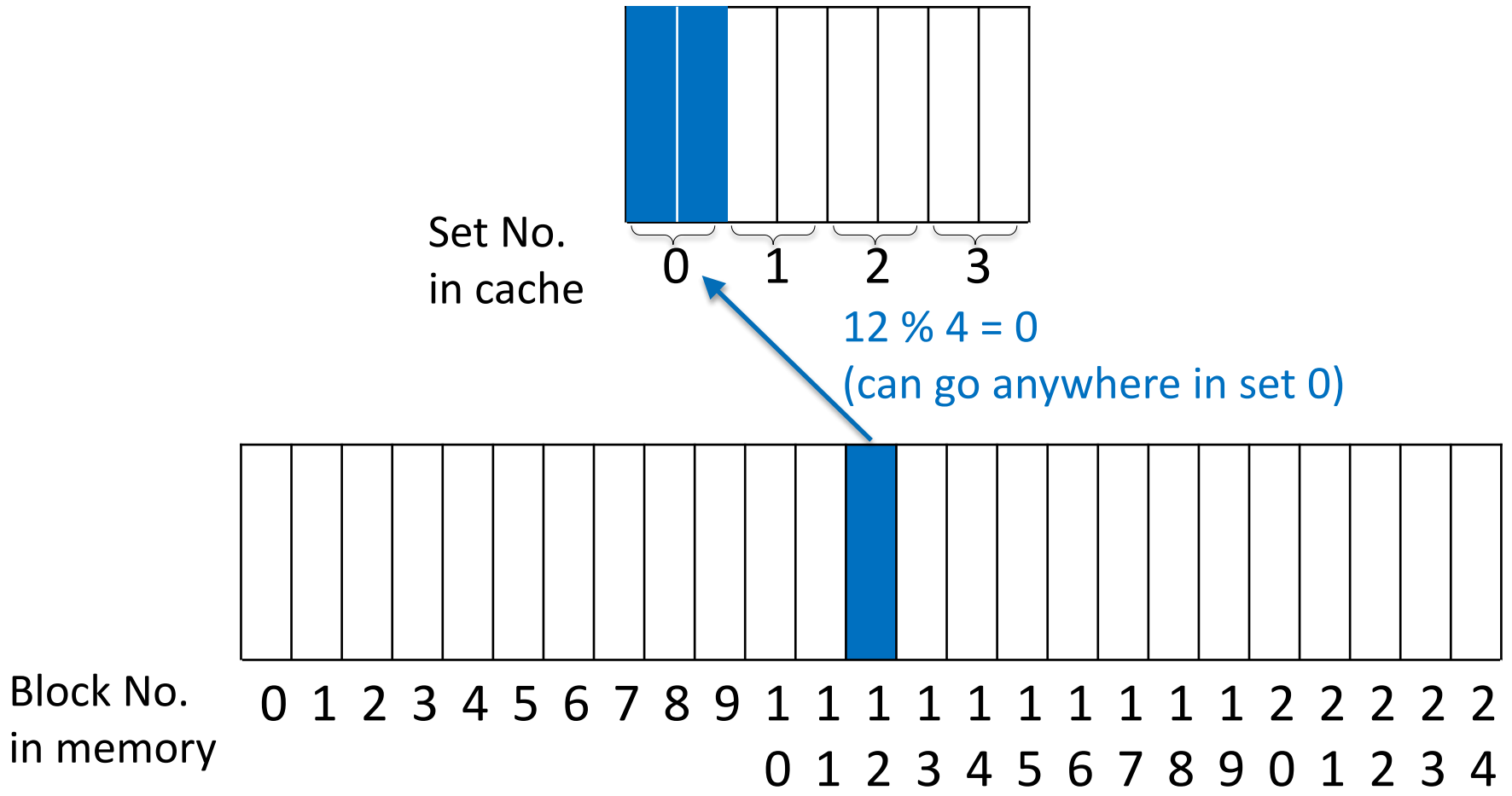
Mapping Example: Direct Mapped



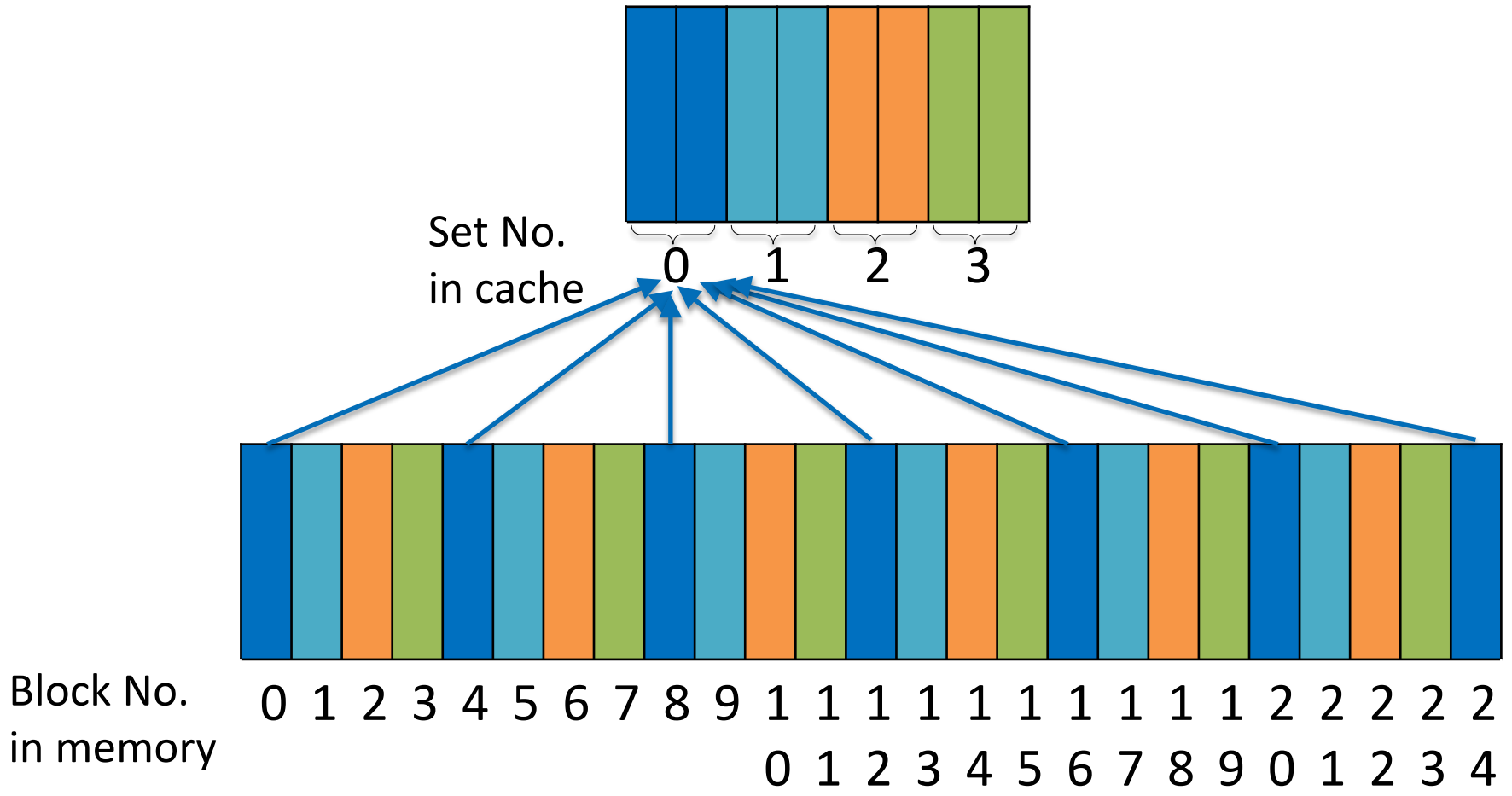
Mapping Example: Direct Mapped



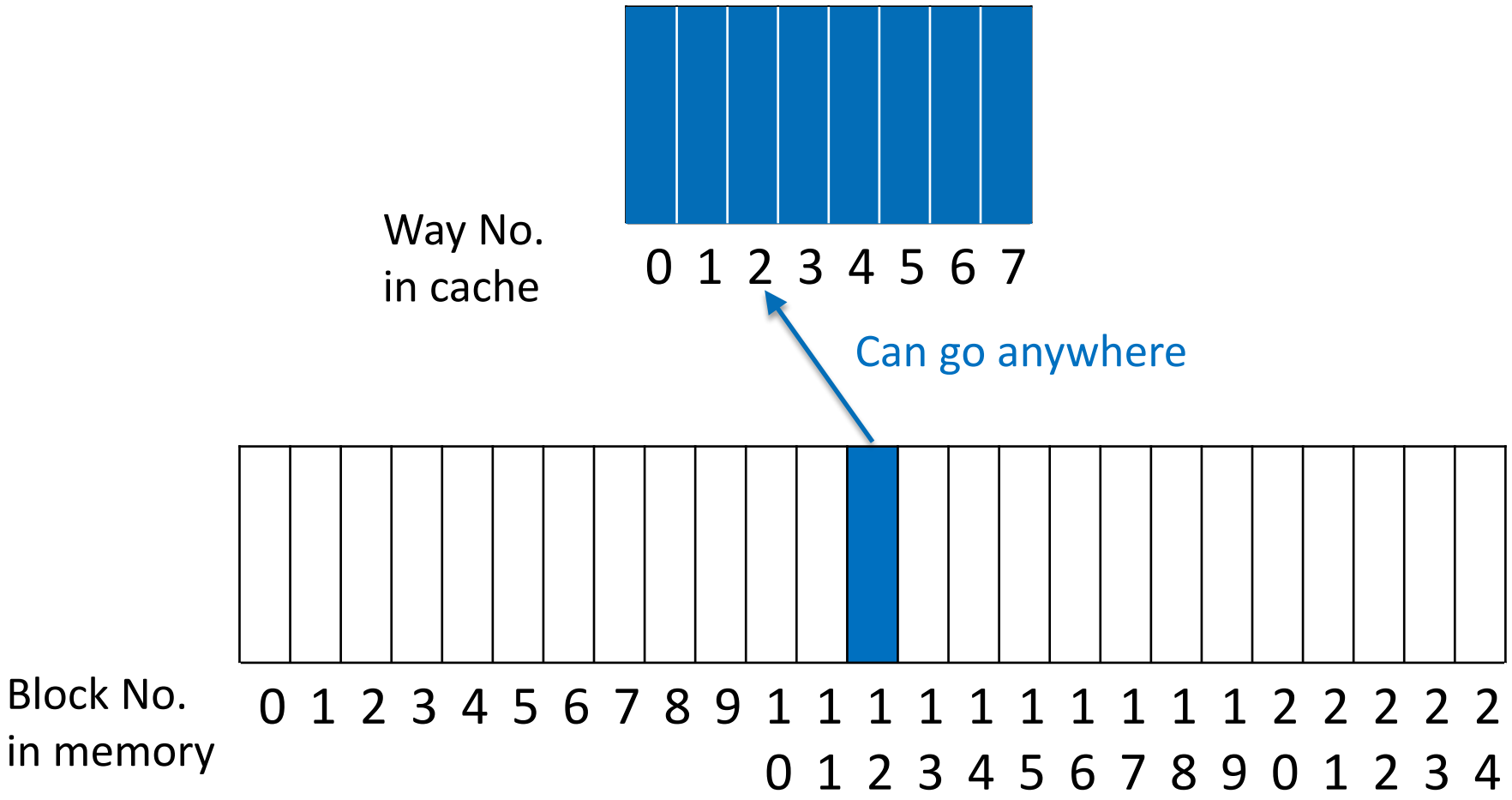
Mapping Example: 2-way



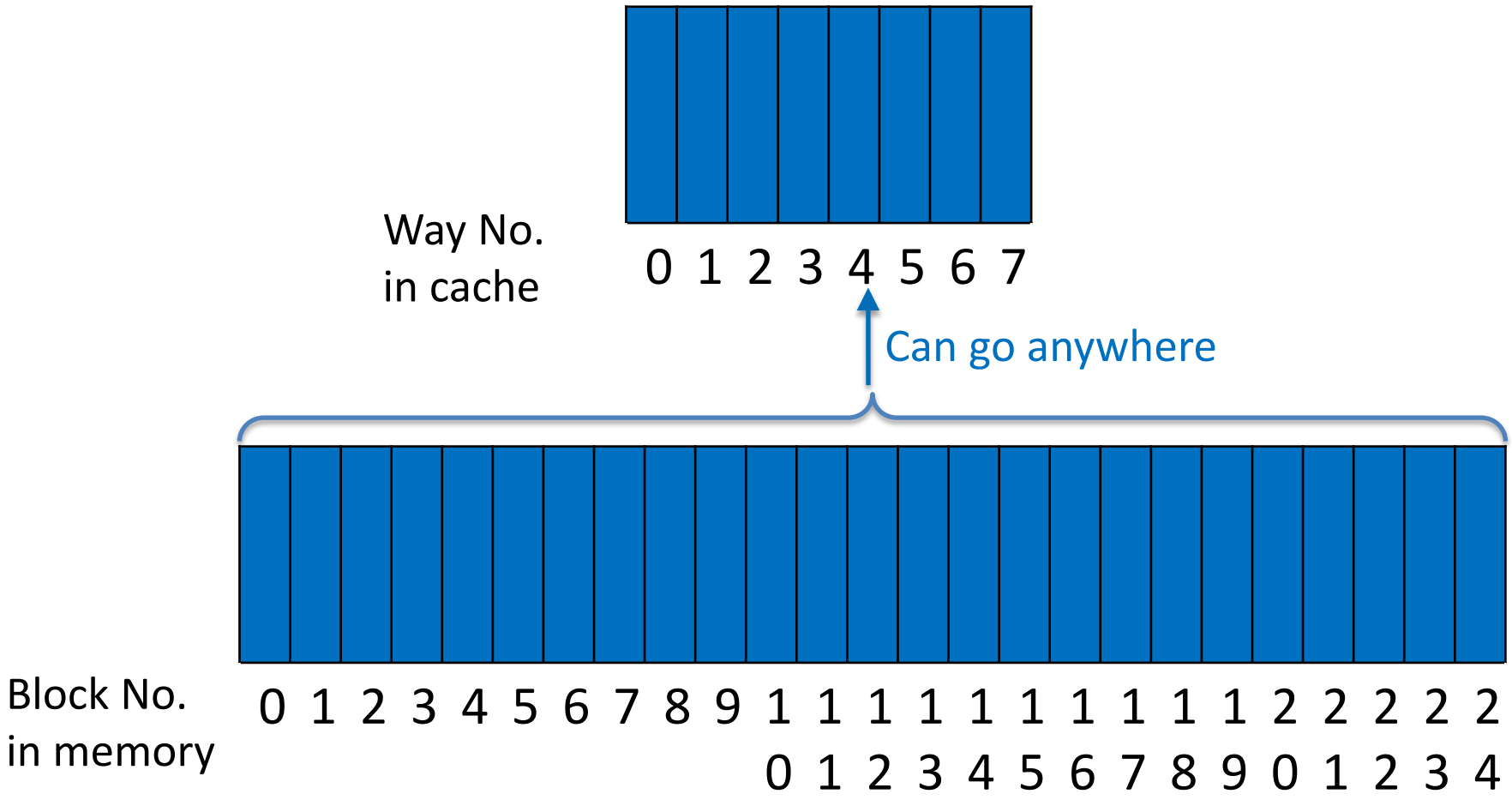
Mapping Example: 2-way



Mapping Example: Fully-associative

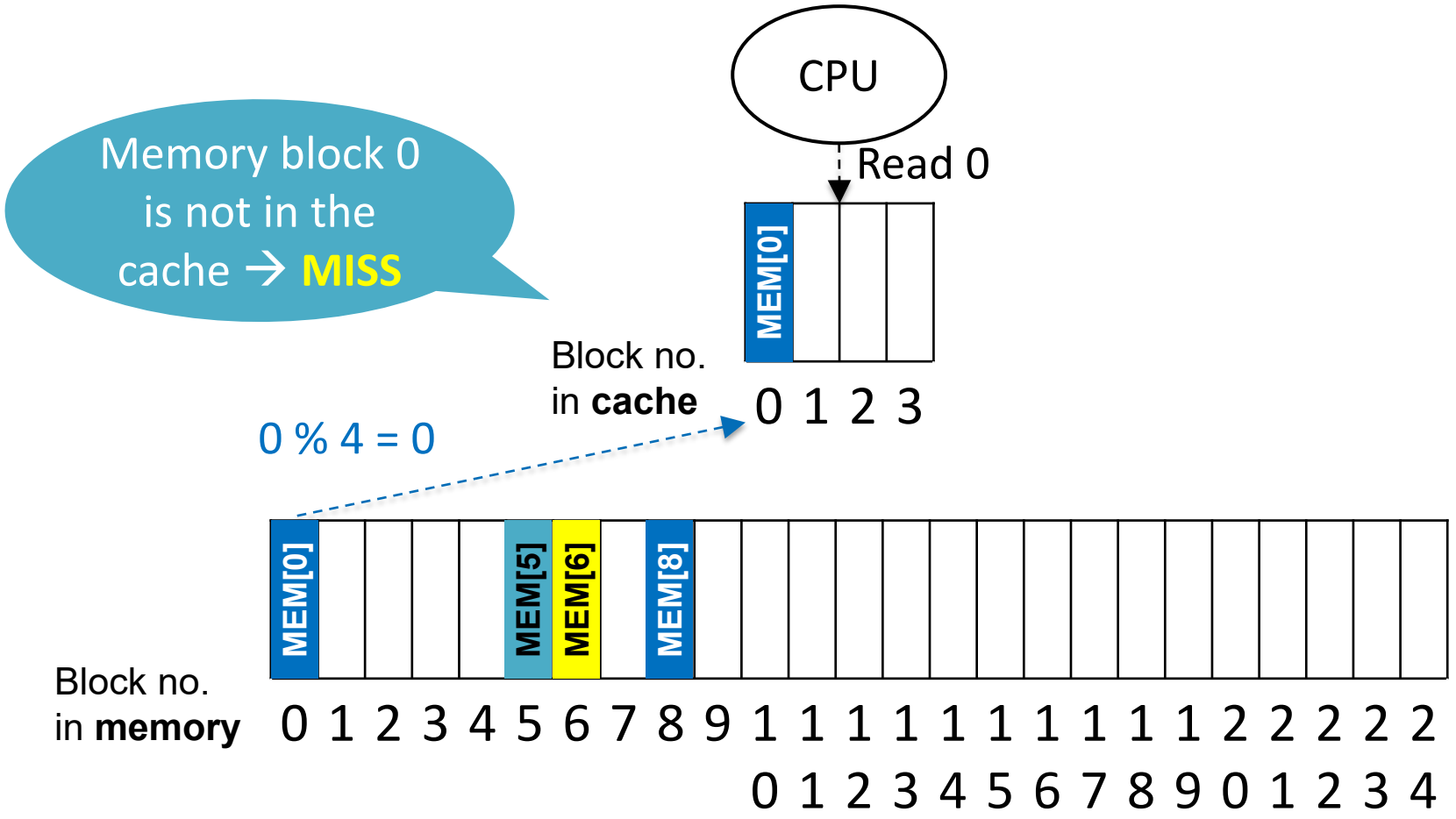


Mapping Example: Fully-associative



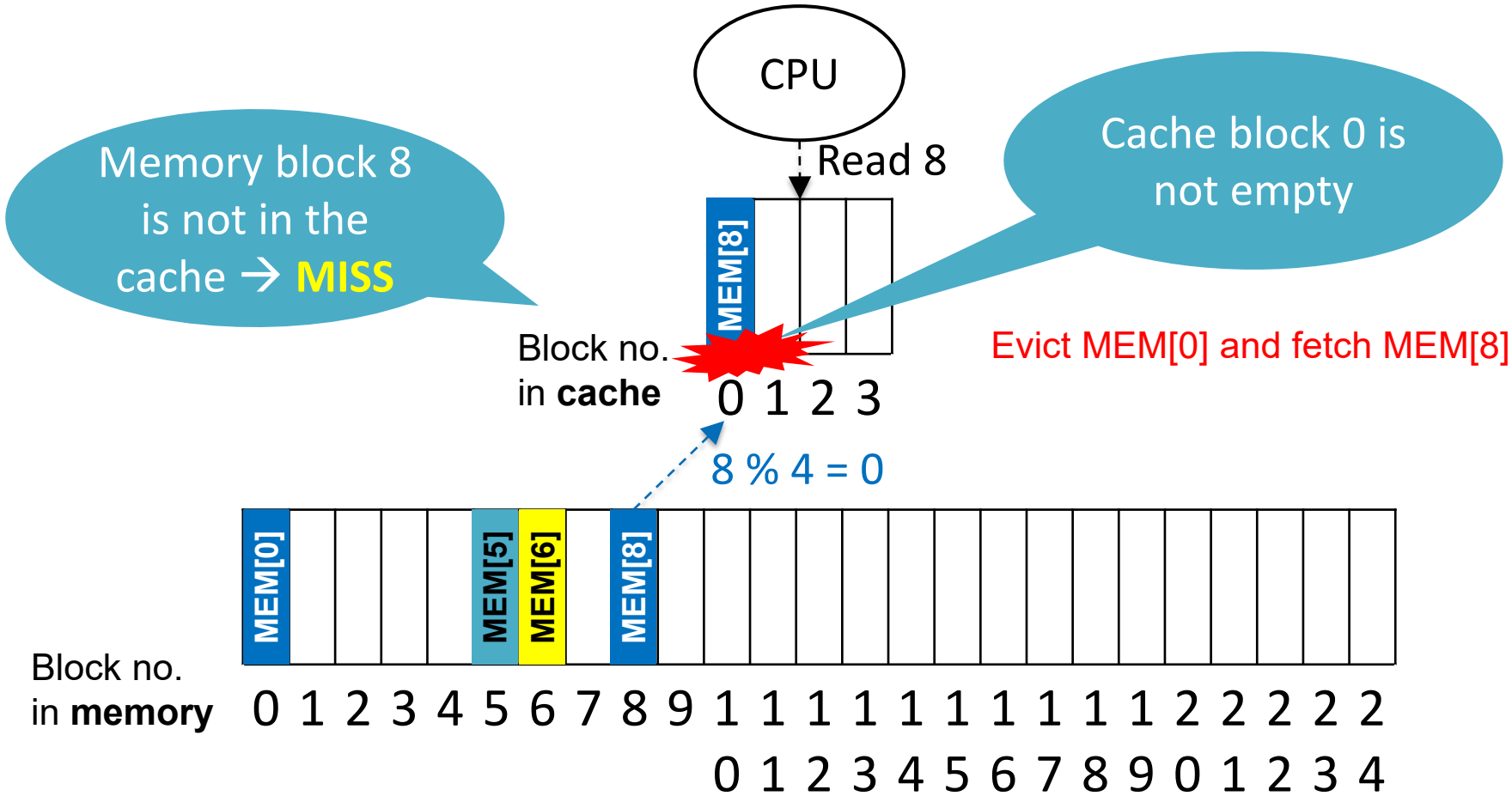
Miss Example: Direct Mapped

- **Memory block access sequence: 0, 8, 0, 6, 5**



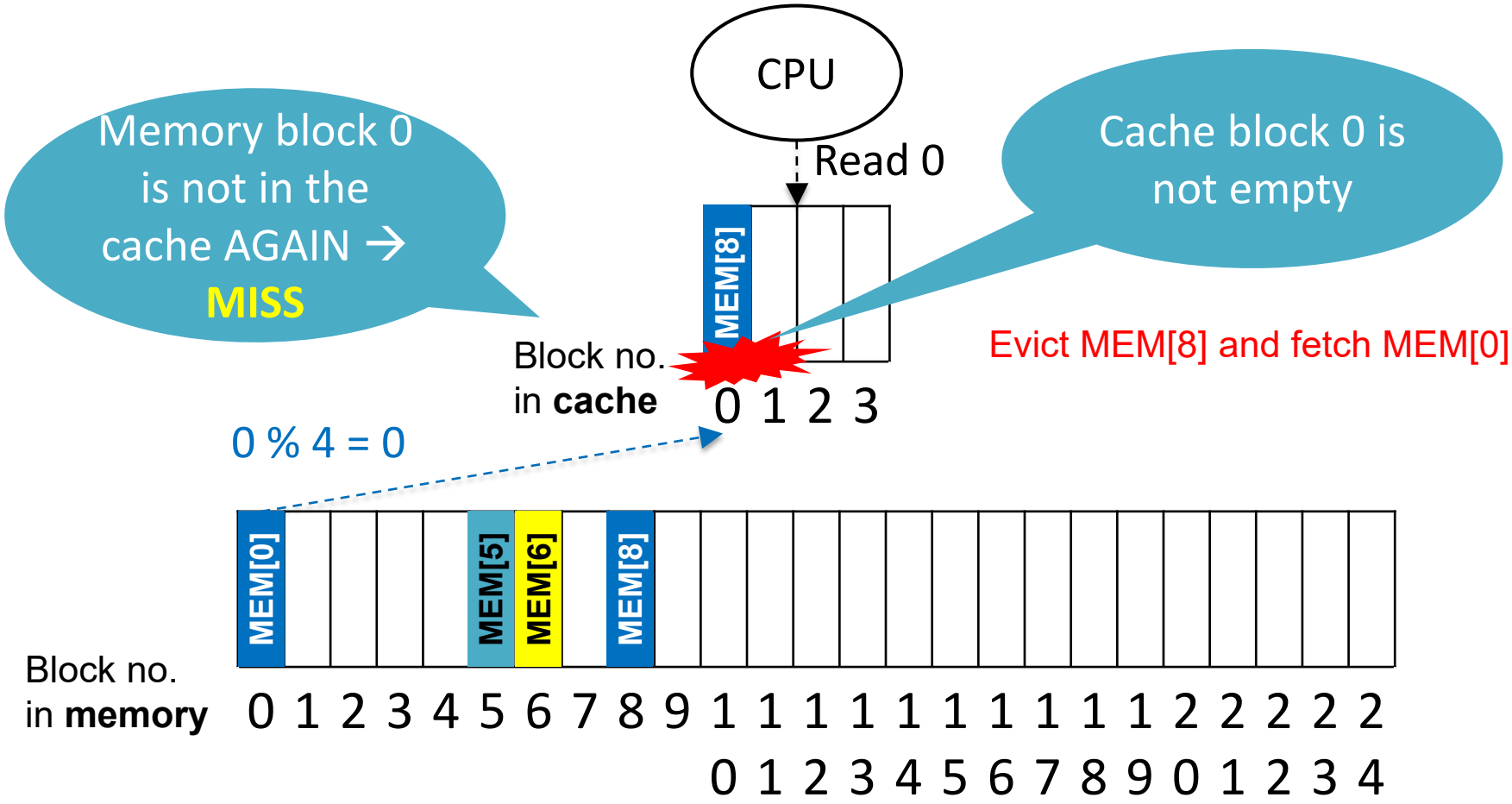
Miss Example: Direct Mapped

- **Memory block access sequence: 0, 8, 0, 6, 5**



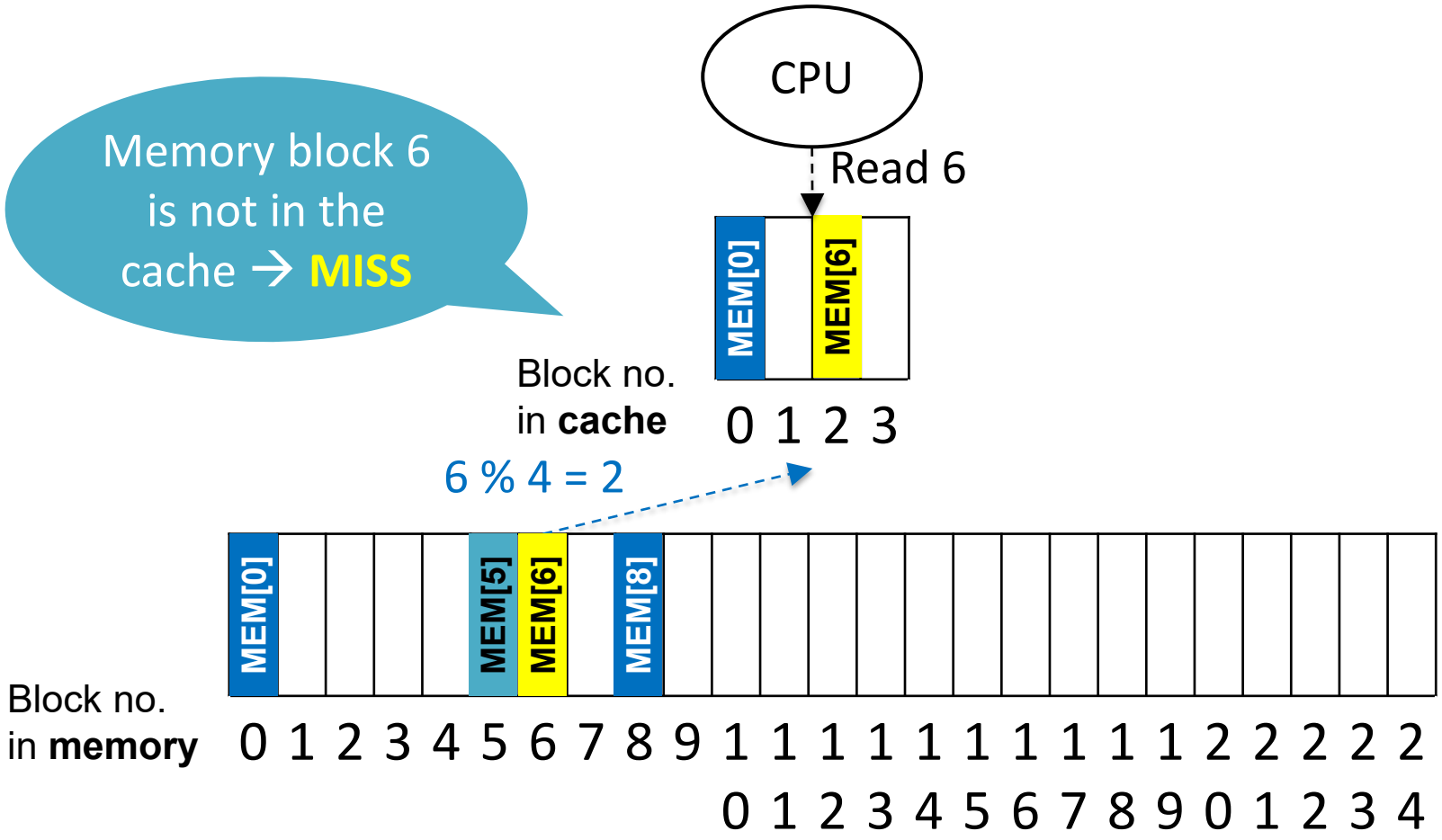
Miss Example: Direct Mapped

- **Memory block access sequence: 0, 8, 0, 6, 5**



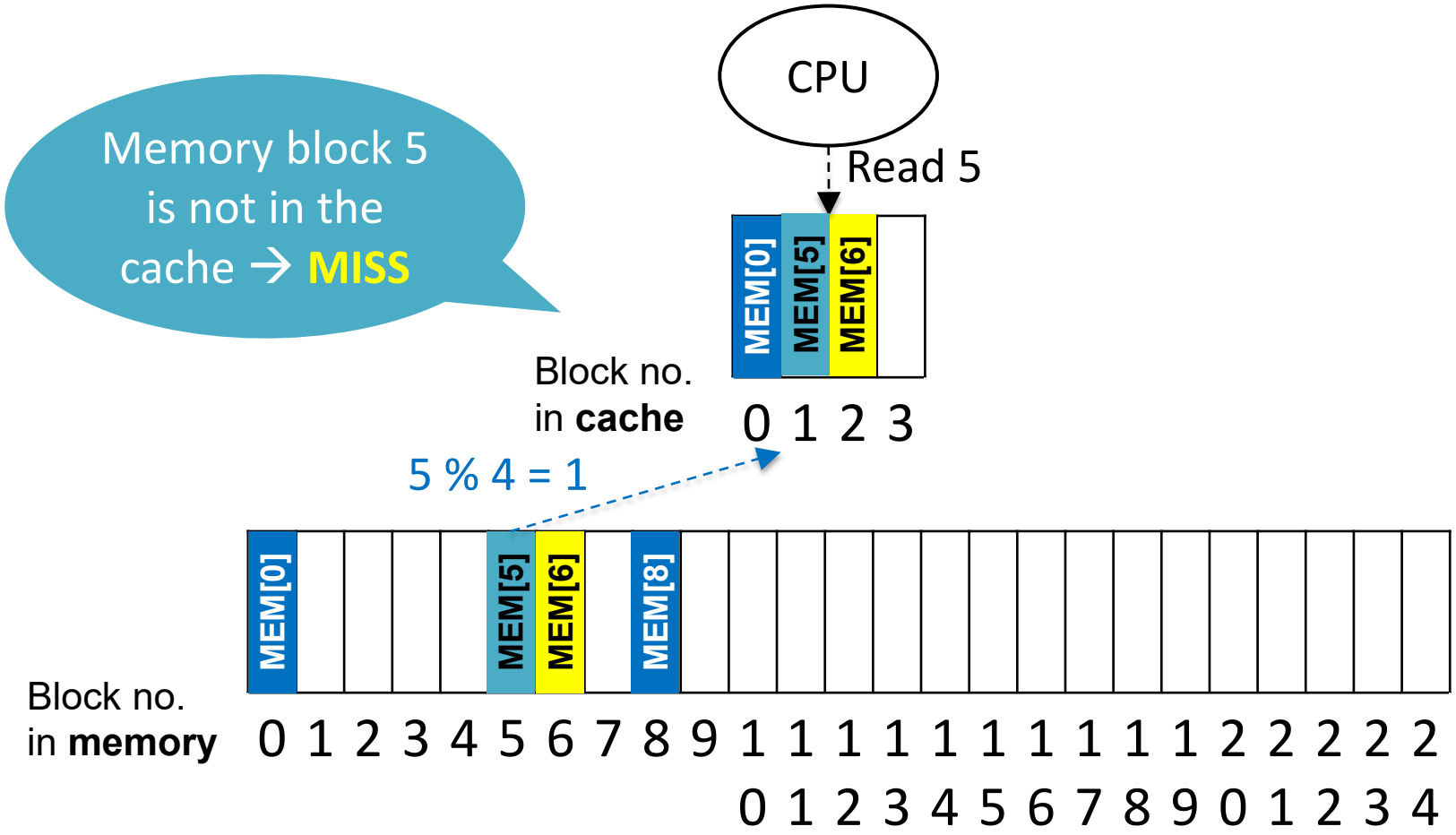
Miss Example: Direct Mapped

- **Memory block access sequence: 0, 8, 0, 6, 5**



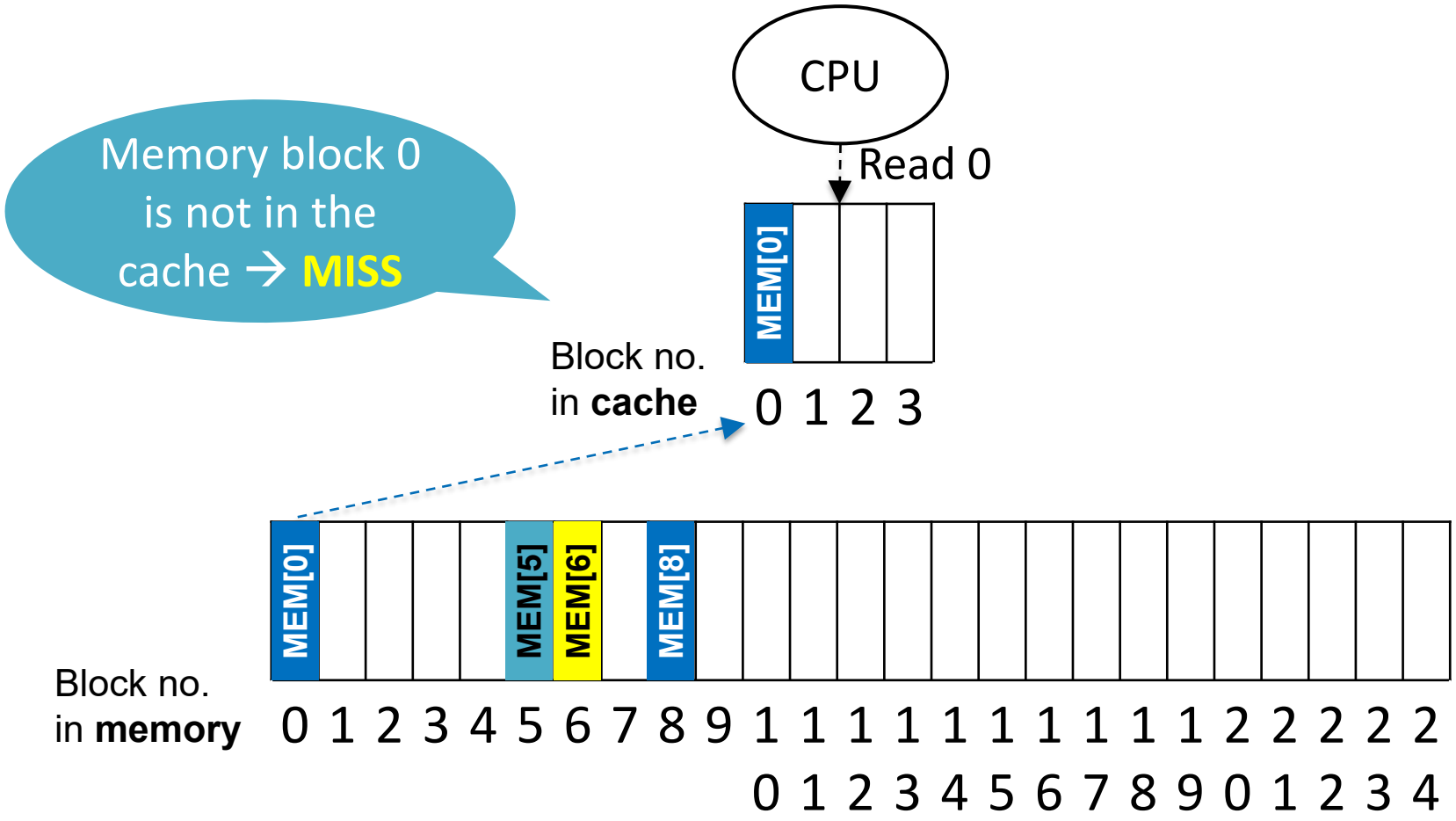
Miss Example: Direct Mapped

- **Memory block access sequence: 0, 8, 0, 6, 5**



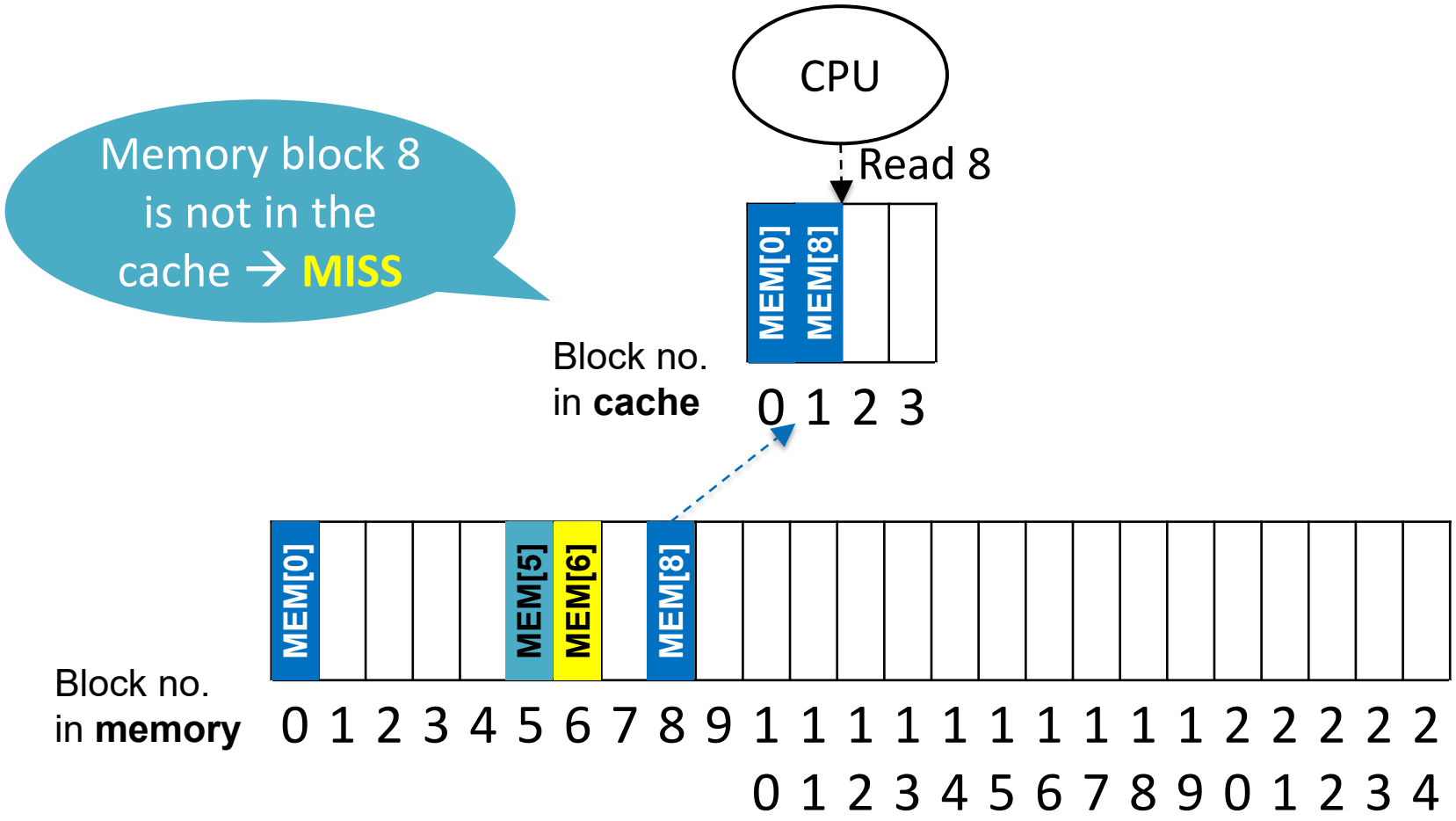
Miss Example: Fully-associative

- **Memory block access sequence: 0, 8, 0, 6, 5**



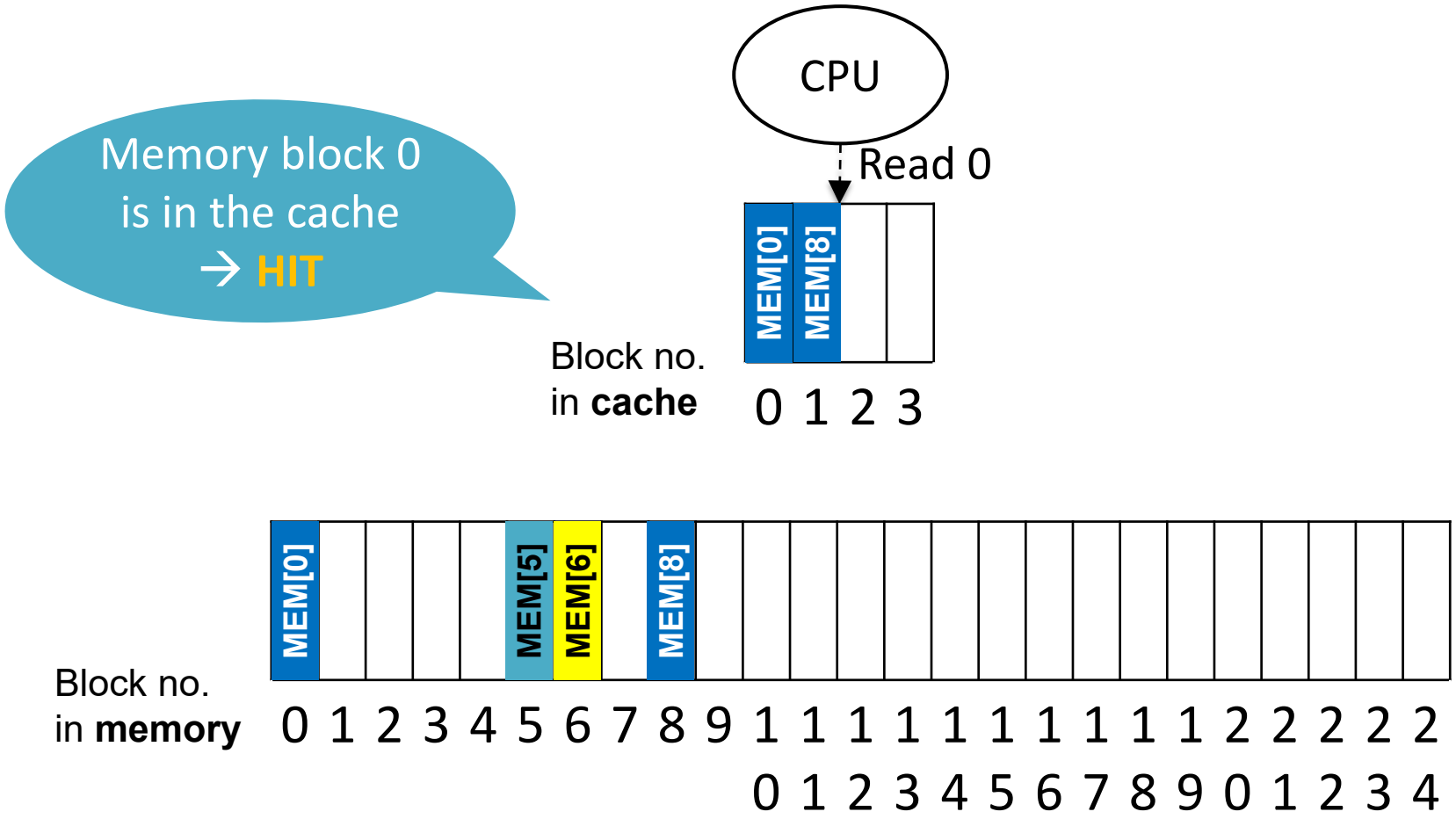
Miss Example: Fully-associative

- **Memory block access sequence: 0, 8, 0, 6, 5**



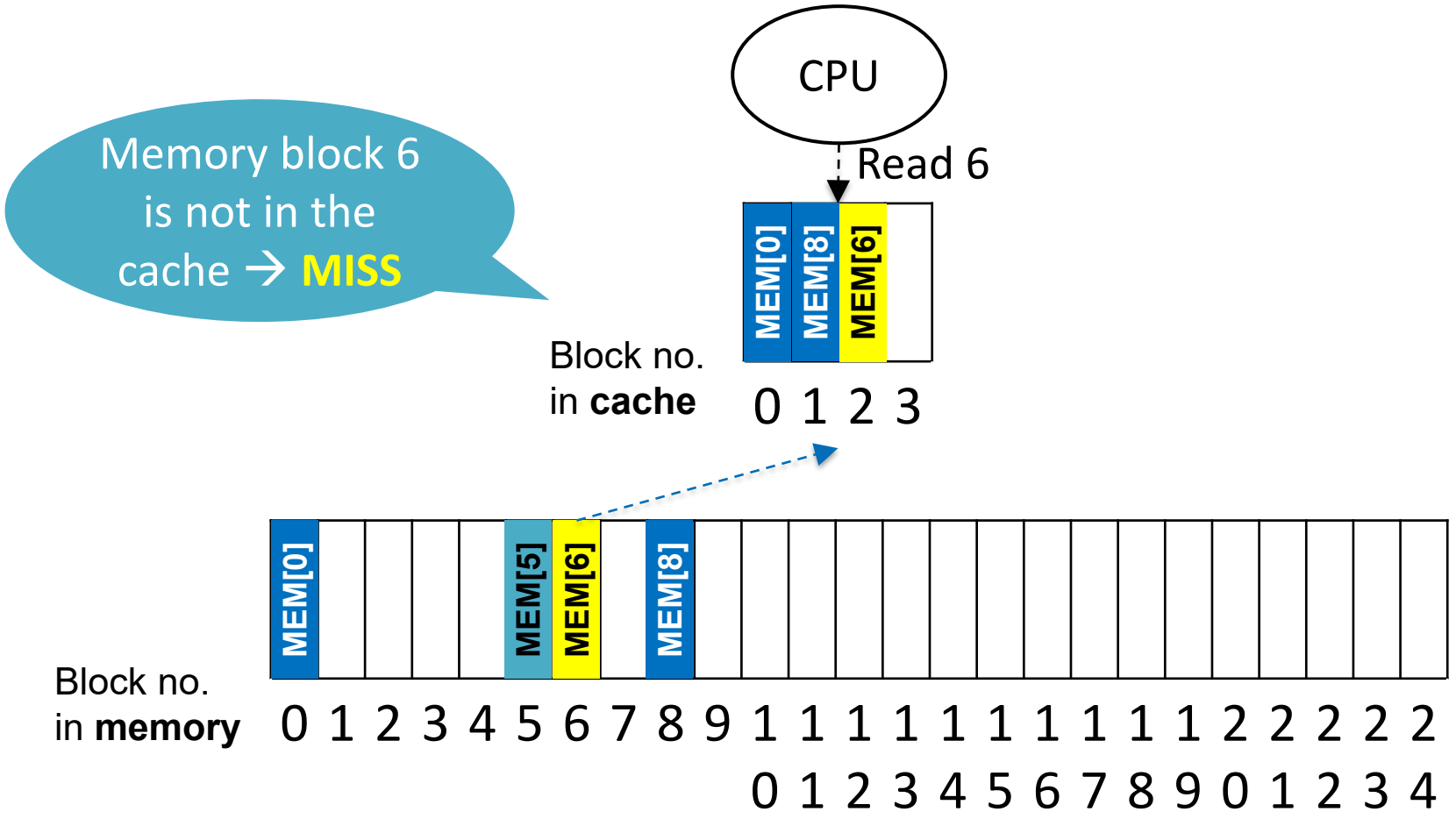
Miss Example: Fully-associative

- **Memory block access sequence: 0, 8, 0, 6, 5**



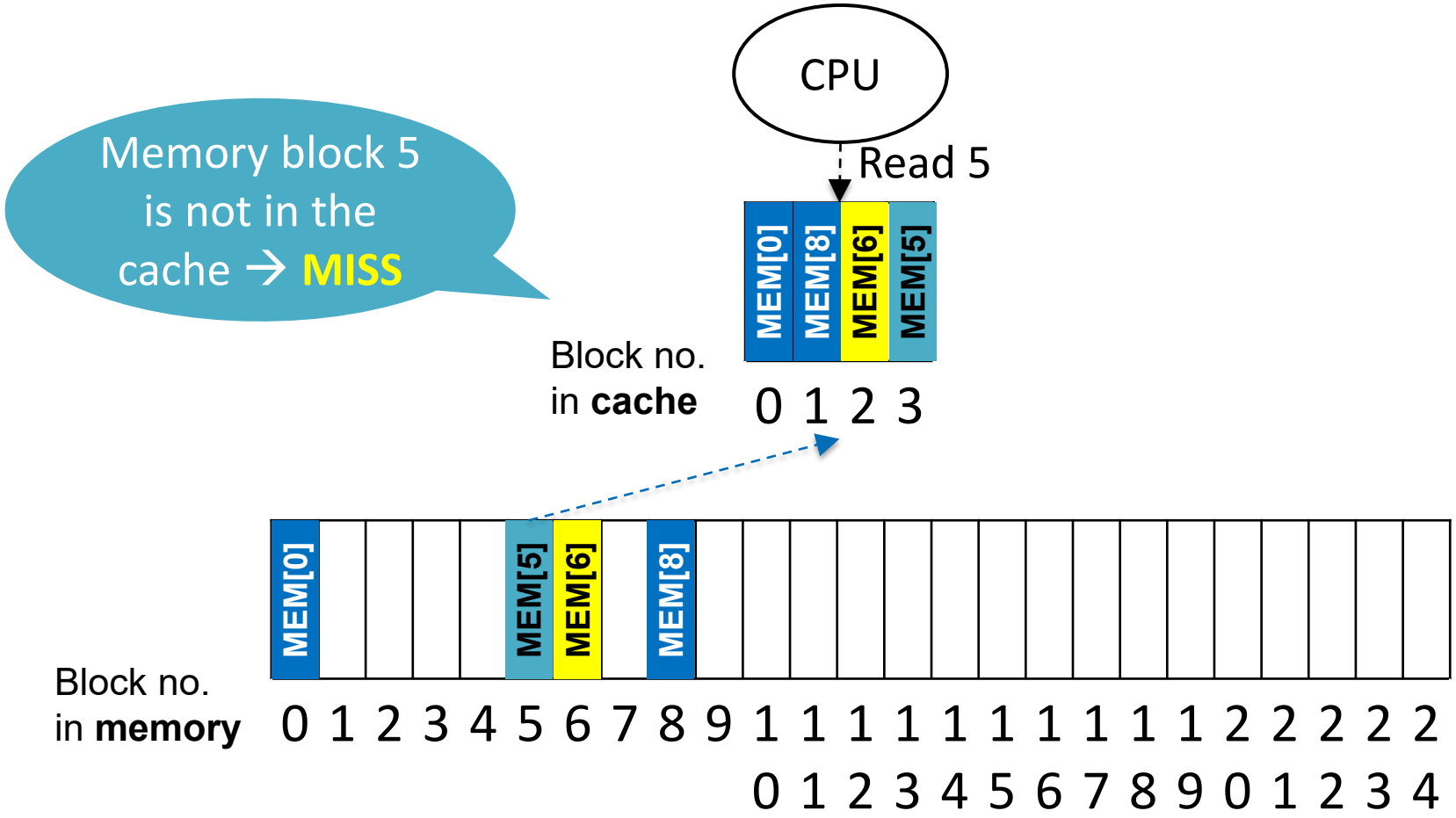
Miss Example: Fully-associative

- **Memory block access sequence: 0, 8, 0, 6, 5**



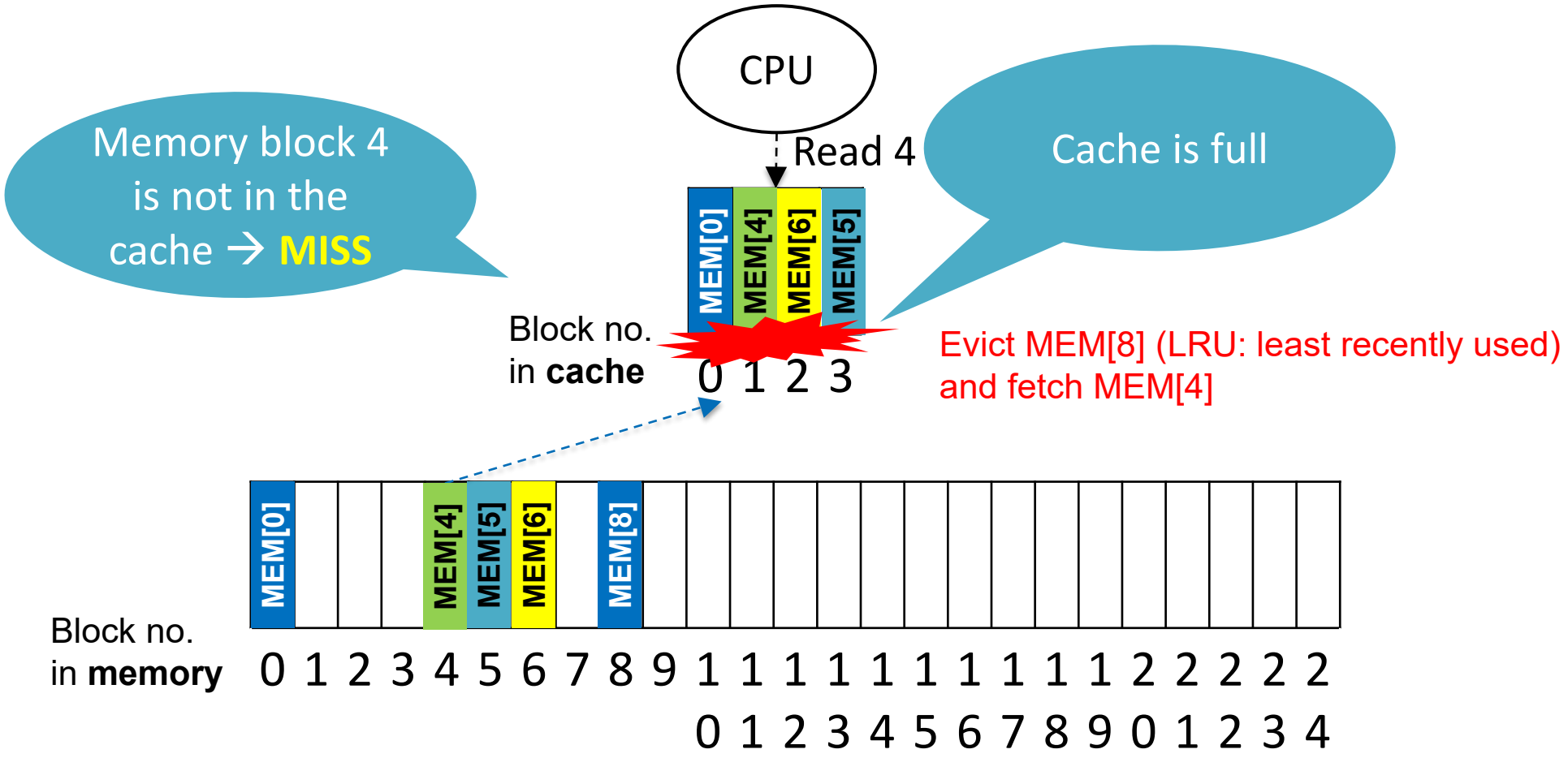
Miss Example: Fully-associative

- **Memory block access sequence: 0, 8, 0, 6, 5**



Miss Example: Fully-associative

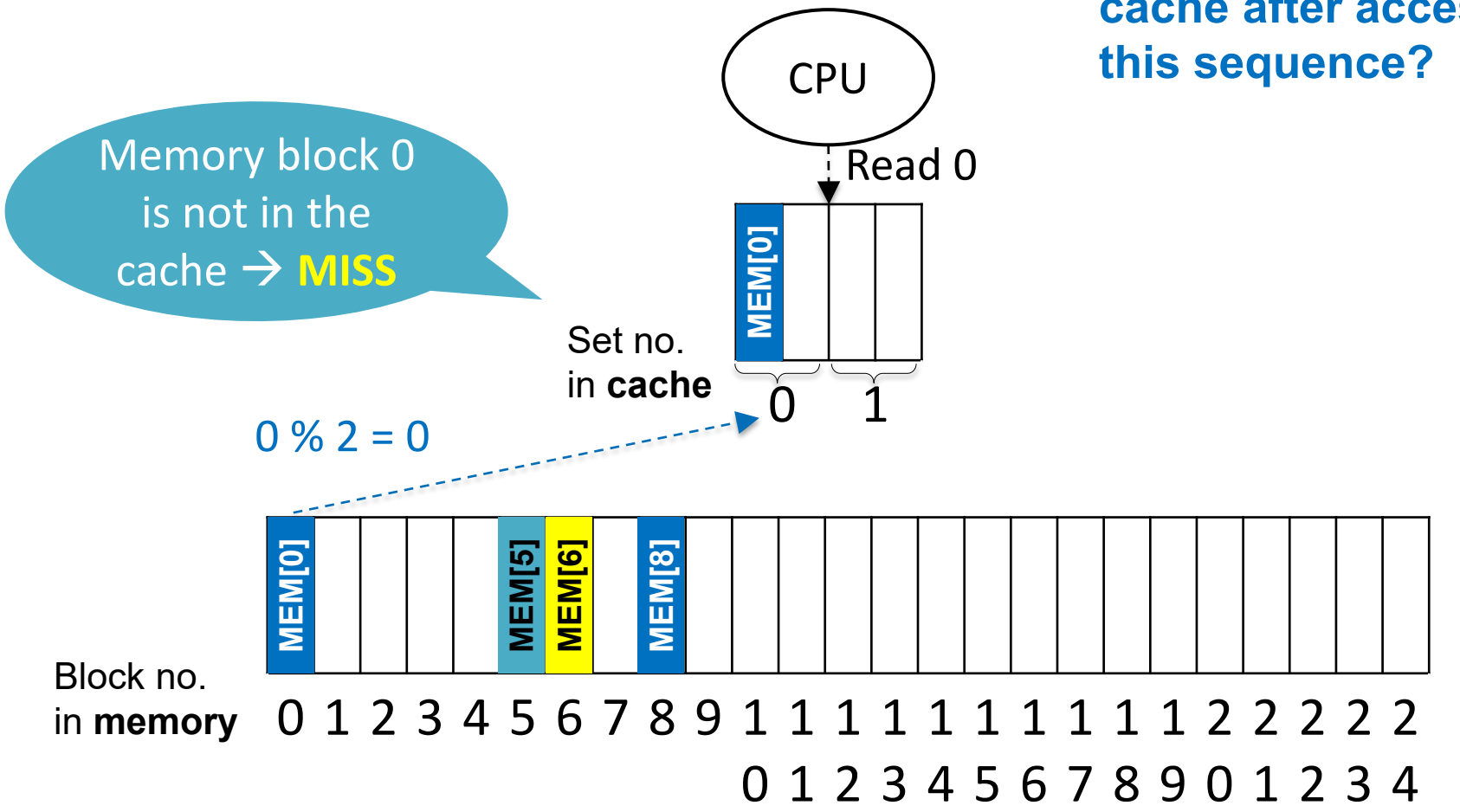
- **Memory block access sequence: 0, 8, 0, 6, 5, 4**



Miss Example: 2-way

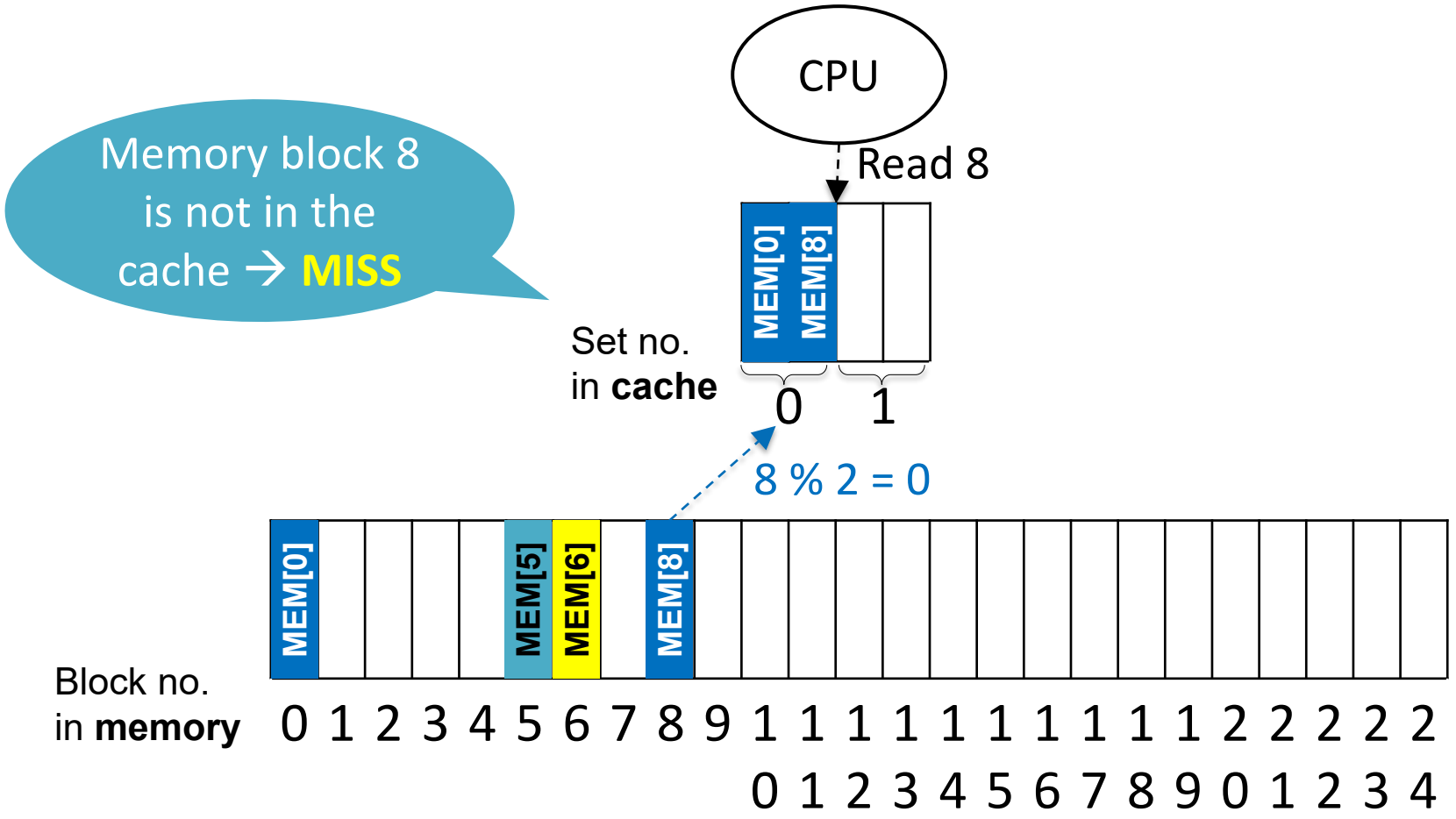
- **Memory block access sequence: 0, 8, 0, 6, 5**

Which data will be in the cache after accessing this sequence?



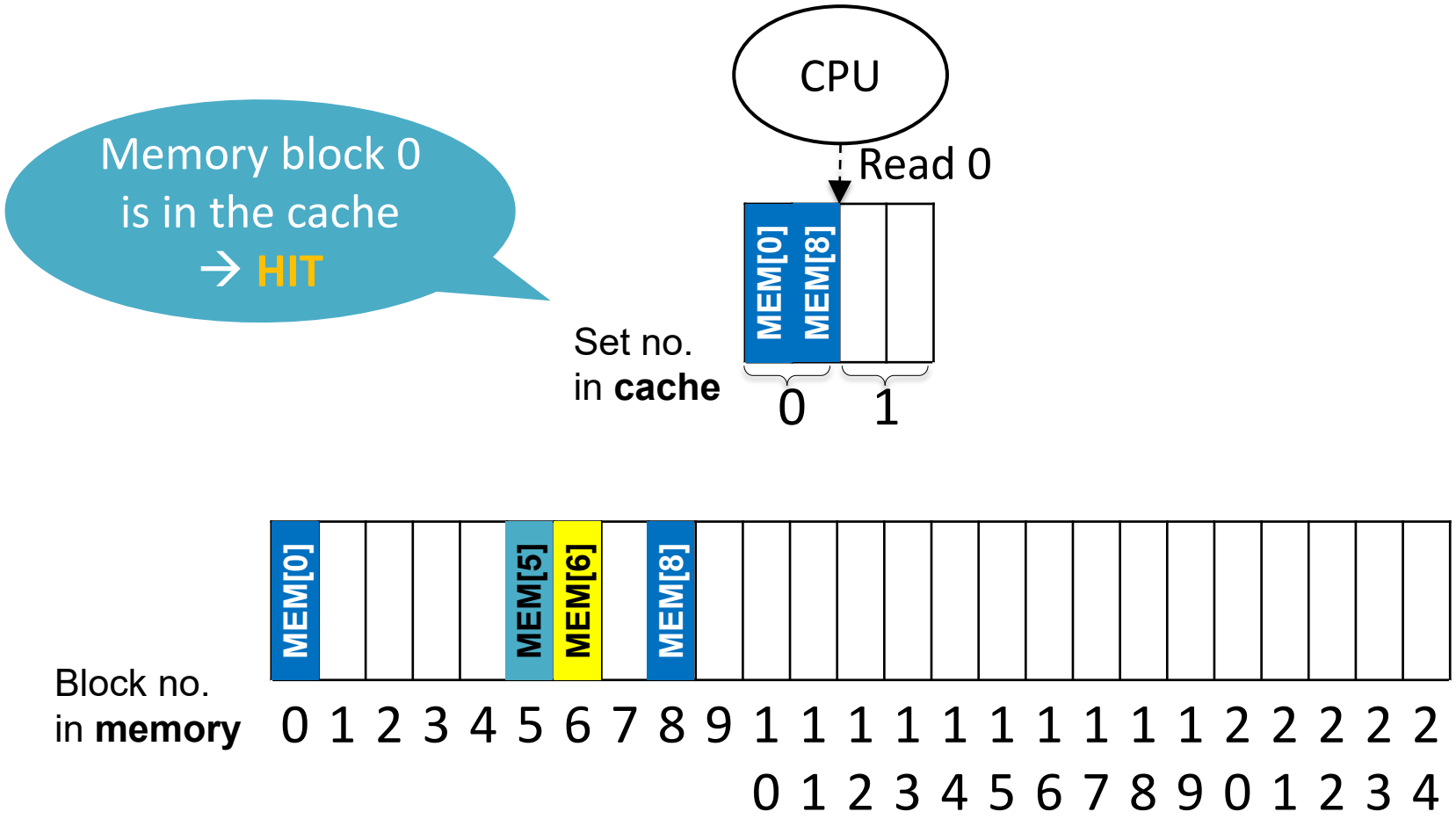
Miss Example: 2-way

- **Memory block access sequence: 0, 8, 0, 6, 5**



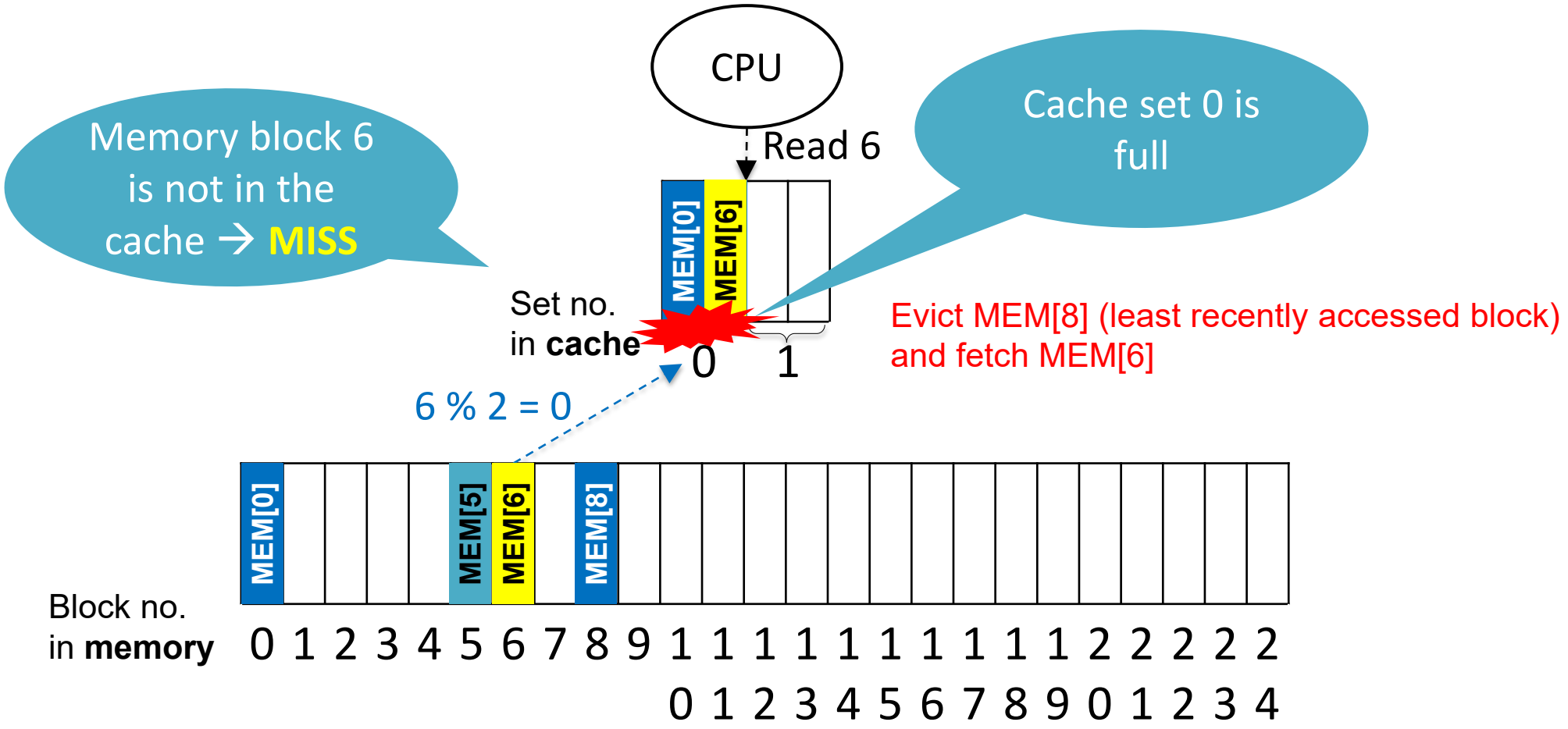
Miss Example: 2-way

- **Memory block access sequence: 0, 8, 0, 6, 5**



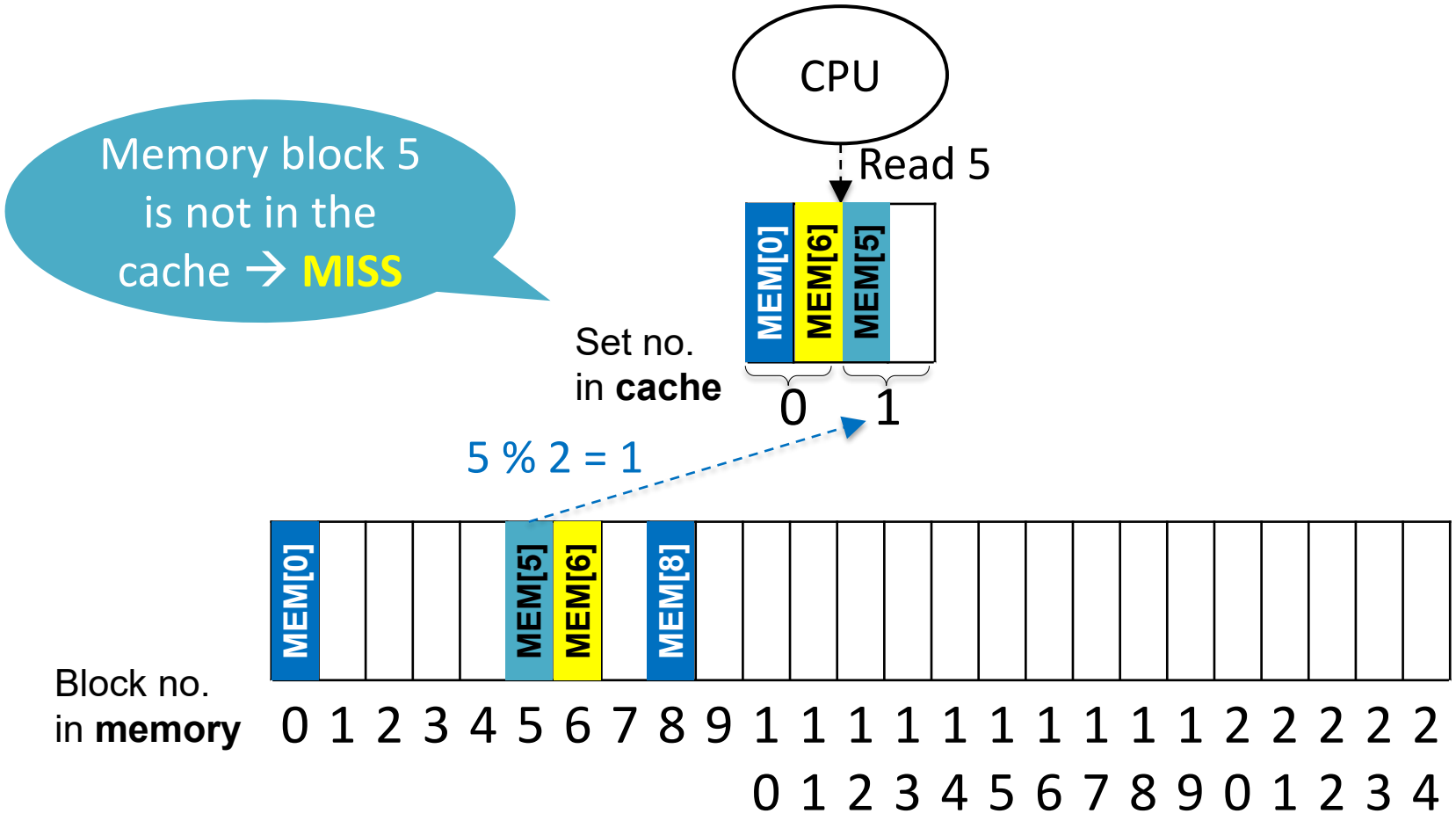
Miss Example: 2-way

- **Memory block access sequence: 0, 8, 0, 6, 5**



Miss Example: 2-way

- **Memory block access sequence: 0, 8, 0, 6, 5**

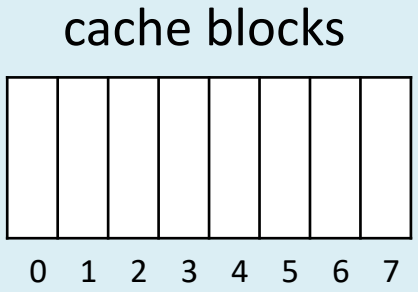


Cache Miss Classification: The 3 C's

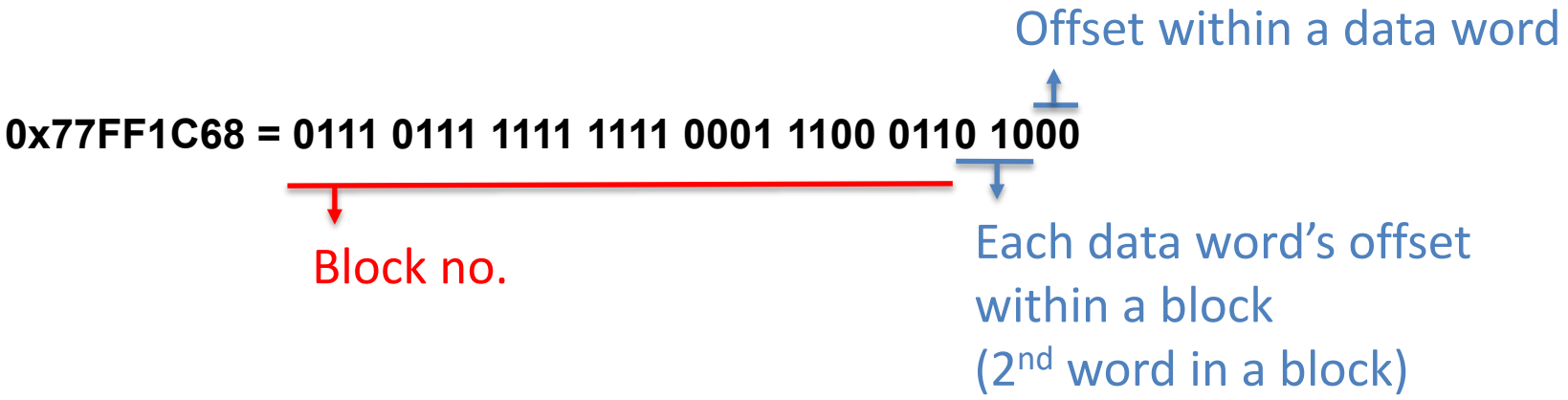
- **Compulsory (cold) Misses**
  - On the 1st reference to a block
  - Related to # blocks accessed by a code, not related to the configuration of a cache
- **Capacity Misses**
  - The program's working set size exceeds the cache capacity
- **Conflict Misses**
  - Multiple memory blocks map to the same set in set-associative caches

More Detailed Mapping Example

- We have a cache that has following configuration
  - Consists of 8 cache blocks (lines)
  - A data word is 4-byte
  - A block is 32-byte (8 words per block)
  - **Direct mapped**
- We load a word from 0x77FF1C68

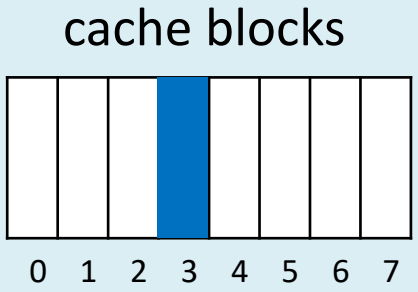


- 1st step: Find data block no. that the word belongs to

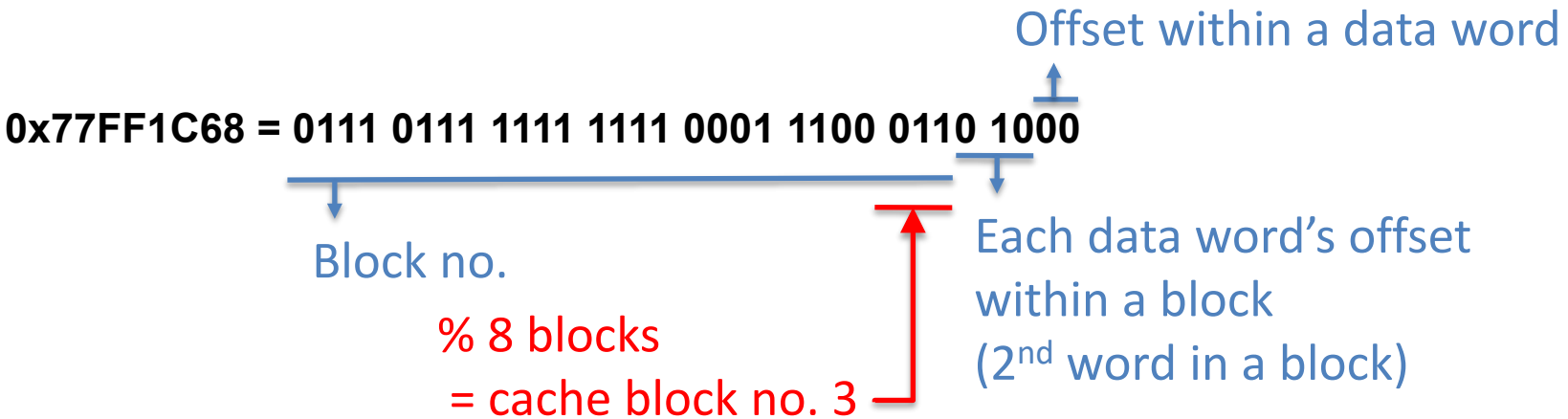


More Detailed Mapping Example

- We have a cache that has following configuration
  - Consists of 8 cache blocks (lines)
  - A data word is 4-byte
  - A block is 32-byte (8 words per block)
  - **Direct mapped**
- We load a word from 0x77FF1C68

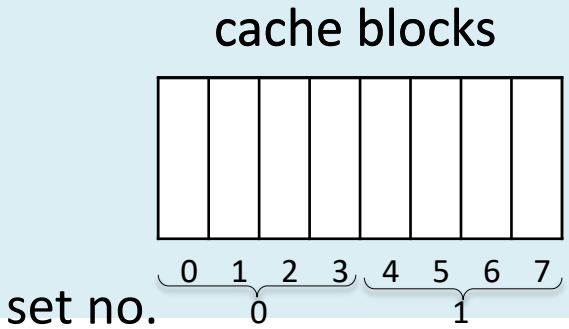


- 2nd step: Map the block no. to cache block no.



More Detailed Mapping Example

- We have a cache that has following configuration
  - Consists of 8 cache blocks (lines)
  - A data word is 4-byte
  - A block is 32-byte (8 words per block)
  - **4-way Set-Associative**
- We load a word from 0x77FF1C68

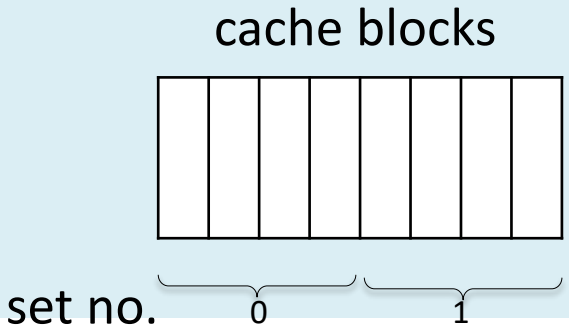


- 1st step: Calculate how many sets exist

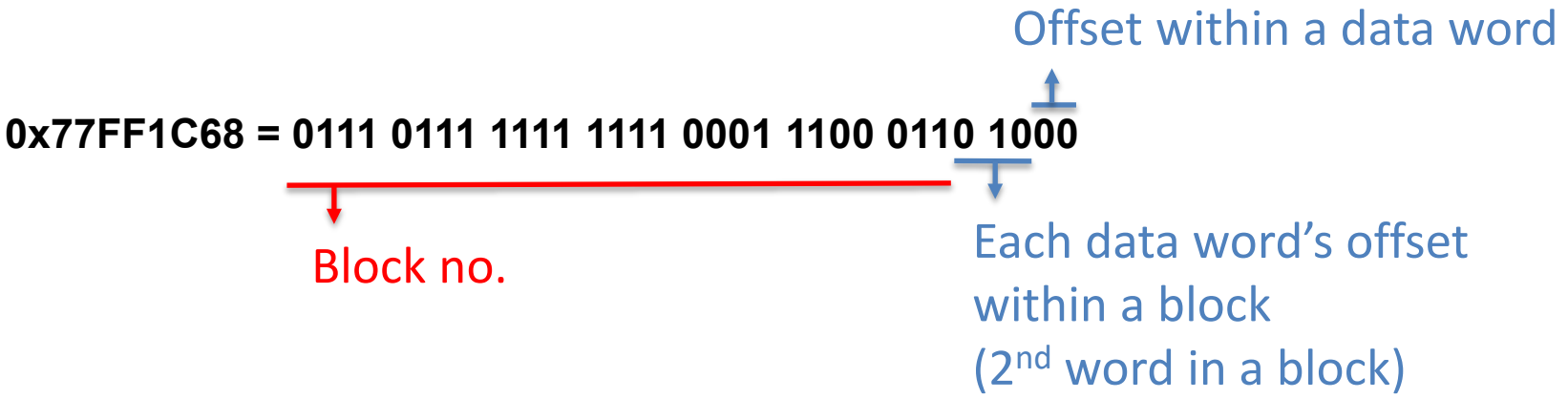
  8 blocks / 4 entries per set = 2 sets

More Detailed Mapping Example

- We have a cache that has following configuration
  - Consists of 8 cache blocks (lines)
  - A data word is 4-byte
  - A block is 32-byte (8 words per block)
  - **4-way Set-Associative**
- We load a word from 0x77FF1C68

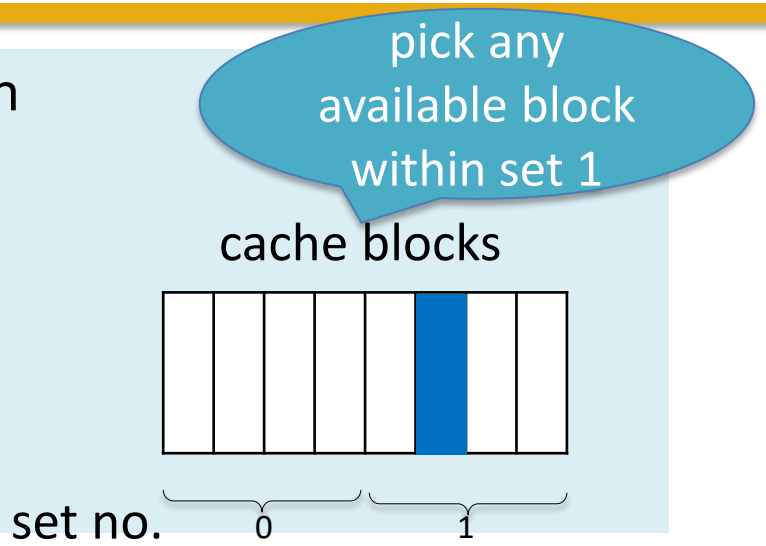


- 2nd step: Find data block no. that the word belongs to

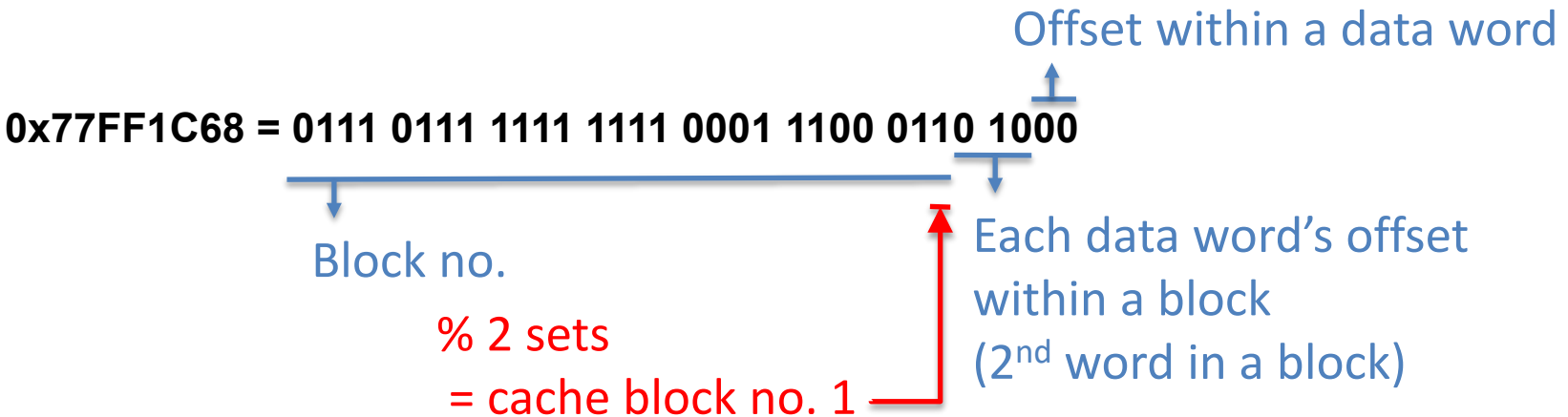


More Detailed Mapping Example

- We have a cache that has following configuration
  - Consists of 8 cache blocks (lines)
  - A data word is 4-byte
  - A block is 32-byte (8 words per block)
  - **4-way Set-Associative**
- We load a word from 0x77FF1C68

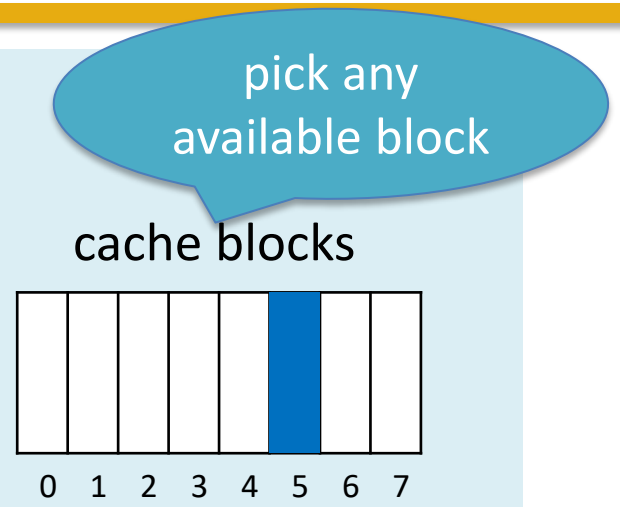


- 3rd step: Map the block no. to cache block no.



More Detailed Mapping Example

- We have a cache that has following configuration
  - Consists of 8 cache blocks (lines)
  - A data word is 4-byte
  - A block is 32-byte (8 words per block)
  - **Fully-Associative**
- We load a word from 0x77FF1C68



- 1st and all step: Find any available cache block and map

Cache Replacement Policy

- **When loading a new block (on miss), if the cache is already full, which block should be replaced (in the set)?**
  - Random: Replace a randomly chosen line
  - FIFO: Replace the oldest line
  - LRU (Least Recently Used): Replace the least recently used line
  - Many others: Round Robin, LIFO, MRU, etc..

LRU Example

- **Cache configuration:**
  - Size: 4 blocks
  - Mapping: fully-associative
  - Replacement: LRU
- **Mem access sequence: (each character indicates a block address)**
  - ABCDCDECG

Accesses		A	B	C	D	C	D	E	C	G
priority order (LRU)	0 (MRU)	A								
	1									
	2									
	3 (LRU)									
cache miss		**m**								

LRU Example

- **Cache configuration:**
  - Size: 4 blocks
  - Mapping: fully-associative
  - Replacement: LRU
- **Mem access sequence: (each character indicates a block address)**
  - ABCDCDECG

Accesses		A	B	C	D	C	D	E	C	G
priority order (LRU)	0 (MRU)	A	B							
	1		A							
	2									
	3 (LRU)									
cache miss		**m**	**m**							

LRU Example

- **Cache configuration:**
  - Size: 4 blocks
  - Mapping: fully-associative
  - Replacement: LRU
- **Mem access sequence: (each character indicates a block address)**
  - ABCDCDECG

Accesses		A	B	C	D	C	D	E	C	G
priority order (LRU)	0 (MRU)	A	B	C						
	1		A	B						
	2			A						
	3 (LRU)									
cache miss		**m**	**m**	**m**						

LRU Example

- **Cache configuration:**
  - Size: 4 blocks
  - Mapping: fully-associative
  - Replacement: LRU
- **Mem access sequence: (each character indicates a block address)**
  - ABCDCDECG

Accesses		A	B	C	D	C	D	E	C	G
priority order (LRU)	0 (MRU)	A	B	C	D					
	1		A	B	C					
	2			A	B					
	3 (LRU)				A					
cache miss		**m**	**m**	**m**	**m**					

LRU Example

- **Cache configuration:**
  - Size: 4 blocks
  - Mapping: fully-associative
  - Replacement: LRU
- **Mem access sequence: (each character indicates a block address)**
  - ABCDCDECG

Accesses		A	B	C	D	C	D	E	C	G
priority order (LRU)	0 (MRU)	A	B	C	D	C				
	1		A	B	C	D				
	2			A	B	B				
	3 (LRU)				A	A				
cache miss		**m**	**m**	**m**	**m**	h				

LRU Example

- **Cache configuration:**
  - Size: 4 blocks
  - Mapping: fully-associative
  - Replacement: LRU
- **Mem access sequence: (each character indicates a block address)**
  - ABCDCDECG

Accesses		A	B	C	D	C	D	E	C	G
priority order (LRU)	0 (MRU)	A	B	C	D	C	D			
	1		A	B	C	D	C			
	2			A	B	B	B			
	3 (LRU)				A	A	A			
cache miss		**m**	**m**	**m**	**m**	h	h			

LRU to be replaced for E

LRU Example

- **Cache configuration:**
  - Size: 4 blocks
  - Mapping: fully-associative
  - Replacement: LRU
- **Mem access sequence: (each character indicates a block address)**
  - ABCDCDECG

Accesses		A	B	C	D	C	D	E	C	G
priority order (LRU)	0 (MRU)	A	B	C	D	C	D	E		
	1		A	B	C	D	C	D		
	2			A	B	B	B	C		
	3 (LRU)				A	A	A	B		
cache miss		**m**	**m**	**m**	**m**	h	h	**m**		

LRU Example

- **Cache configuration:**
  - Size: 4 blocks
  - Mapping: fully-associative
  - Replacement: LRU
- **Mem access sequence: (each character indicates a block address)**
  - ABCDCDECG

Accesses		A	B	C	D	C	D	E	C	G
priority order (LRU)	0 (MRU)	A	B	C	D	C	D	E	C	
	1		A	B	C	D	C	D	E	
	2			A	B	B	B	C	D	
	3 (LRU)				A	A	A	B	B	
cache miss		**m**	**m**	**m**	**m**	h	h	**m**	h	

LRU to be replaced for G

LRU Example

- **Cache configuration:**
  - Size: 4 blocks
  - Mapping: fully-associative
  - Replacement: LRU
- **Mem access sequence: (each character indicates a block address)**
  - ABCDCDECG

Accesses		A	B	C	D	C	D	E	C	G
priority order (LRU)	0 (MRU)	A	B	C	D	C	D	E	C	G
	1		A	B	C	D	C	D	E	C
	2			A	B	B	B	C	D	E
	3 (LRU)				A	A	A	B	B	D
cache miss		**m**	**m**	**m**	**m**	h	h	**m**	h	**m**

Cache Policies: Cases

- **Allocation policy: do we allocate a block in cache for the missed data?**

- **Read policies:**
  - Read Hit: this is what we want. Only one data read from the cache.

  - Read Miss: needs to fetch from lower level, but just write to the register once after that
    - read-allocate (with replacement policy) vs. no-read-allocate (i.e., cache bypassing)

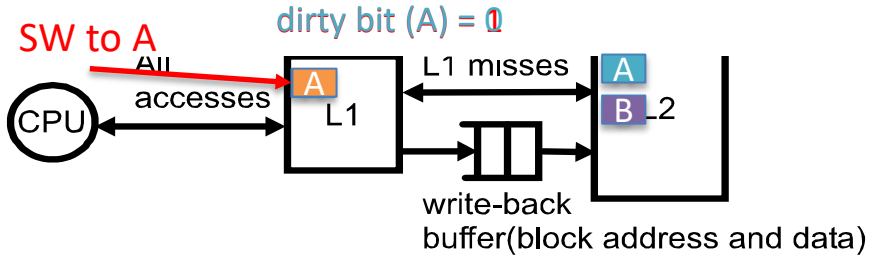
- **Write policies (only for the data cache): consistency & performance tradeoffs**
  - Write Hit: behavior and number of writes depends on write policy
    - Write-through vs. write-back vs. write-evict

  - Write Miss: needs to first read from lower level, then apply write policies
    - Write-allocate (with replacement policy): Write-through vs. Write-back

    - No-write-allocate (bypassing): Write-evict

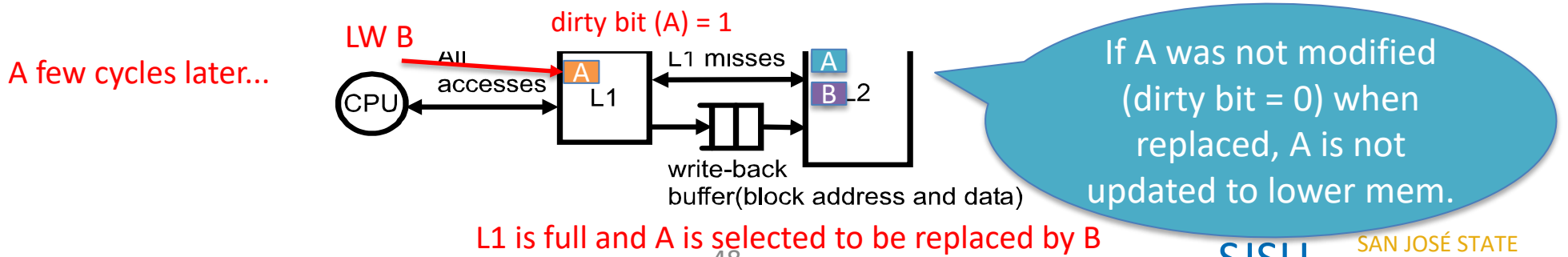
Write Policy: Write-back

- **Write-back: inconsistent with lower level**
  - The value is written only to the cache line.
  - The modified (dirty) cache line is written to the lower level only when it is evicted.
    - 1 dirty bit is needed for each cache line.
  - A write-back buffer to update lower level with evicted dirty blocks
    - Must write a full block at this point since we do not know which word is modified
  - Example: assume cache block containing word address A is initially in the cache



Write Policy: Write-back

- **Write-back: inconsistent with lower level**
  - The value is written only to the cache line.
  - The modified (dirty) cache line is written to the lower level only when it is evicted.
    - 1 dirty bit is needed for each cache line.
  - A write-back buffer to update lower level with evicted dirty blocks
    - Must write a full block at this point since we do not know which word is modified
  - Example: assume cache block containing word address A is initially in the cache



Let us Conclude

Why do we use multiple cache ways?

Redundancy, flexibility

What is overhead of set associative caches?

Delay, area, cost

Let us Conclude

What are the 3 C's of cache misses?

Compulsory, Capacity, Conflict

What are some replacement policies?

Random, FIFO, LRU, LFU, Round Robin …

SAN JOSÉ STATE UNIVERSITY *powering* SILICON VALLEY