CS 222 Computer Organization & Architecture

Lecture 25 [01.04.2019]

Cache Block Mapping Techniques

John Jose

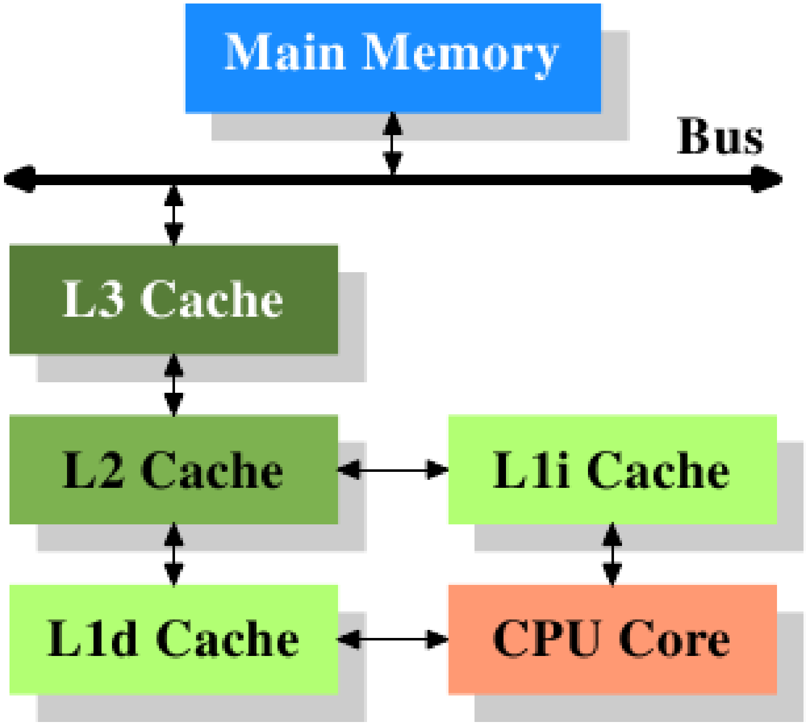
Assistant Professor

Department of Computer Science & Engineering

Indian Institute of Technology Guwahati, Assam.

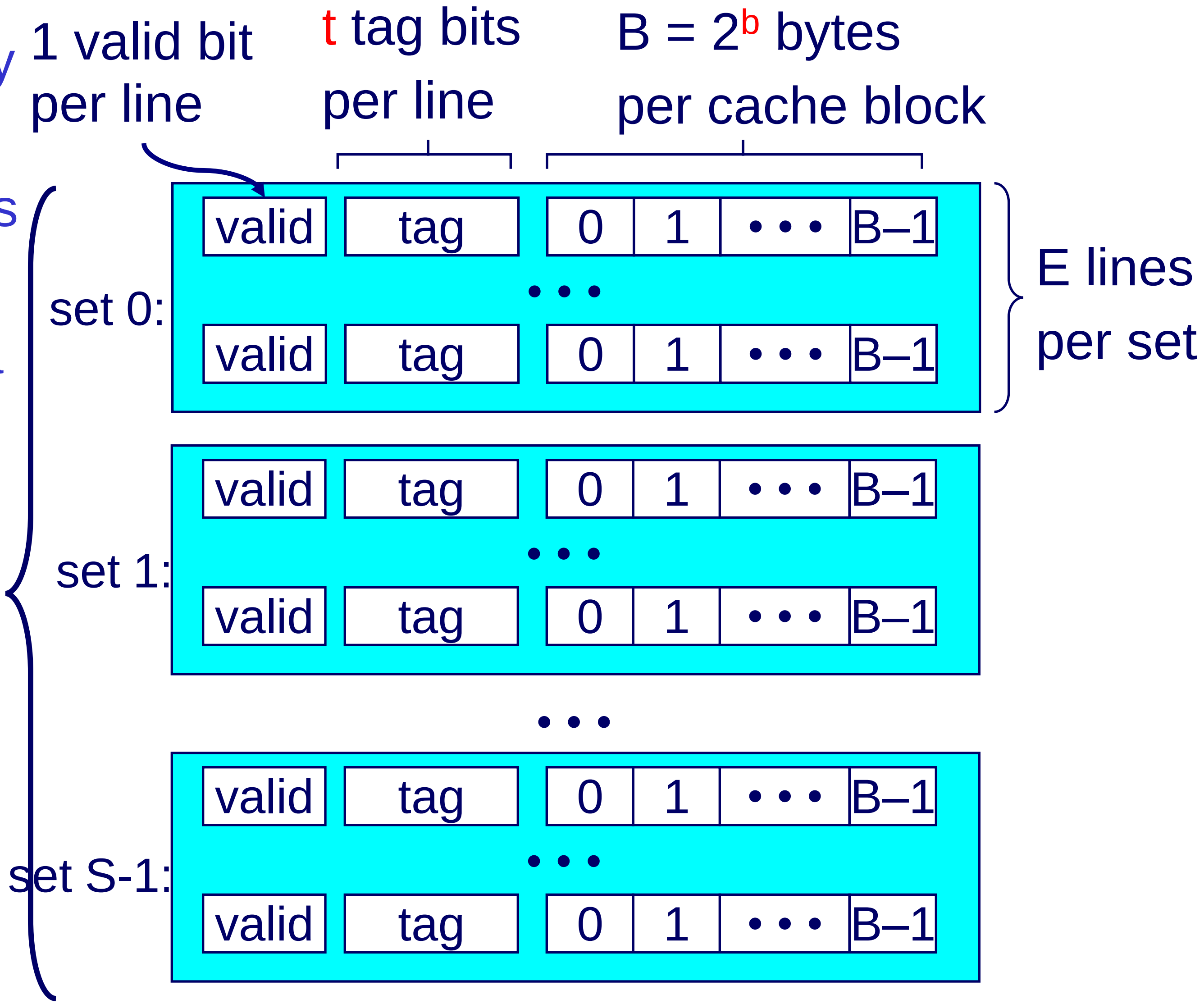
Cache Memory

- Cache memories are small, fast SRAM-based memories managed in hardware by cache controller.
- It hold frequently accessed blocks of main memory
- CPU looks first for data in L1, then in L2, then in main memory.



General Organization of a Cache

- ❖Cache is an array of sets
- ❖Each set contains one or more lines
- ❖Each line holds a block of data



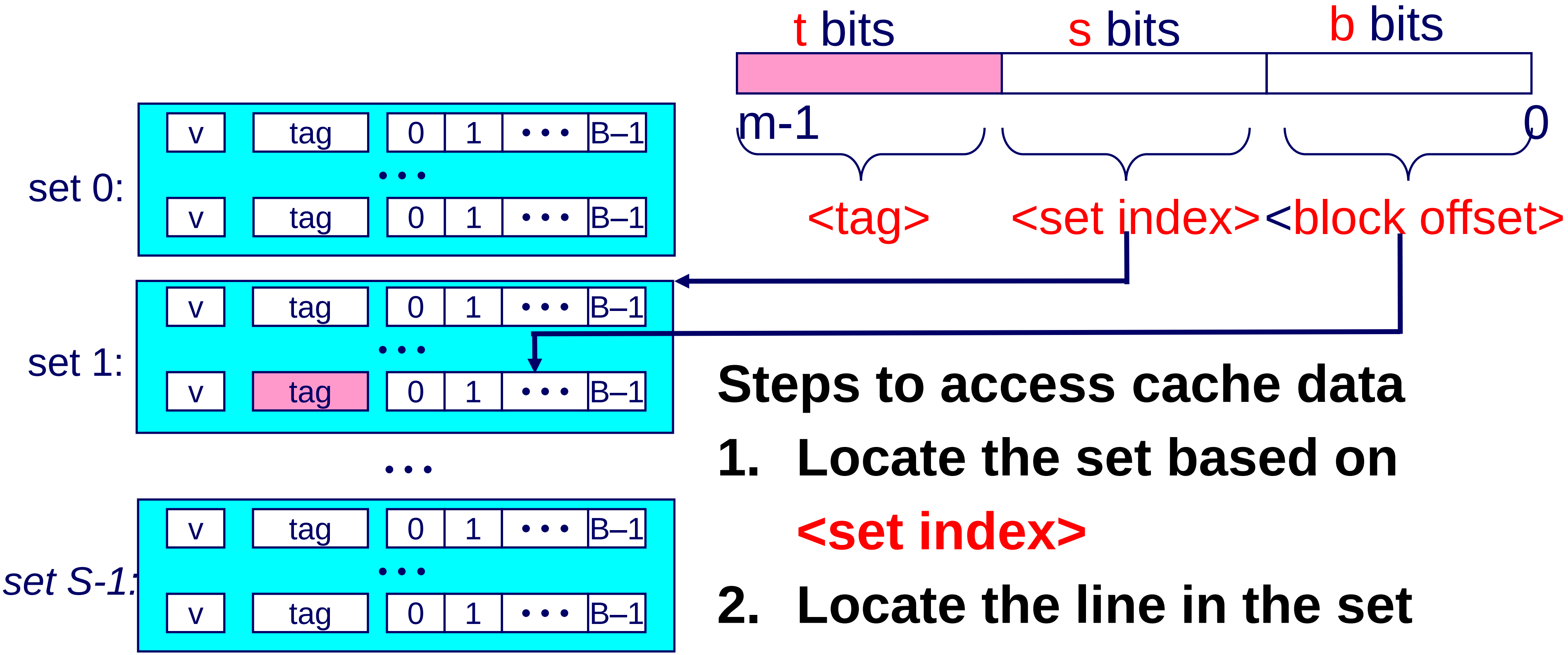
Cache size: C = B x E x S data bytes

Basic Terminologies

- **Block :** Minimum unit of information that can be either present or not present in a cache level
- **Hit :** An access where the data requested by the processor is present in the cache
- **Miss :** An access where the data requested by the processor is not present in the cache
- **Hit Time :** Time to access the cache memory block and return the data to the processor.
- **Hit Rate / Miss Rate:** Fraction of memory access found (not found) in the cache
- **Miss Penalty :** Time to replace a block in the cache with the corresponding block from the next level.

Addressing Caches

Address A: **1001 0011 0101 1010**



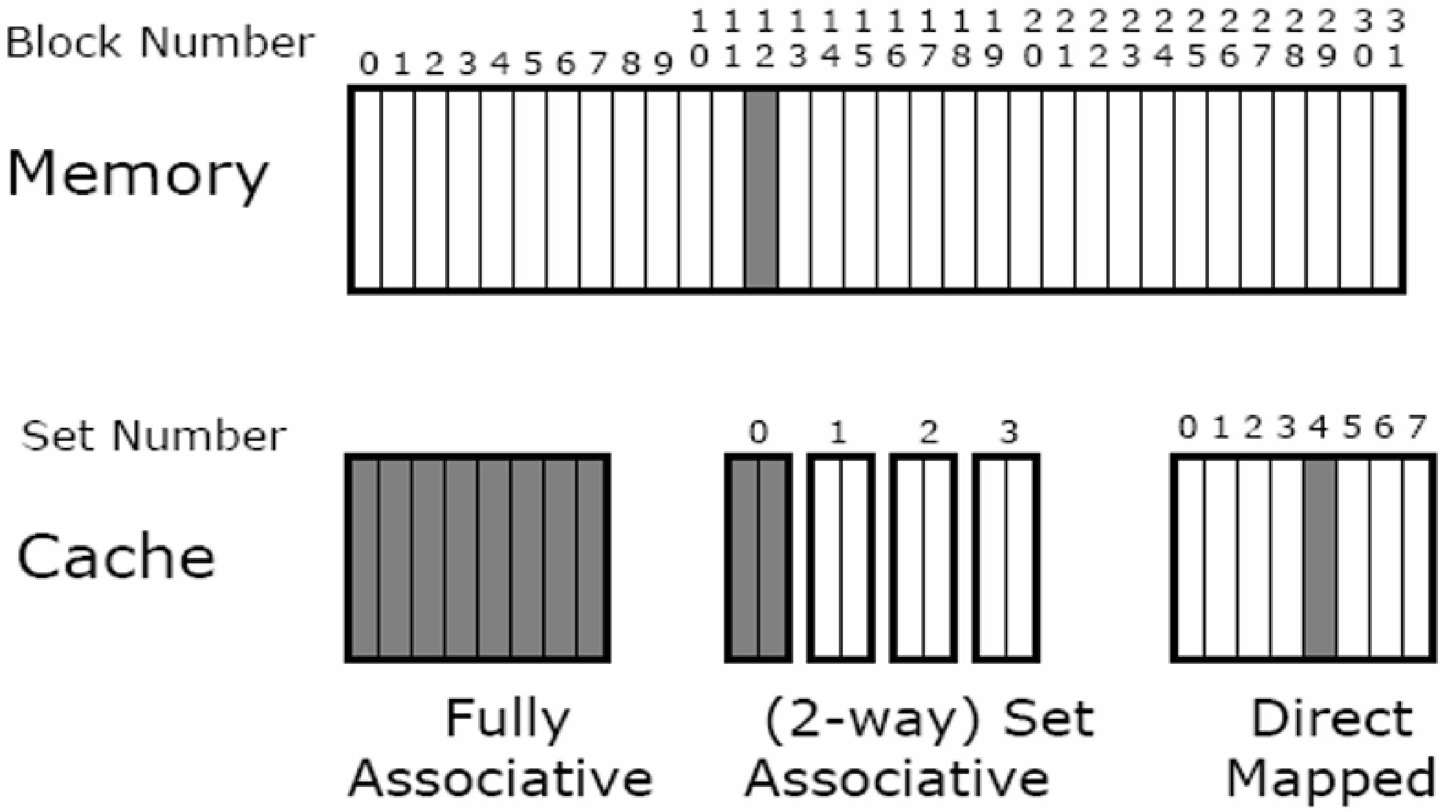
Steps to access cache data

1. **Locate the set based on <set index>**
2. **Locate the line in the set based on <tag>**
3. **Check that the line is valid**
4. **Locate the data in the line based on <block offset>**

Four cache memory design choices

- ❖ Where can a block be placed in the cache?
  - – Block Placement [Mapping]
- ❖ How is a block found if it is in the upper level?
  - – Block Identification
- ❖ Which block should be replaced on a miss?
  - – Block Replacement
- ❖ What happens on a write?
  - – Write Strategy

Block Placement



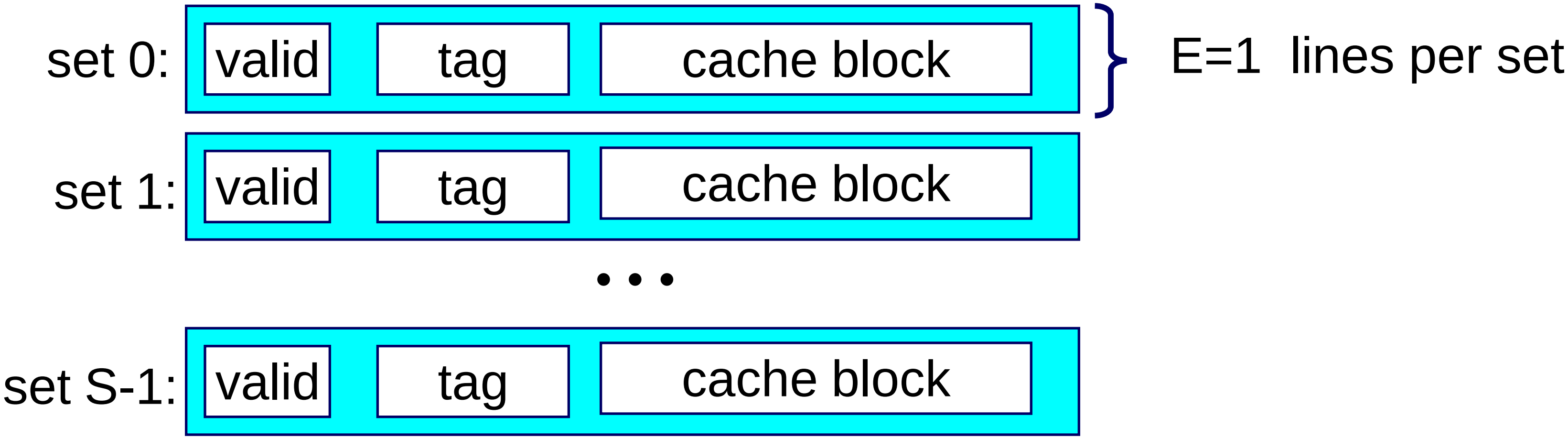
block 12 can be placed	anywhere	anywhere in set 0 *(12 mod 4)*	only into block 4 *(12 mod 8)*

Block Placement

- **Direct mapped**
  - Block can be placed in only one location
  - (Block Number) **Modulo** (Number of blocks in cache)
- **Set associative**
  - Block can be placed in one among a list of locations
  - (Block Number) **Modulo** (Number of sets)
- **Fully associative**
  - Block can be placed anywhere

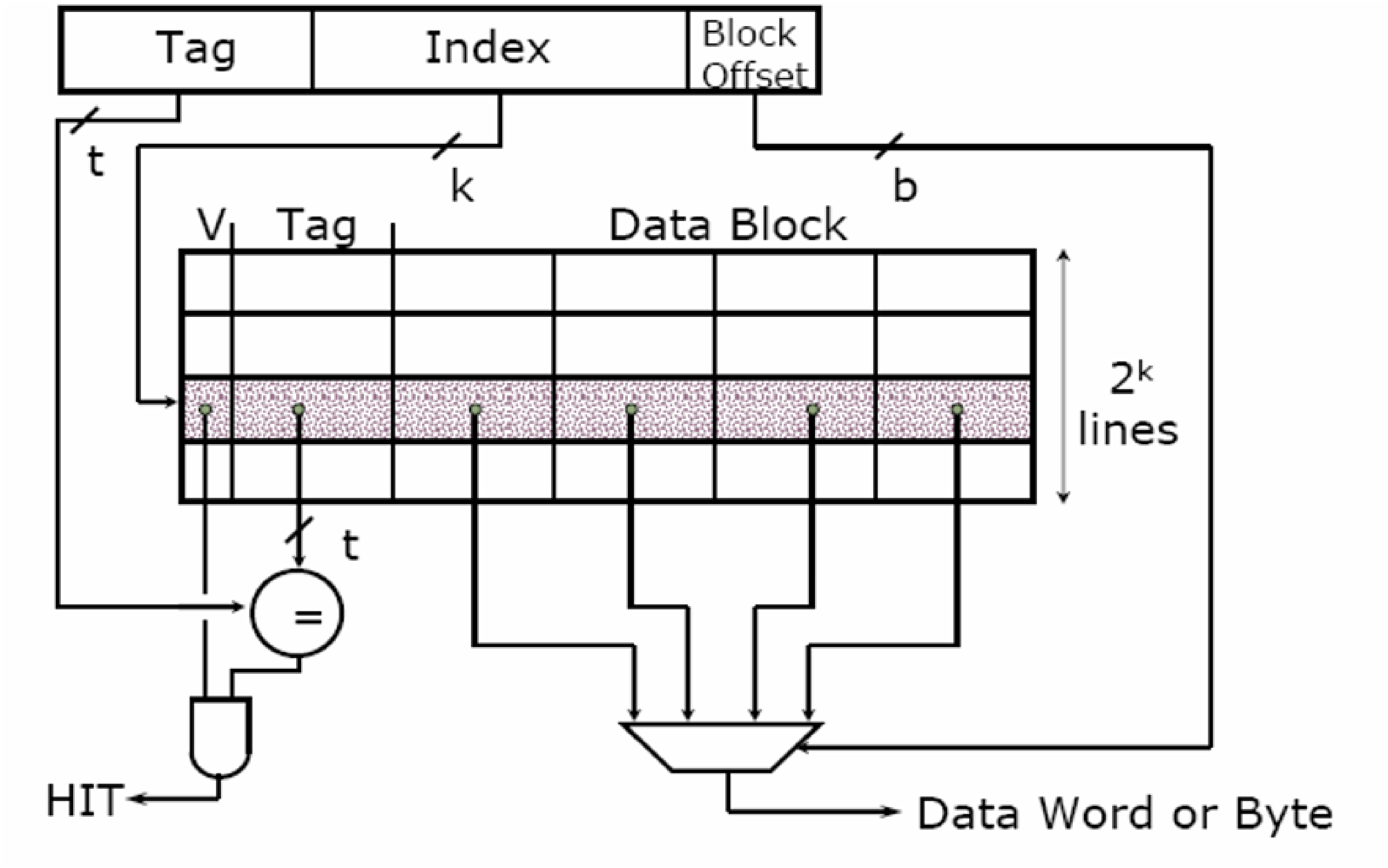
Direct-Mapped Cache

- Simplest kind of cache, easy to build
- Only 1 tag compare required per access
- Characterized by exactly one line per set.



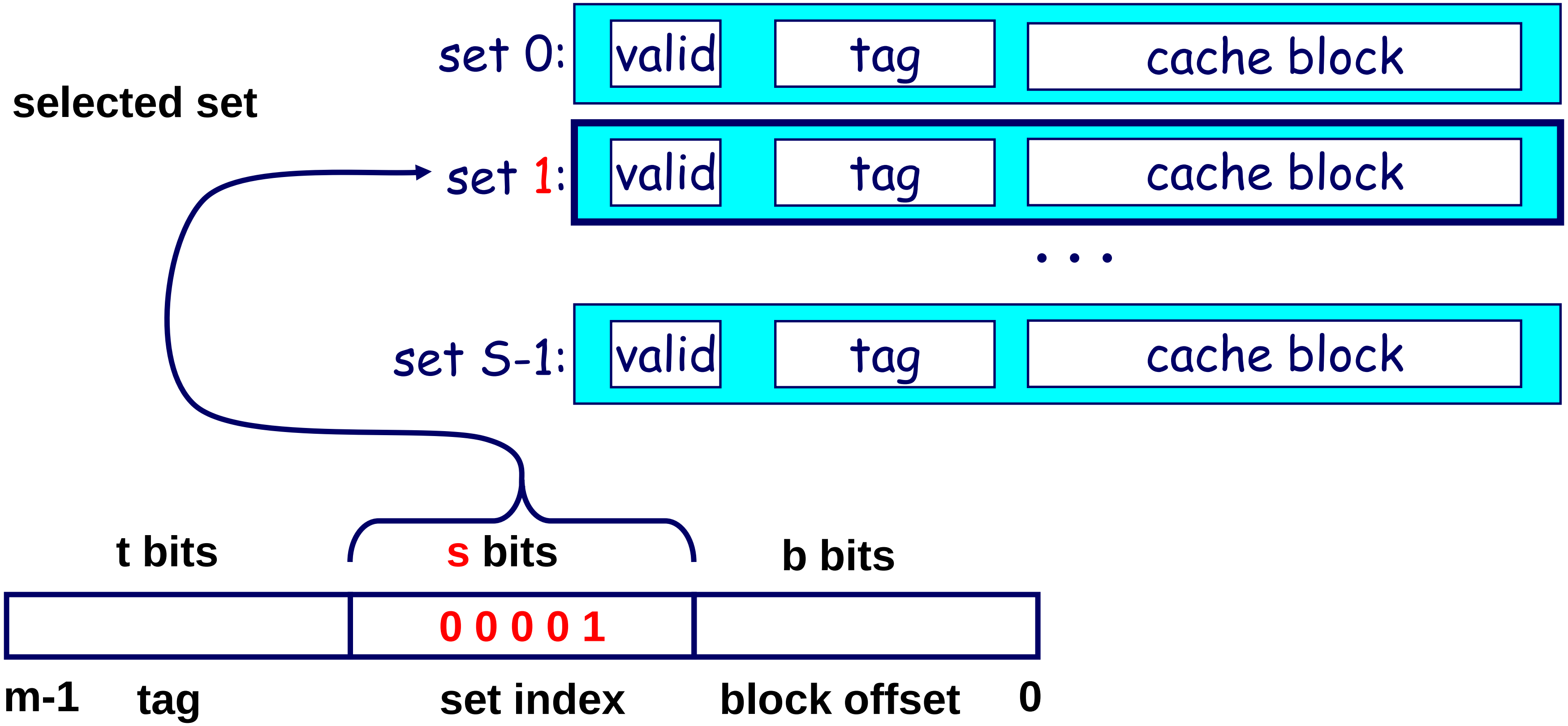
Cache size: C = B x S data bytes

Block Identification – Direct mapped



Accessing Direct-Mapped Caches

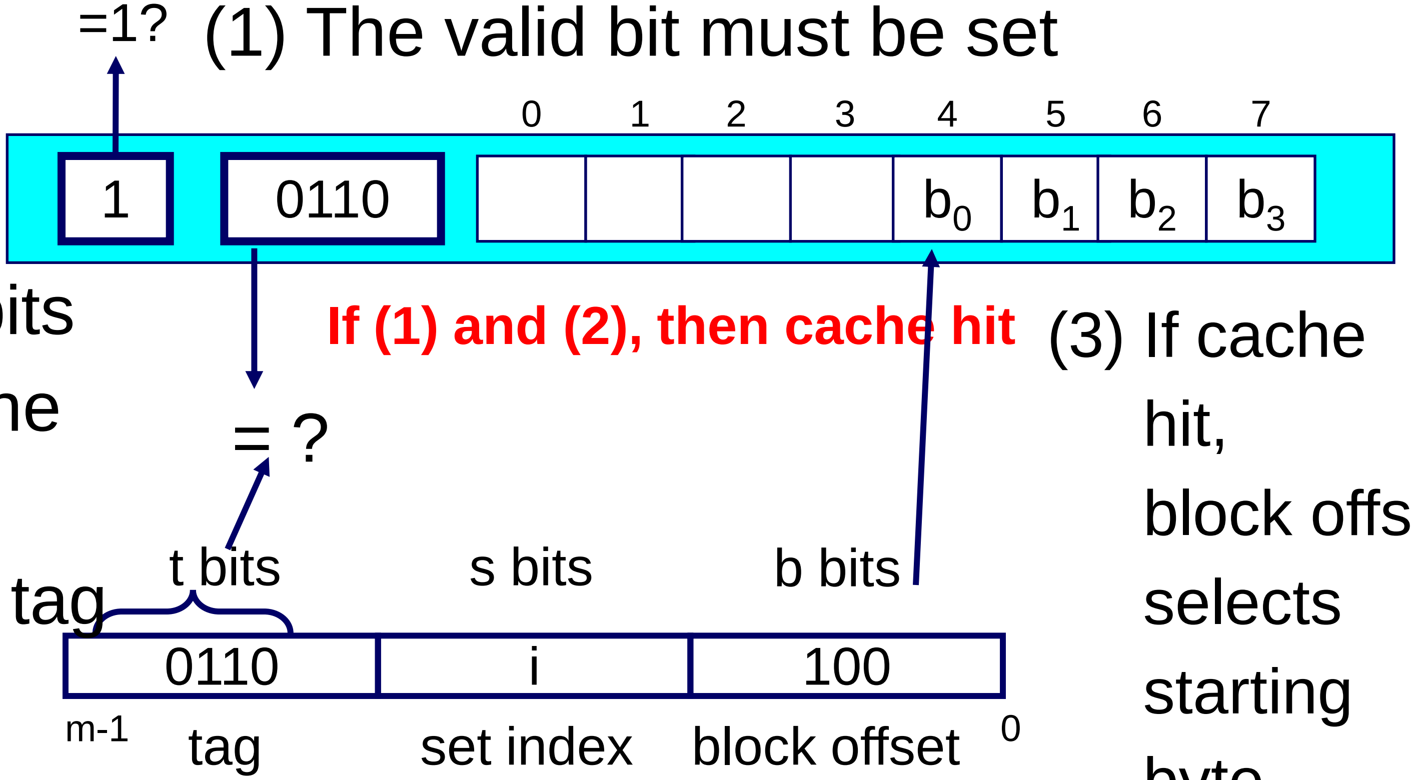
- Set selection is done by the set index bits



Accessing Direct-Mapped Caches

- Line matching and word selection
  - Line matching: Find a valid line in the selected set with a matching tag
  - Word selection: Then extract the word

(1) The valid bit must be set

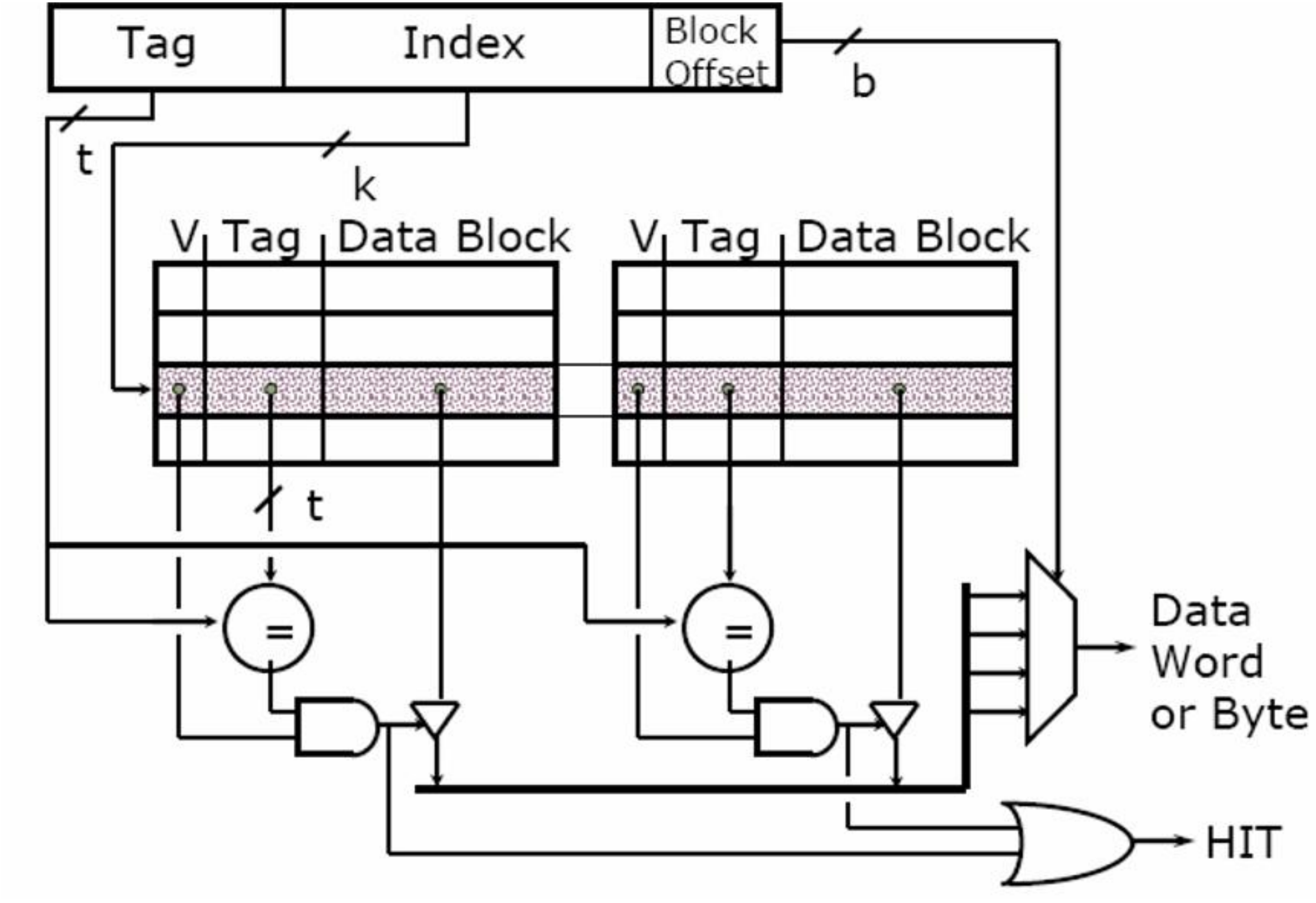


(2) The tag bits in the cache line must match the tag bits in the address

If (1) and (2), then cache hit

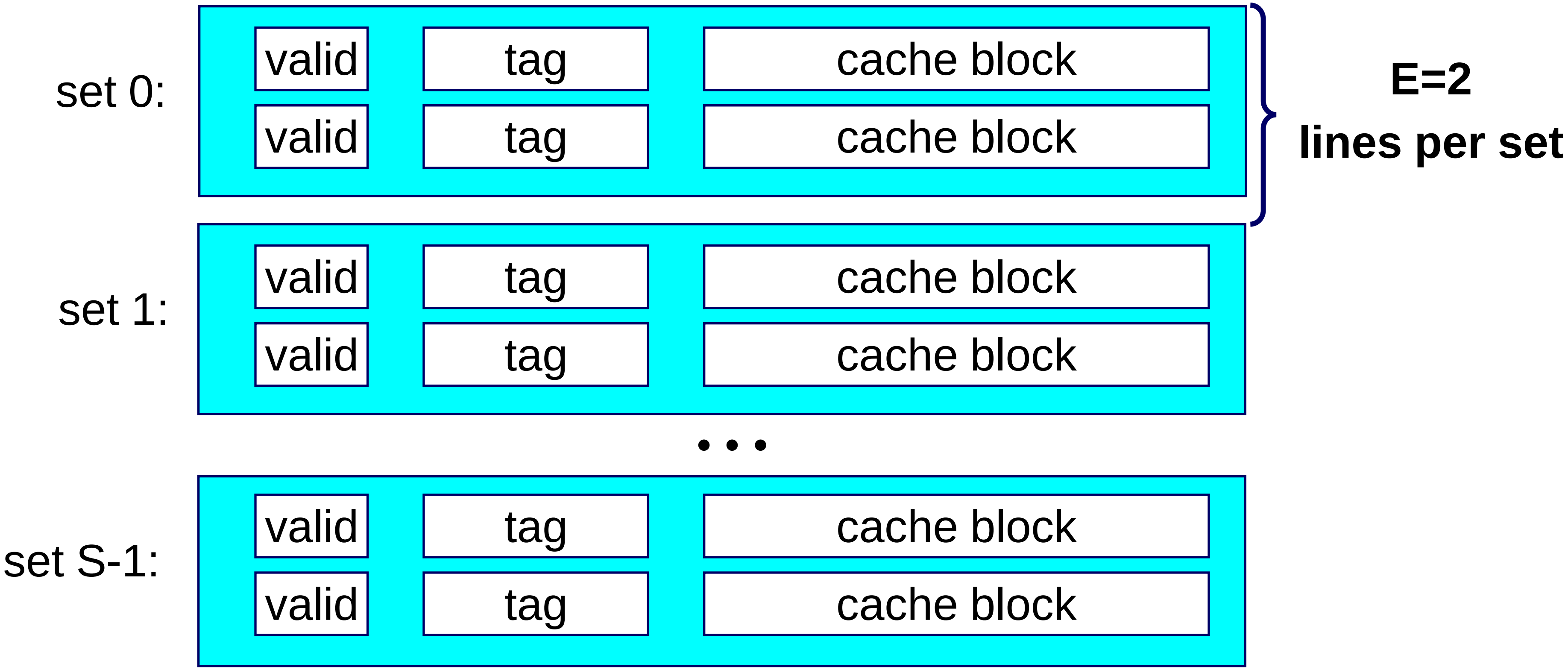
(3) If cache hit, block offset selects starting byte.

Block Identification – Set Associative



Set Associative Cache

- ❖ Characterized by more than one line per set



set 0:

valid tag cache block

valid tag cache block

set 1:

valid tag cache block

valid tag cache block

• • •

set S-1:

valid tag cache block

valid tag cache block

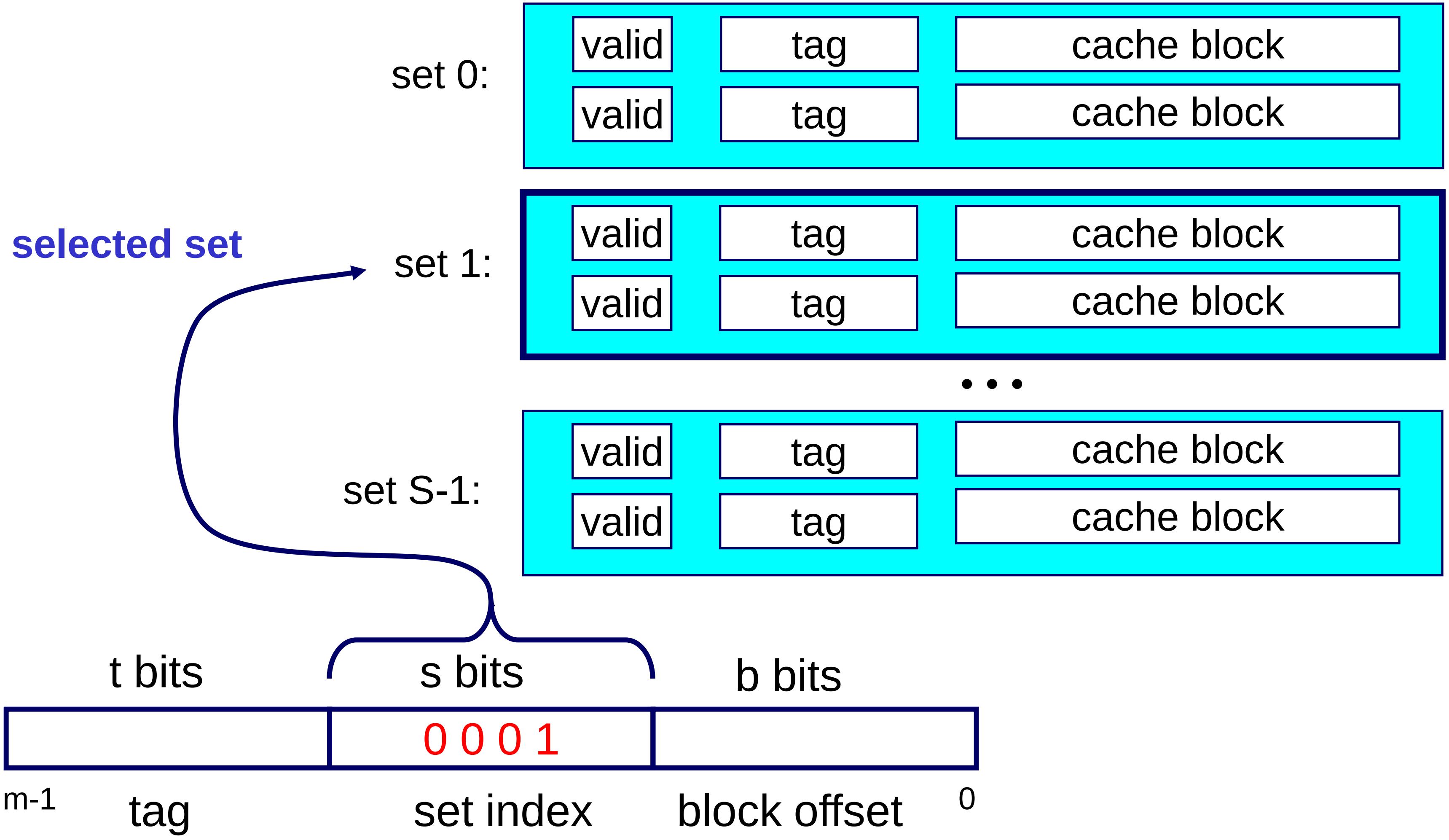
E=2

lines per set

E-way associative cache

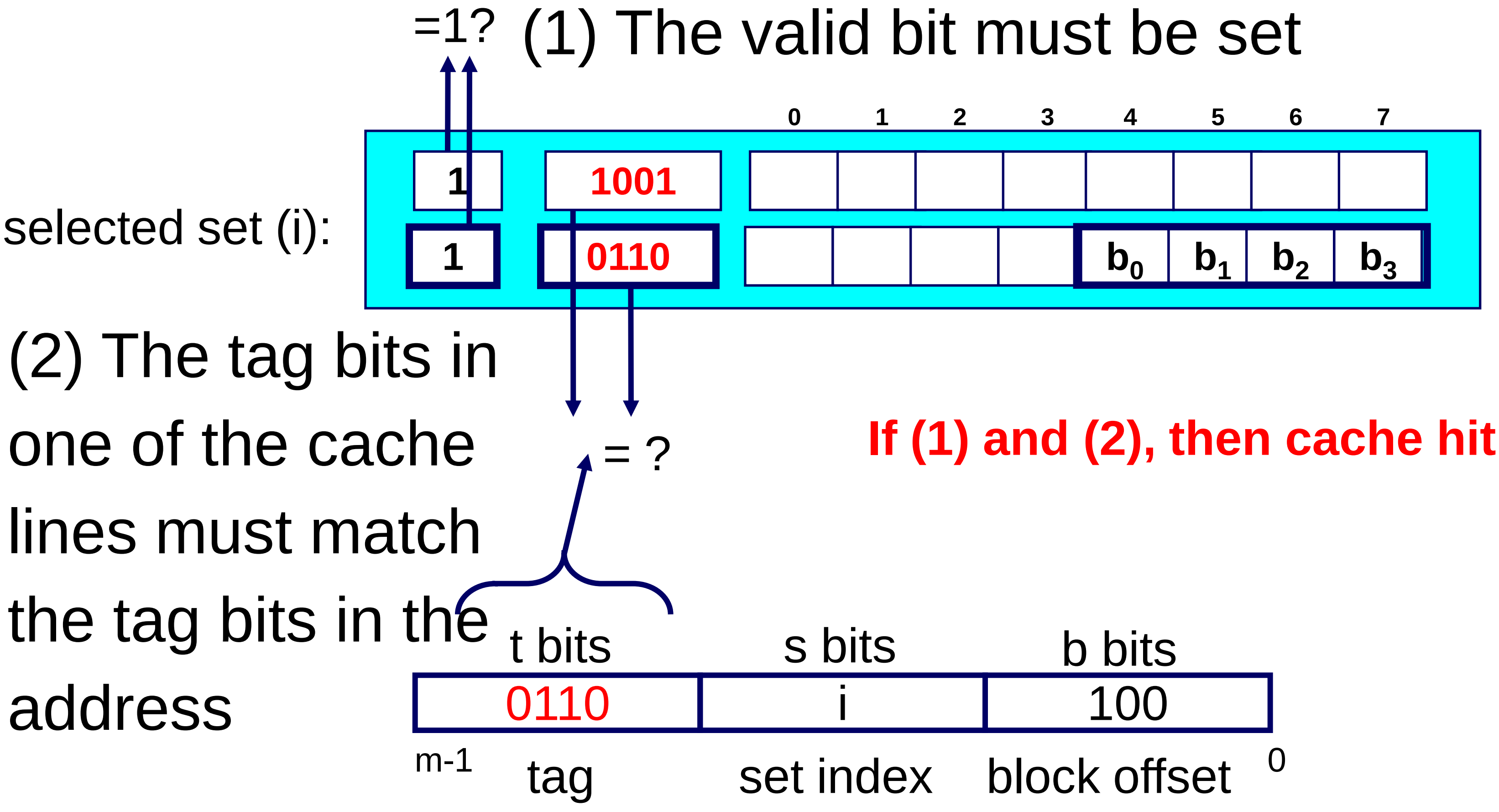
Accessing Set Associative Caches

- Set selection is identical to direct-mapped cache



Accessing Set Associative Caches

- Line matching is done by comparing the tag in each valid line in the selected set.



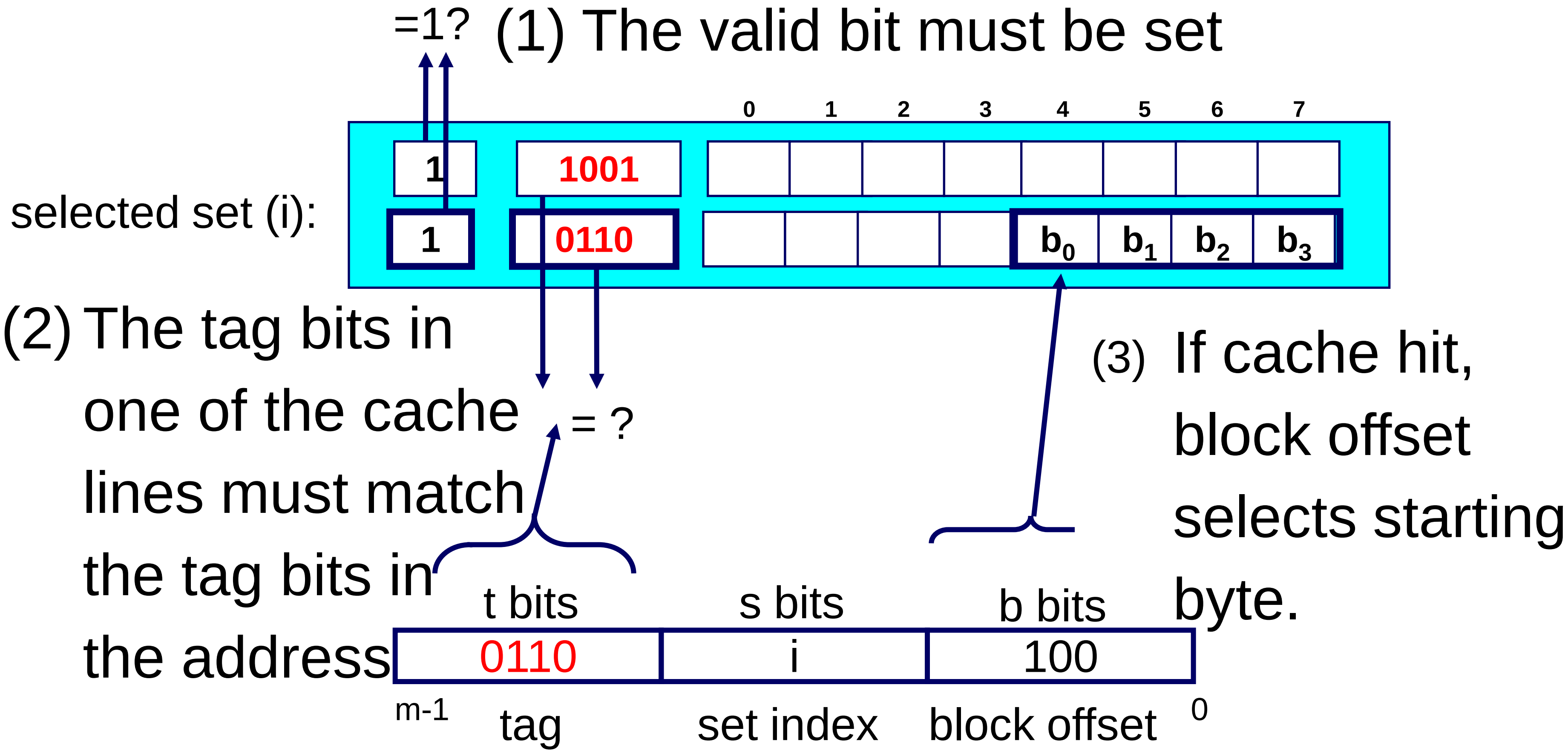
(1) The valid bit must be set

(2) The tag bits in one of the cache lines must match the tag bits in the address

If (1) and (2), then cache hit

Accessing Set Associative Caches

- Word selection is done same as direct mapped cache but chosen only on the line that has produced a hit.



(1) The valid bit must be set

(2) The tag bits in one of the cache lines must match the tag bits in the address

(3) If cache hit, block offset selects starting byte.

Direct Vs Set Associative Cache Simulation

M=16 byte addresses, B=2bytes/block

S=4 sets, E=1 entry/set

t=1	s=2	b=1
x	xx	x

Address trace (reads):

0	[0000]	miss
1	[0001]	hit
7	[0111]	miss
8	[1000]	miss
0	[0000]	miss

4 sets

	v	tag	data
S0	1	0	M[0-1]
S1			
S2			
S3	1	0	M[6-7]

S=2 sets, E=2 entry/set

t=2	s=1	b=1
xx	x	x

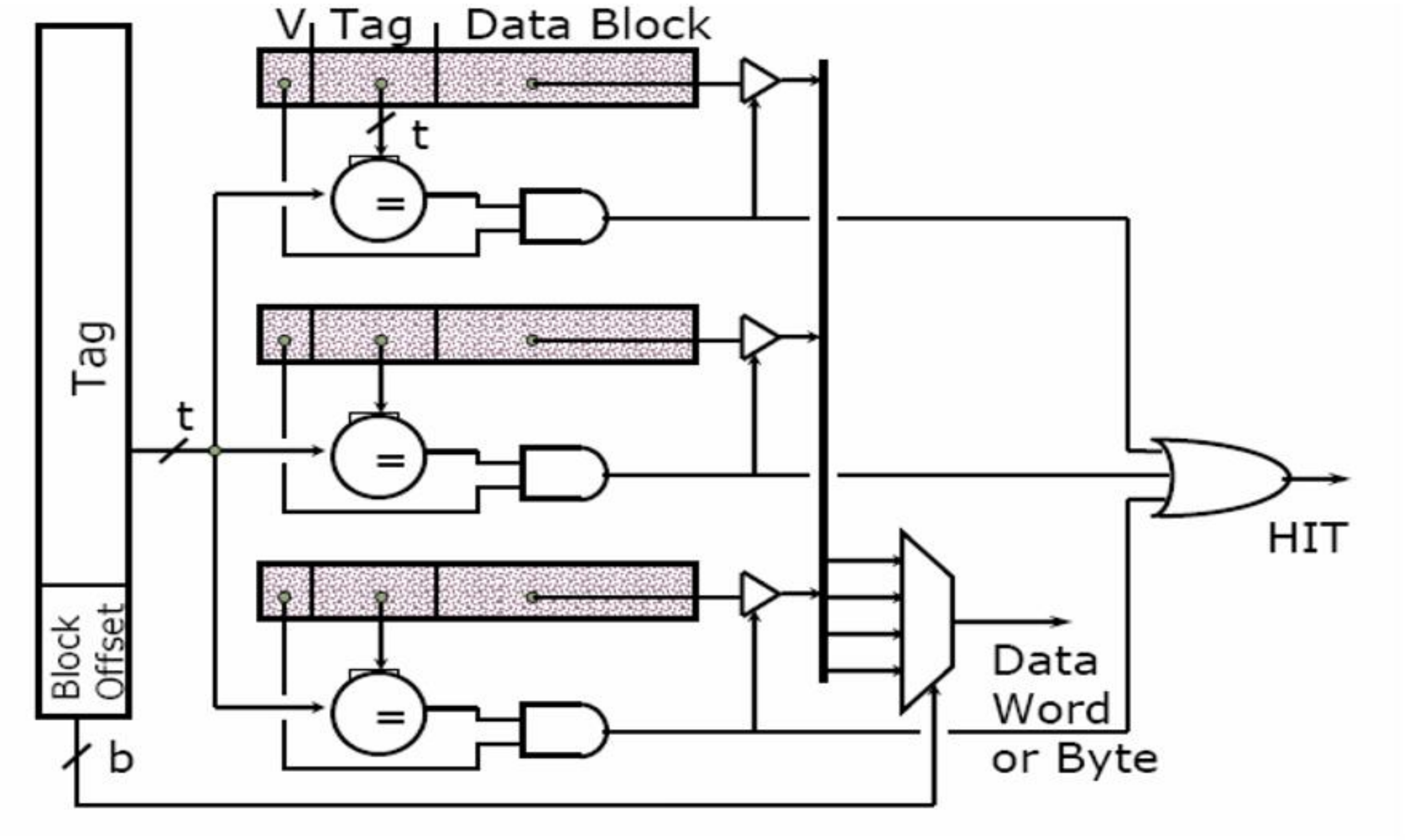
Address trace (reads):

0	[0000]	miss
1	[0001]	hit
7	[0111]	miss
8	[1000]	miss
0	[0000]	hit

2 sets

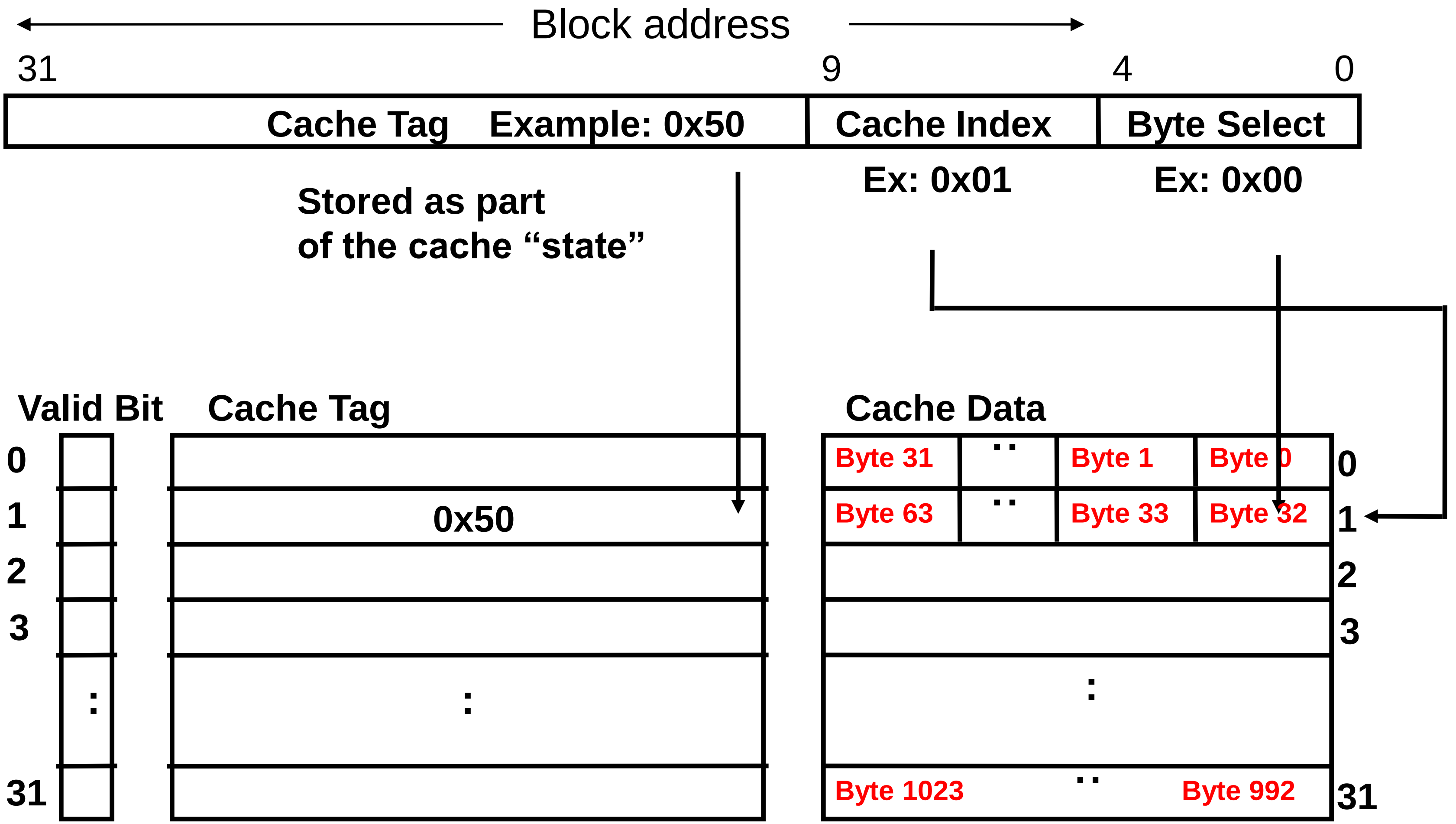
v	tag	data	
1	00	M[0-1]	S0
1	10	M[8-9]	
1	01	M[6-7]	S1
0			

Block Identification – Fully Associative



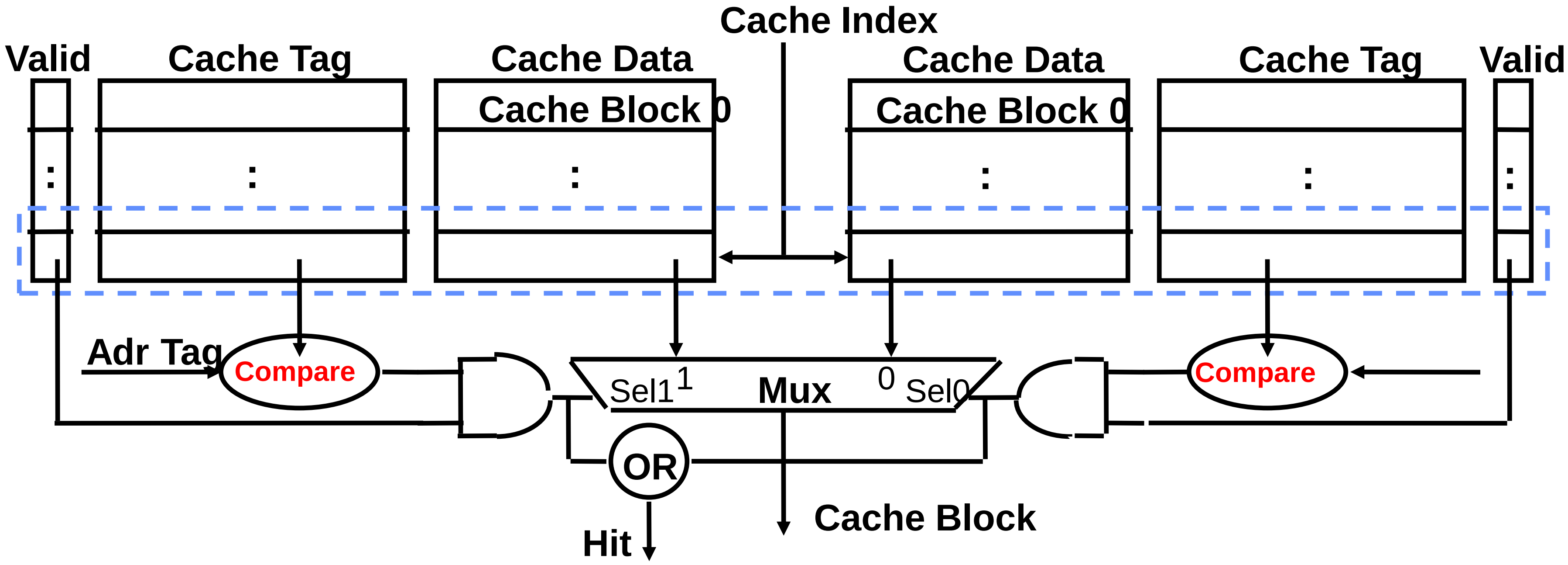
Direct Mapped Cache

Eg: 1KB direct mapped cache with 32 B cache line



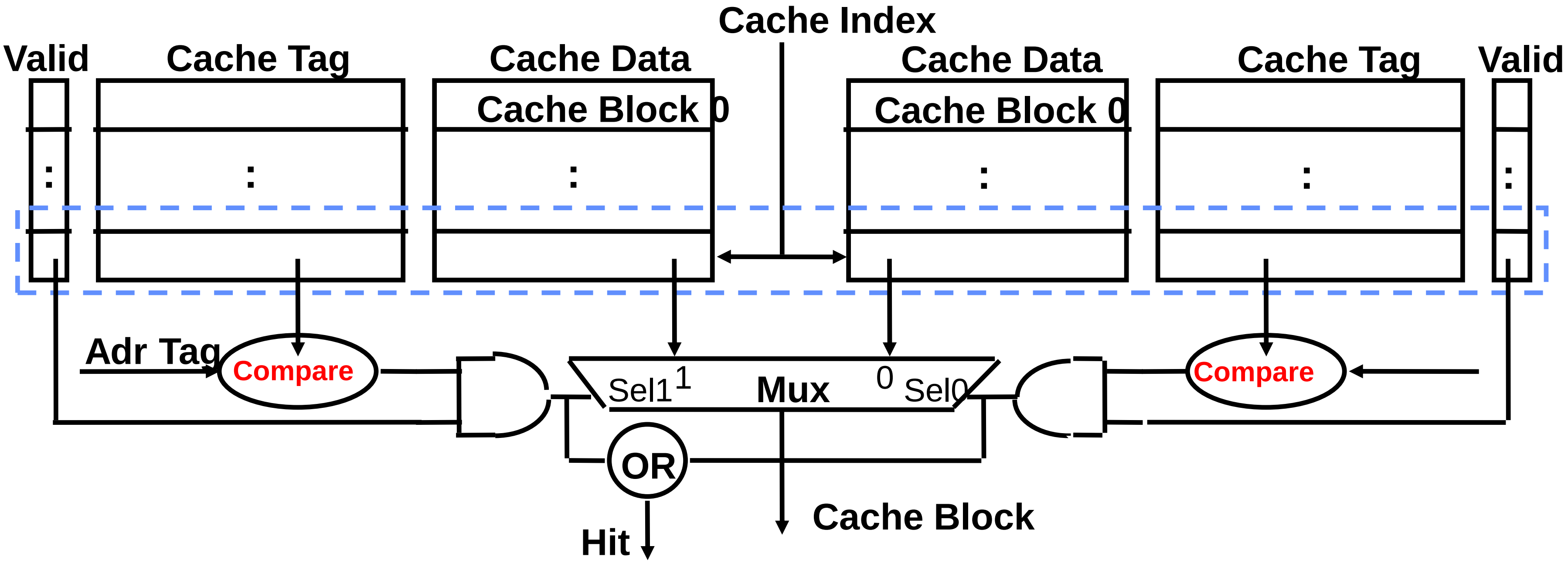
Set Associative Cache

- **N-way set associative**: N direct mapped caches in parallel
- Example: Two-way set associative cache
  - Cache Index selects a set from the cache
  - The two tags in the set are compared to the input in parallel
  - Data is selected based on the tag result

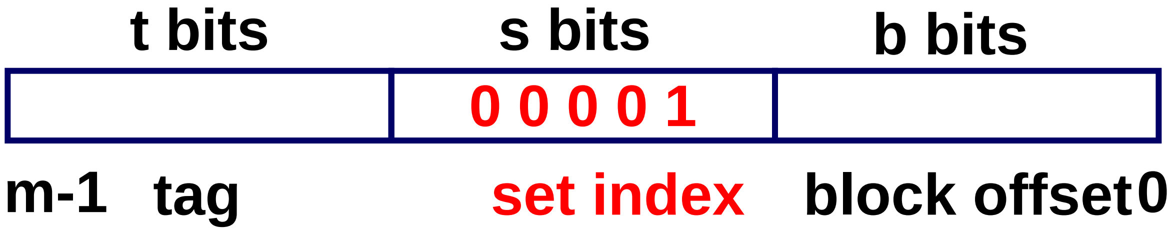


Direct vs Set Associative Cache

- N-way Set Associative Cache versus Direct Mapped Cache:
  - N comparators vs. 1
  - Extra MUX delay for the data
  - Data comes AFTER Hit/Miss decision and set selection
- In a direct mapped cache, Cache Block is available BEFORE Hit/Miss:



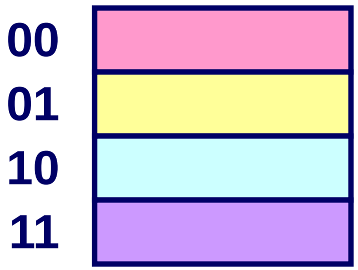
Cache Indexing



- Decoders are used for indexing
- Indexing time depends on decoder size (s: 2^s)
- Smaller number of sets, less indexing time.

Why Use Middle Bits as Index?

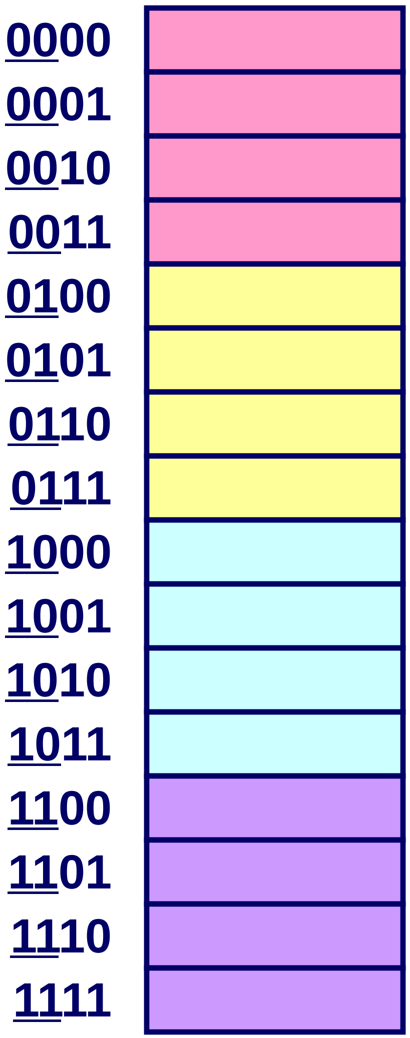
4-line Cache


00
01
10
11

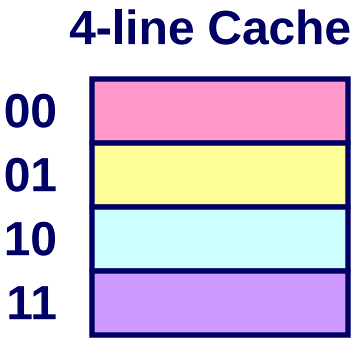
High-Order Bit Indexing

- ❖ Adjacent memory lines would map to same cache entry
- ❖ Poor use of spatial locality

High-Order Bit Indexing

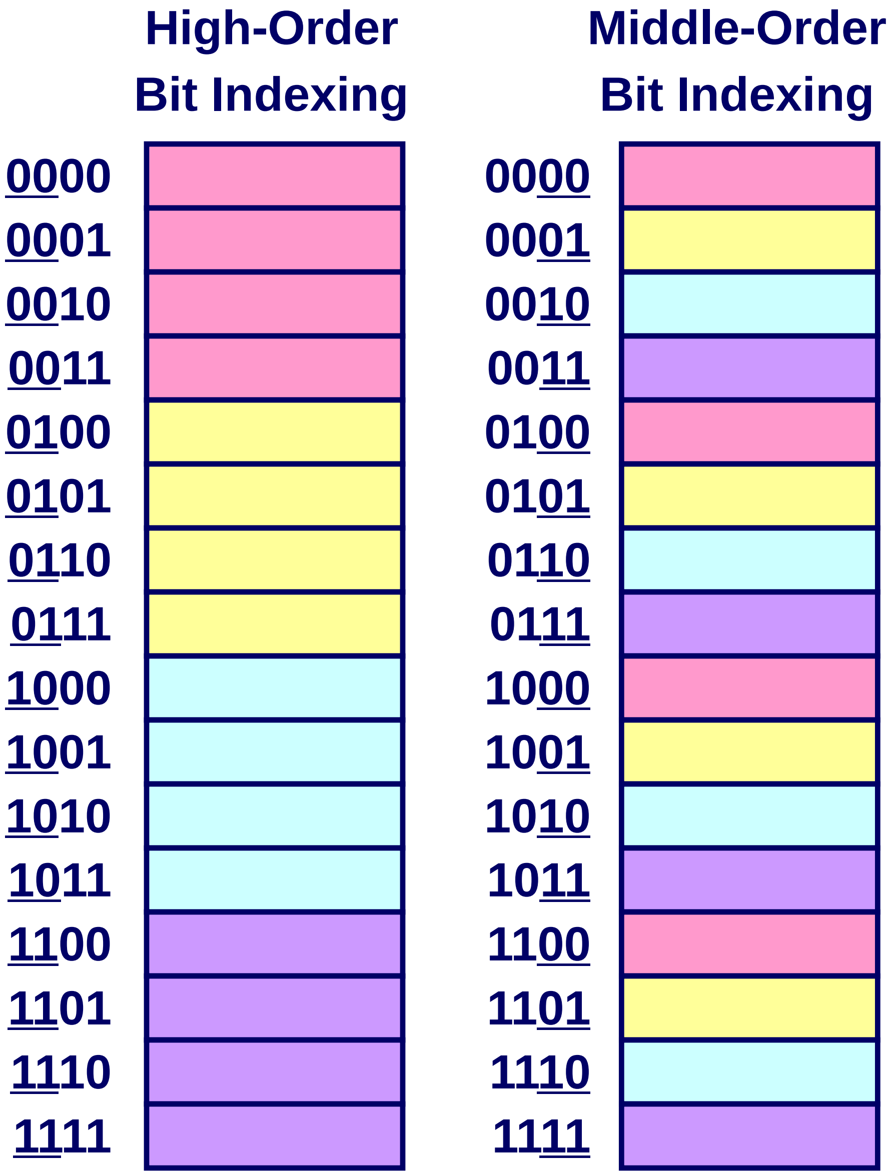

0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111

Why Use Middle Bits as Index?

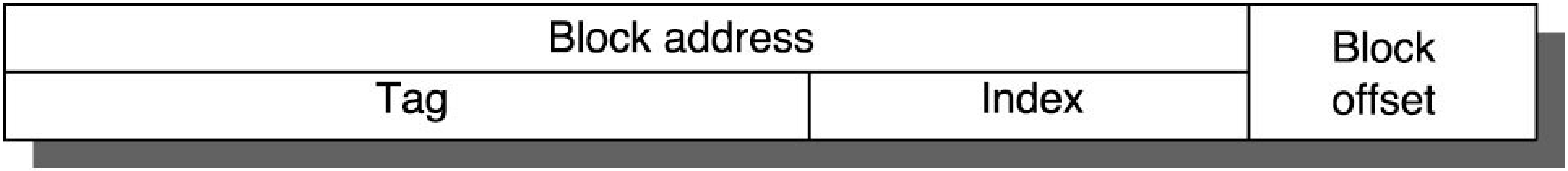


Middle-Order Bit Indexing

- ❖Consecutive memory lines map to different cache lines
- ❖Better use of spacial locality without replacement



Block Identification



Block address		Block offset
Tag	Index	

- ❖ Tag on each block
- ❖ Increasing associativity shrinks index, expands tag
- ❖ Fully Associative: No index
- ❖ Direct Mapped: Large index

johnjose@iitg.ac.in

http://www.iitg.ac.in/johnjose/