Discussion Session 9

CS/ECE 552

Ramkumar Ravi

26 Mar 2012

Mark your calendar

IMPORTANT DATES

- **04/02** – Spring break
- **04/11** – HW5 due
- **04/23** – Project Demo 2

TODAY

- HW5 discussion
- Error in mem_system_randbench.v (will be corrected and uploaded in course webpage)
- Other test-bench modules look OK
- Modification to problem 5
- Discussion of a sample state-machine and show sample code (if time permits)

Problems 3 and 4
Cache Parameters

→ *SIZE* = total amount of cache data storage, in bytes

→ *BLOCKSIZE* = total number of bytes in a single block

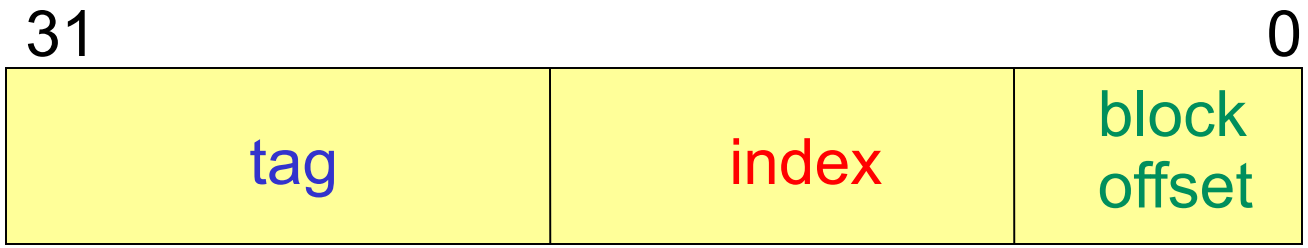
→ *ASSOC* = associativity, i.e., # of lines in a set

→ Equation for # of *cache lines* in cache = size / blocksize

→ Equation for # of *sets* in cache = size / (blocksize * associativity)

→ You may need this for few of the problems in this HW

Address Fields



Tag field is compared to the tag(s) of the indexed cache line(s).
- If it matches, block is there (hit).
- If it doesn't match, block is not there (miss).

Used to look up a "set," whose lines contain one or more memory blocks. (The # of lines per set is the "associativity".)

Once a block is found, the offset selects a particular byte or word of data in the block.

Address Fields (cont.)

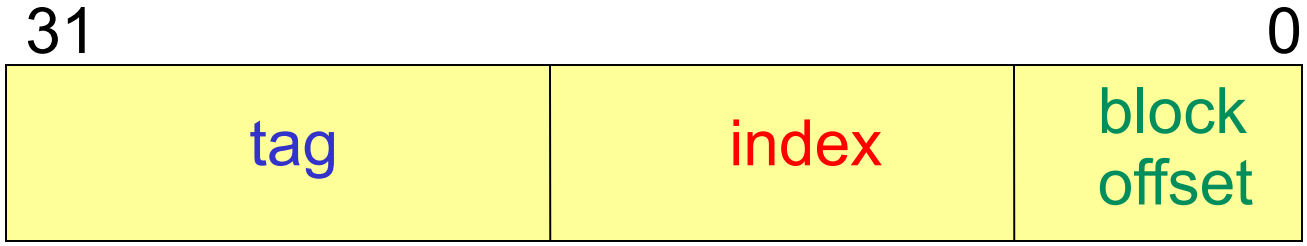
Widths of address fields (# bits)

 # index bits = $\log_2$(*# sets*)

 # block offset bits = $\log_2$(*block size*)

 # tag bits = 32 – *# index bits* – *# block offset bits*

Assuming 32-bit addresses



Example of cache operation

Example: Processor accesses a 256-byte direct-mapped cache, which has 32-byte lines, with the following sequence of addresses.

- Show contents of the cache after each access.
- Count # of hits and # of replacements.

index bits = $\log_2$(# *sets*) = $\log_2$(8) = 3

block offset bits = $\log_2$(*block size*) = $\log_2$(32 bytes) = 5

tag bits = *total # address bits* – # *index bits* – # *block offset bits* = 32 bits – 3 bits – 5 bits = 24

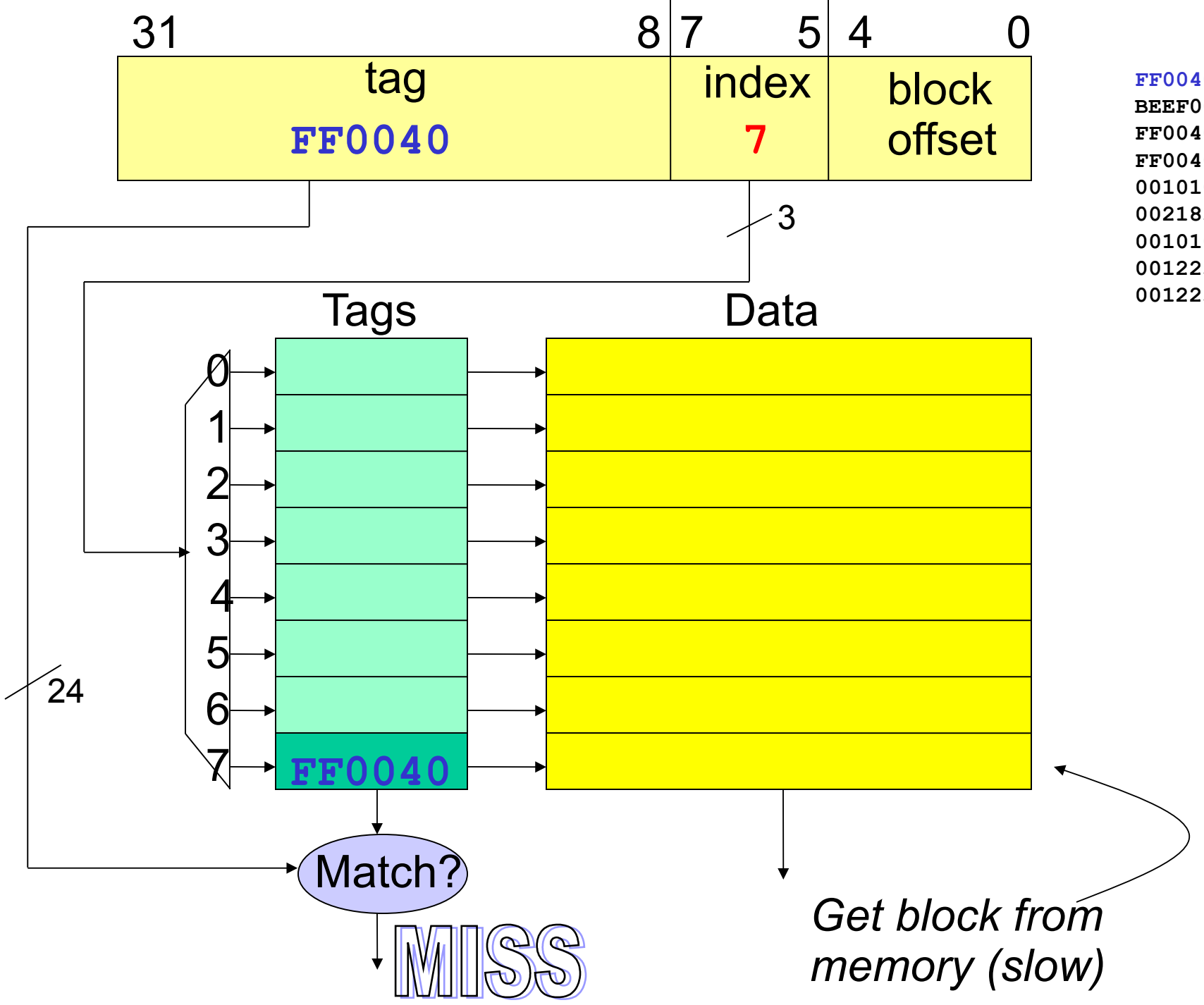
→ Thus, the top 6 nibbles (24 bits) of address form the tag and lower 2 nibbles (8 bits) of address form the index and block offset fields.

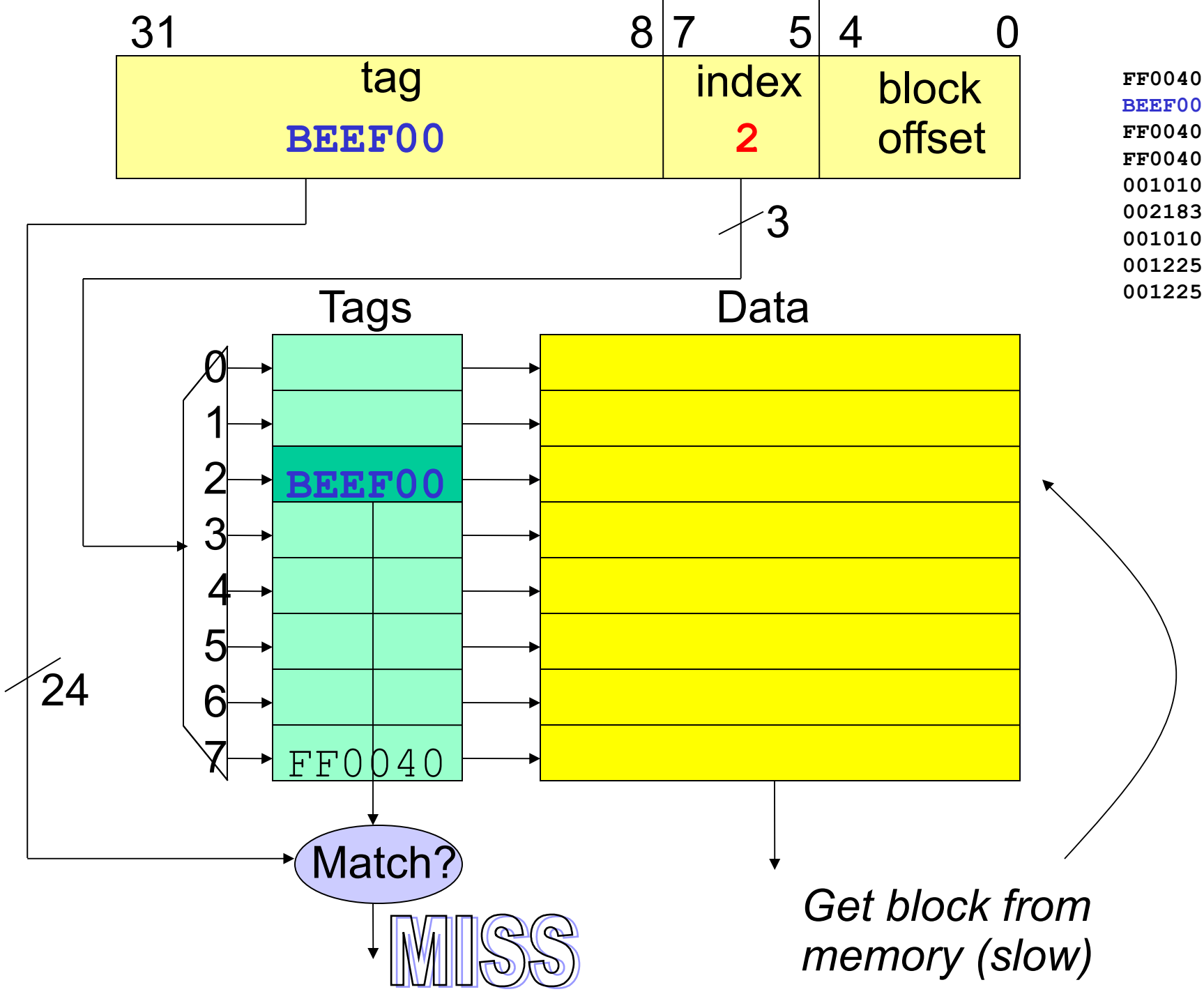
Example address sequence

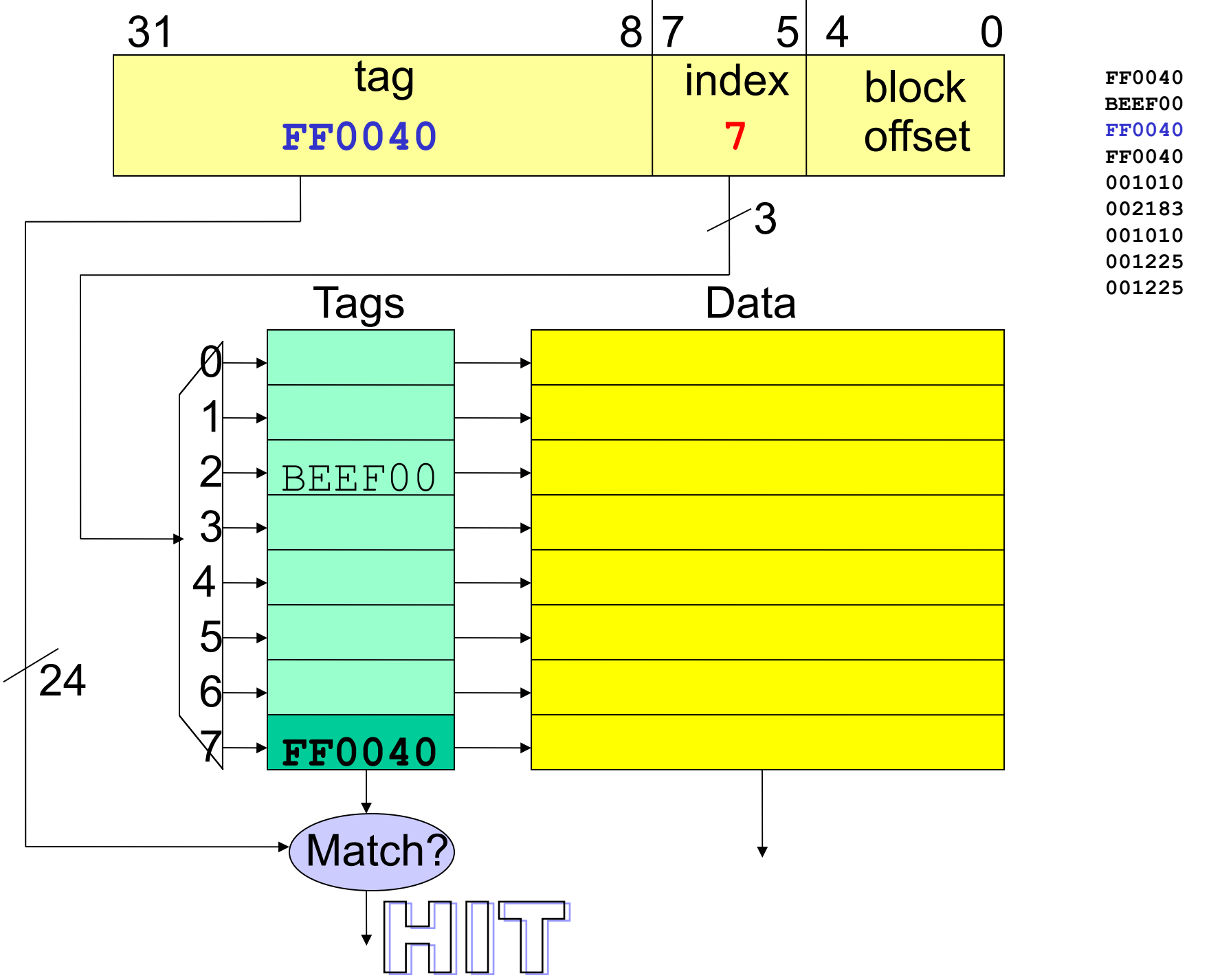
Address (hex)	Tag (hex)	Index & offset bits (binary)	Index (decimal)	Comment
0xFF0040E0				
0xBEEF005C				
0xFF0040E2				
0xFF0040E8				
0x00101078				
0x002183E0				
0x00101064				
0x0012255C				
0x00122544				

Analysis of results follows next

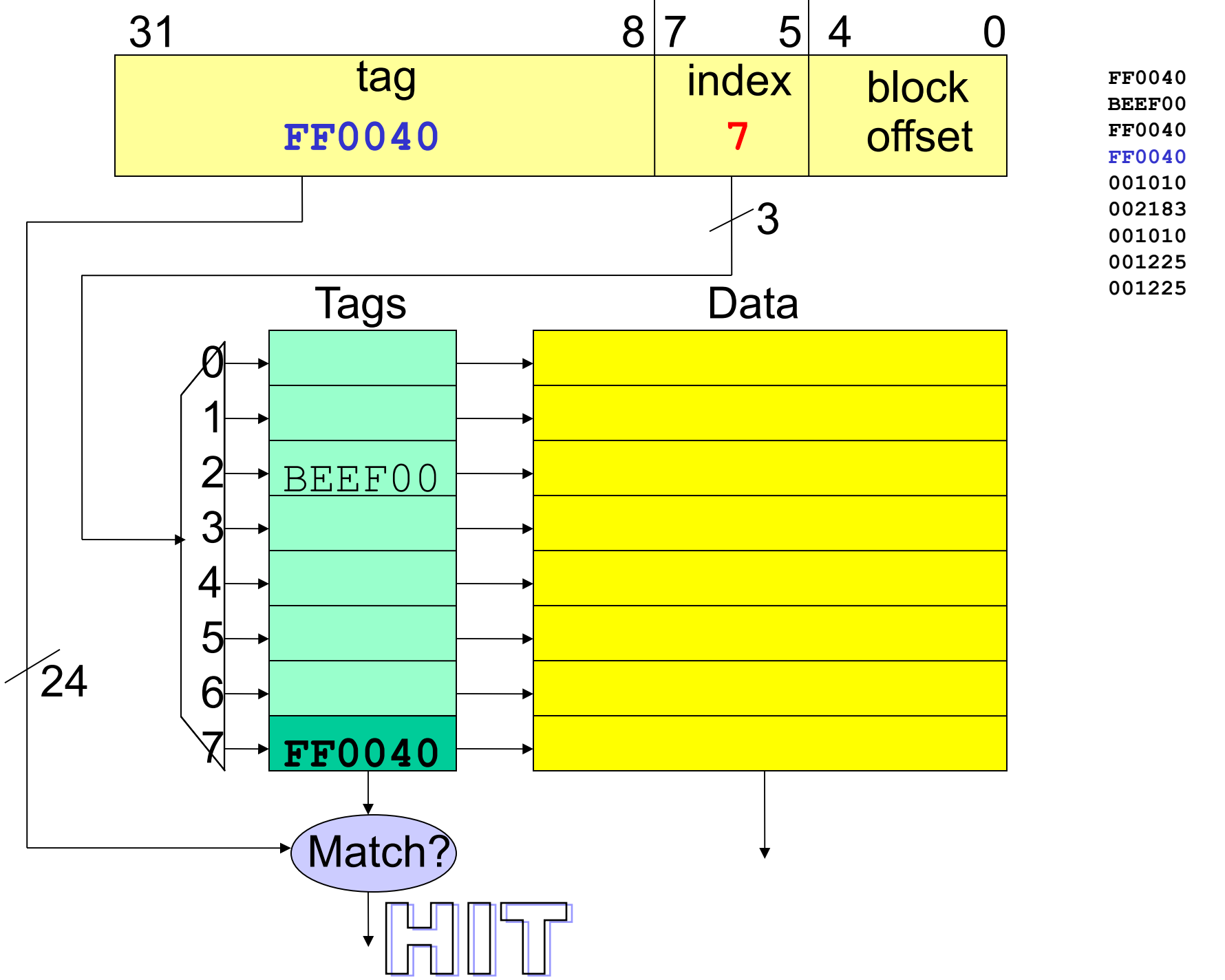
Address (hex)	Tag (hex)	Index & offset bits (binary)	Index (decimal)	Comment
0xFF0040E0	0xFF0040	1110 0000	7	Miss
0xBEEF005C	0xBEEF00	0101 1100	2	Miss
0xFF0040E2	0xFF0040	1110 0010	7	Hit
0xFF0040E8	0xFF0040	1110 1000	7	Hit
0x00101078	0x001010	0111 1000	3	Miss
0x002183E0	0x002183	1110 0000	7	Miss+Replace
0x00101064	0x001010	0110 0100	3	Hit
0x0012255C	0x001225	0101 1100	2	Miss+Replace
0x00122544	0x001225	0100 0100	2	Hit


31
8
7
5
4
0
tag
FF0040
index
7
block
offset
3
24
Tags
Data
0
1
2
3
4
5
6
7
FF0040
Match?
MISS
Get block from
memory (slow)
FF0040
BEEF00
FF0040
FF0040
001010
002183
001010
001225
001225

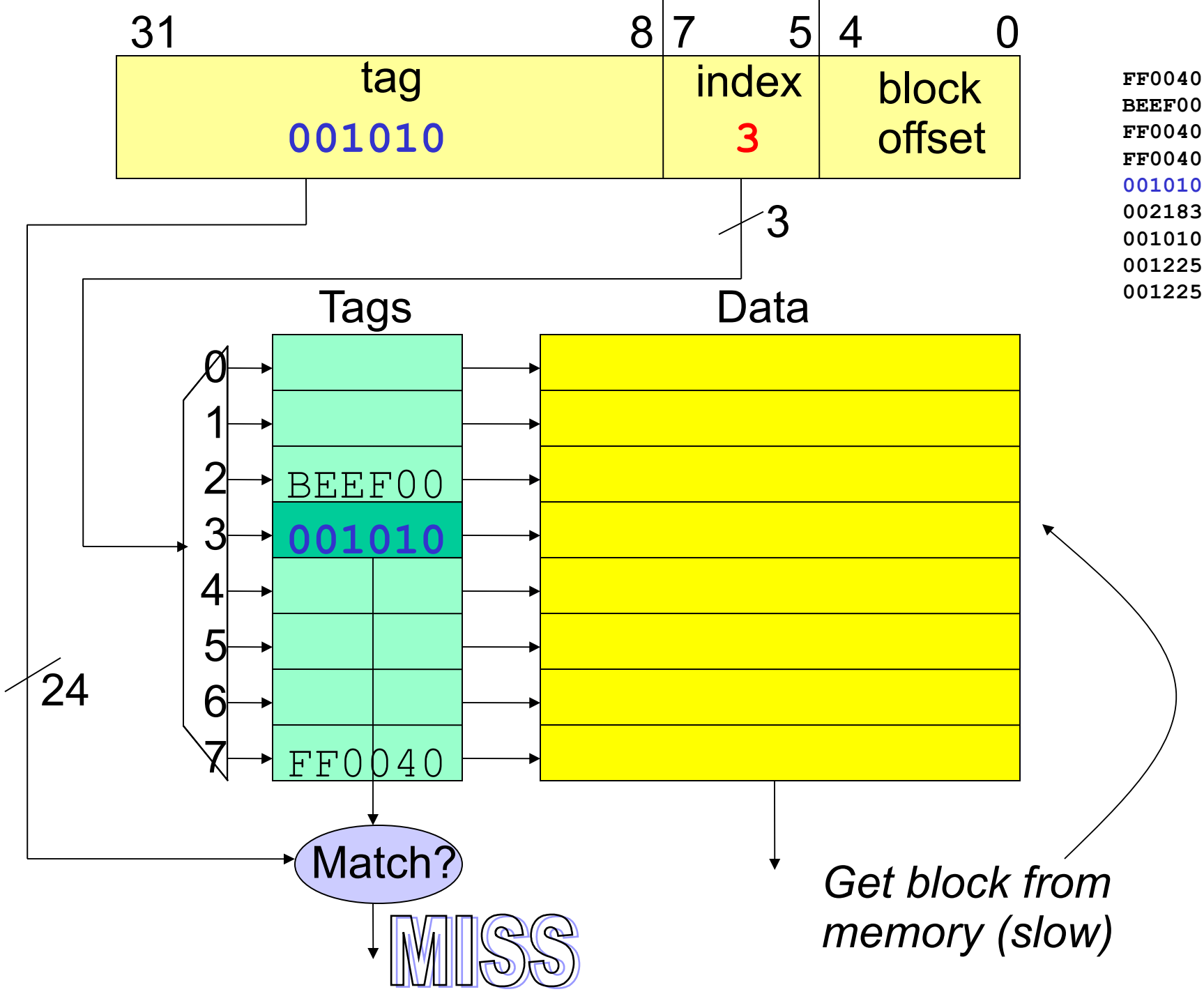

31
8
7
5
4
0
tag
BEEF00
index
2
block
offset
3
Tags
Data
0
1
2
3
4
5
6
7
BEEF00
FF0040
24
Match?
MISS
Get block from
memory (slow)
FF0040
BEEF00
FF0040
FF0040
001010
002183
001010
001225
001225

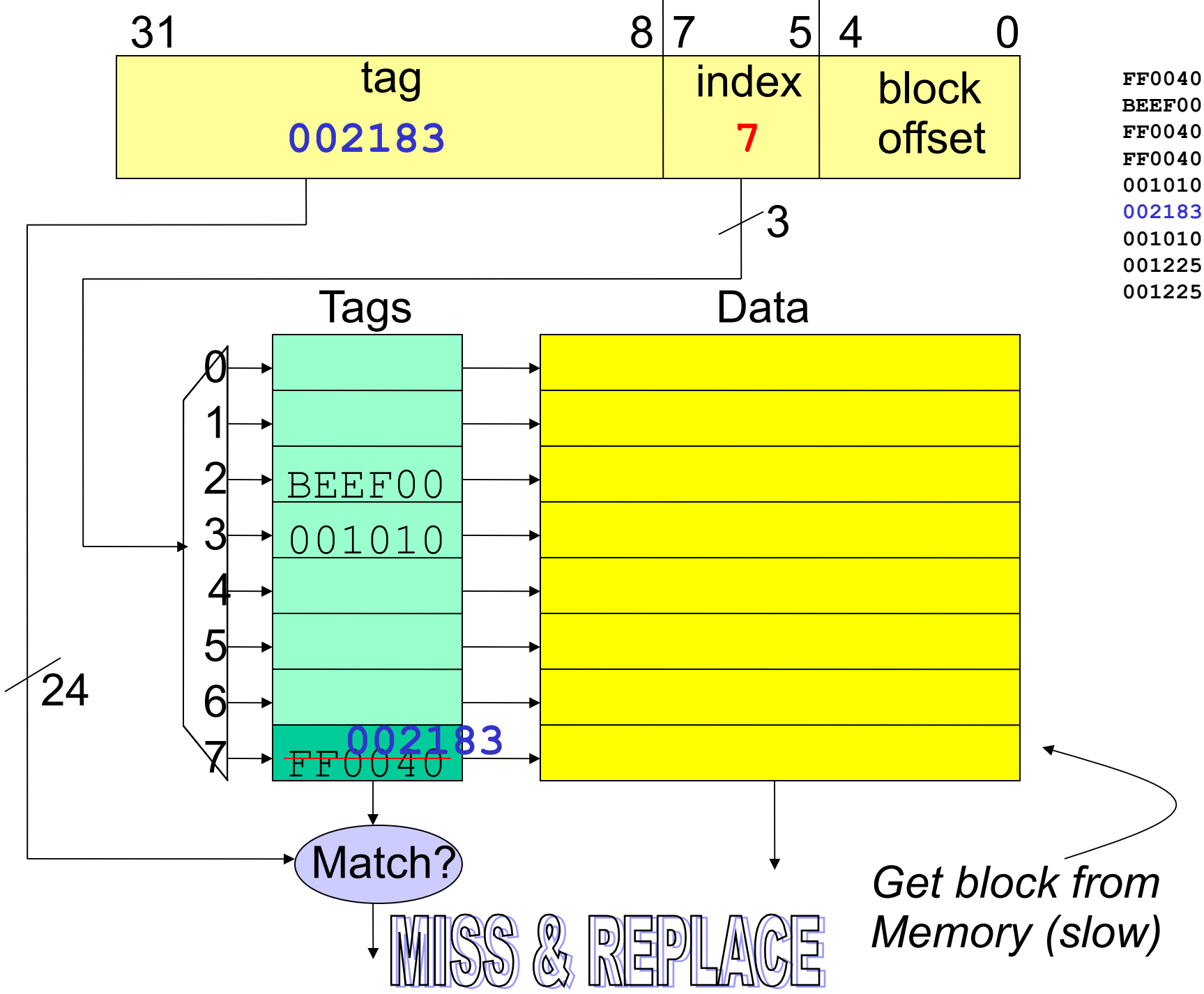


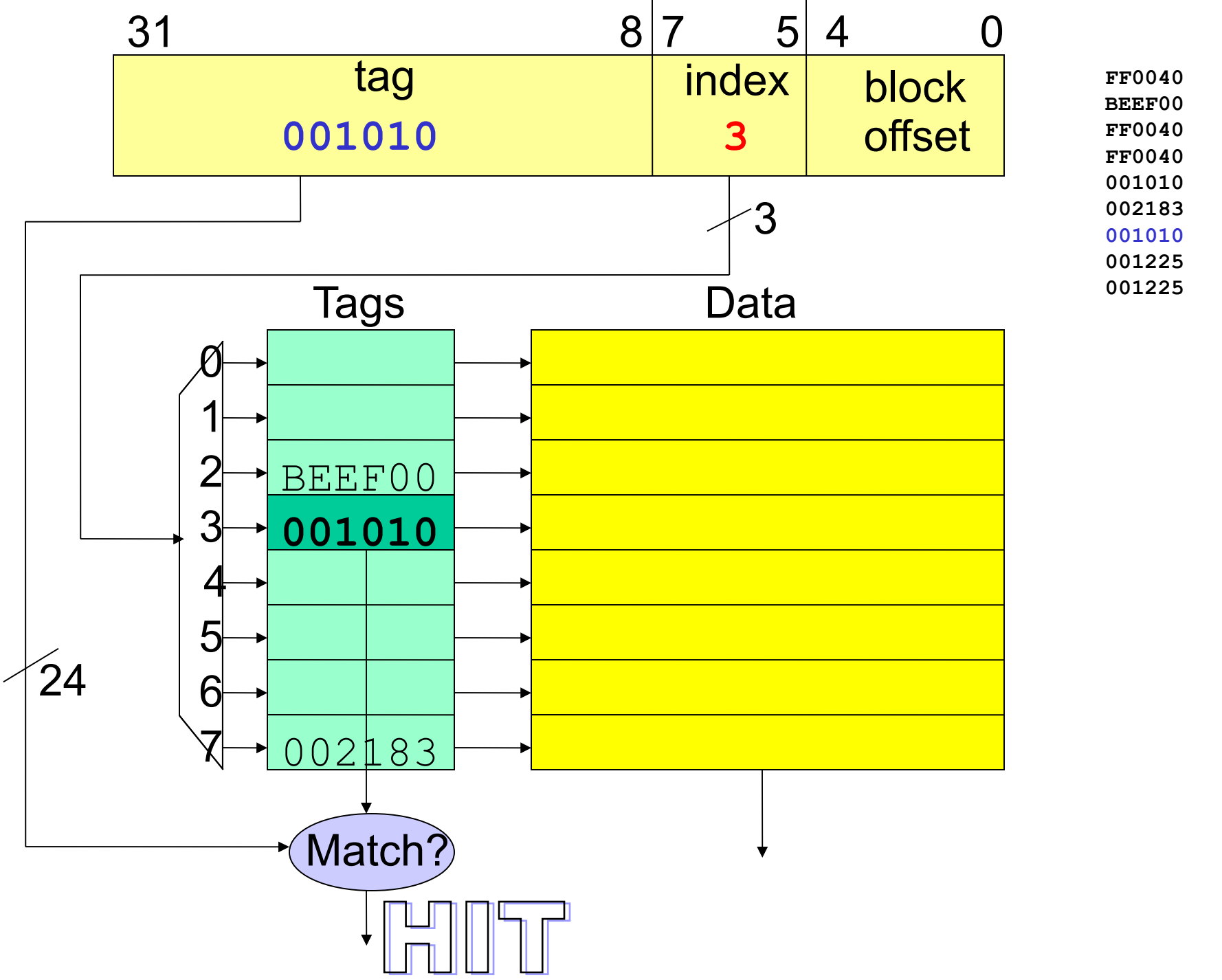
FF0040
BEEF00
FF0040
FF0040
001010
002183
001010
001225
001225

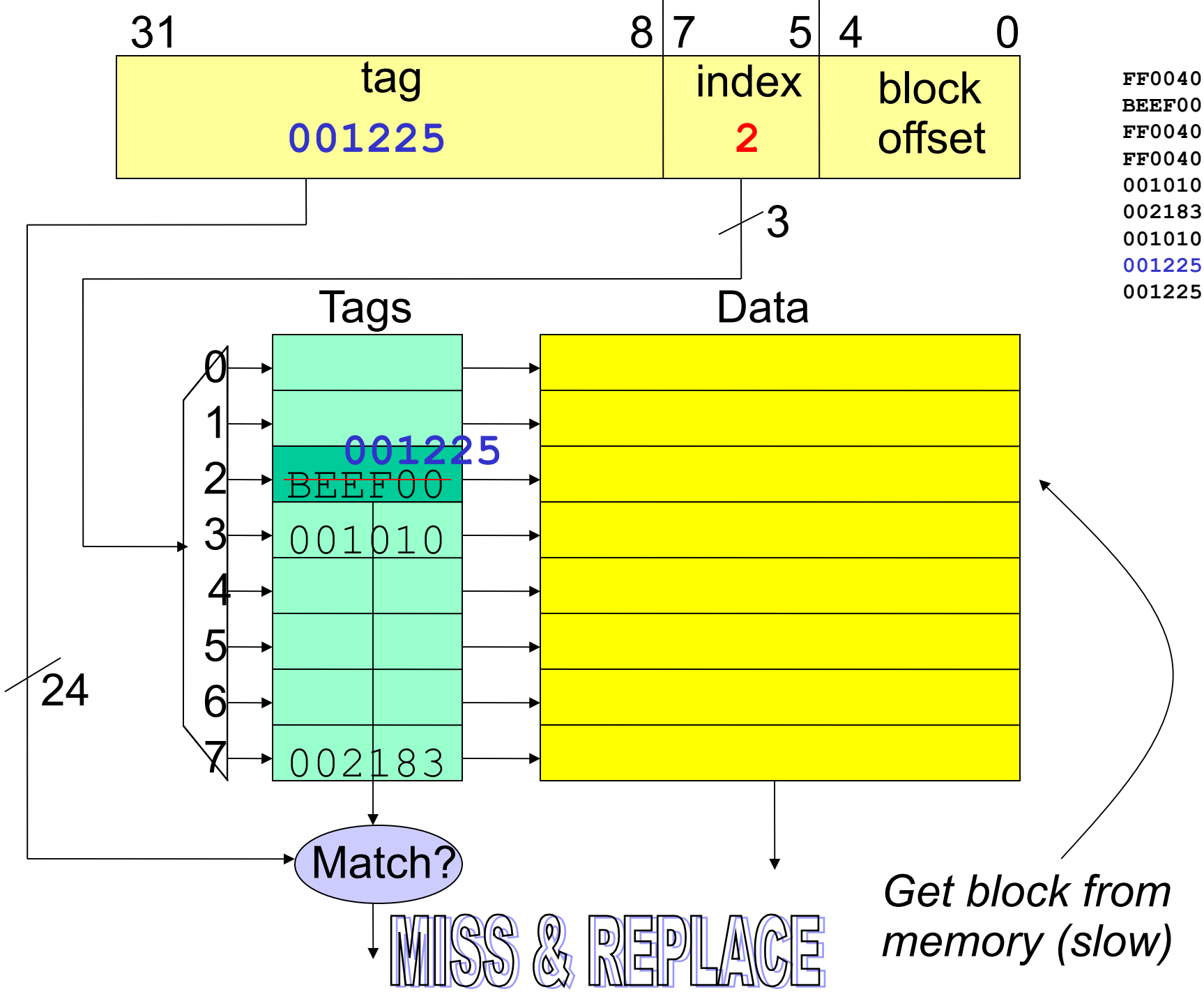


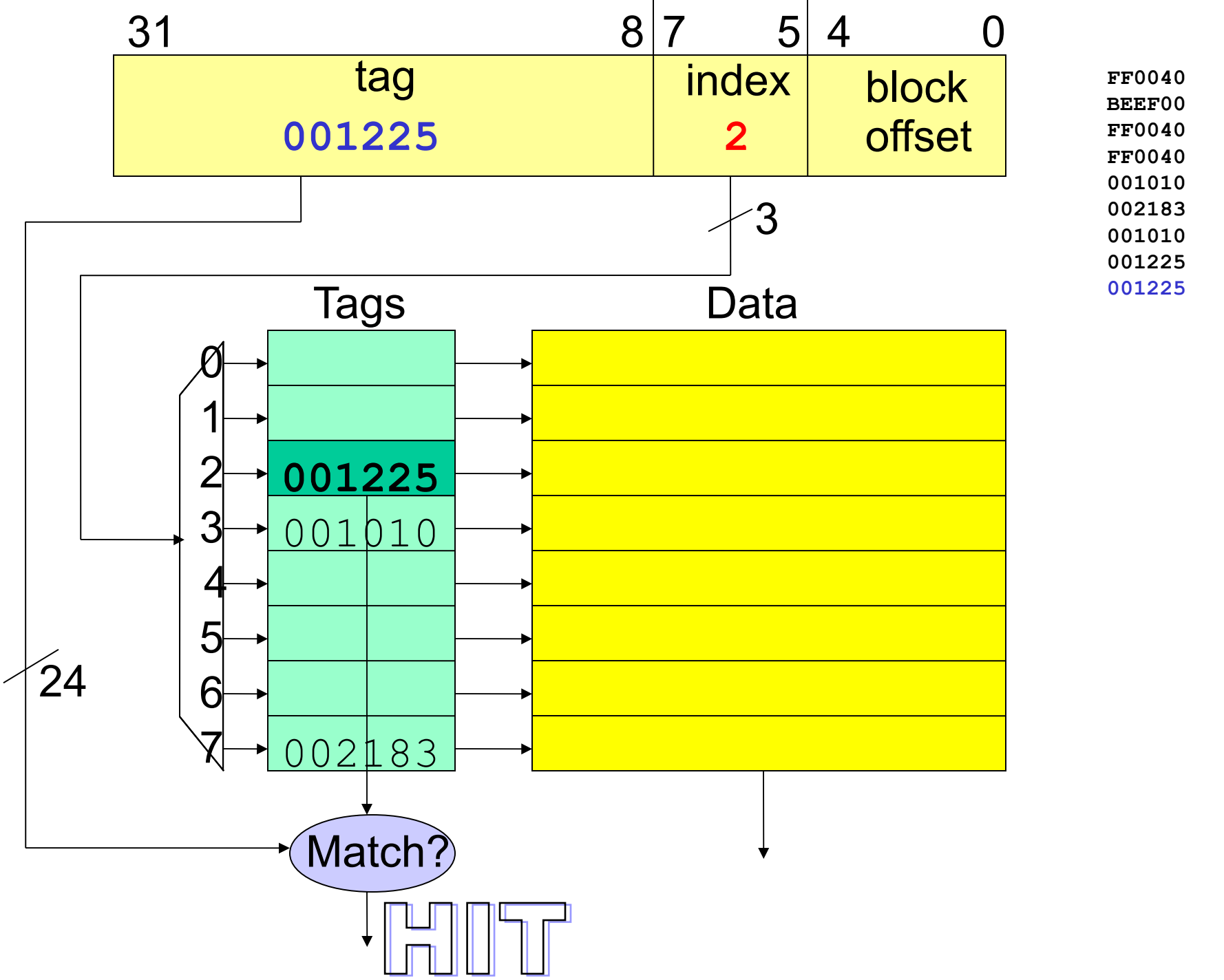
FF0040
BEEF00
FF0040
FF0040
001010
002183
001010
001225
001225


31
8
7
5
4
0
tag
001010
index
3
block
offset
FF0040
BEEF00
FF0040
FF0040
001010
002183
001010
001225
001225
3
Tags
Data
0
1
2
3
4
5
6
7
BEEF00
001010
FF0040
24
Match?
MISS
Get block from
memory (slow)


31
8
7
5
4
0
tag
002183
index
7
block
offset
FF0040
BEEF00
FF0040
FF0040
001010
002183
001010
001225
001225
3
Tags
Data
0
1
2
3
4
5
6
7
BEEF00
001010
002183
FF0040
24
Match?
MISS & REPLACE
Get block from
Memory (slow)


31
8
7
5
4
0
tag
001010
index
3
block
offset
3
24
Tags
Data
0
1
2
3
4
5
6
7
BEEF00
001010
002183
Match?
HIT
FF0040
BEEF00
FF0040
FF0040
001010
002183
001010
001225
001225


31
8
7
5
4
0
tag
001225
index
2
block
offset
FF0040
BEEF00
FF0040
FF0040
001010
002183
001010
001225
001225
3
Tags
Data
0
1
2
3
4
5
6
7
001225
BEEF00
001010
002183
24
Match?
MISS & REPLACE
Get block from
memory (slow)


31
8
7
5
4
0
tag
001225
index
2
block
offset
3
24
Tags
Data
0
1
2
3
4
5
6
7
001225
001010
002183
Match?
HIT
FF0040
BEEF00
FF0040
FF0040
001010
002183
001010
001225
001225

Set-Associative Example

Example: Processor accesses a 256-byte 2-way set-associative cache, which has line size of 32 bytes, with the following sequence of addresses.

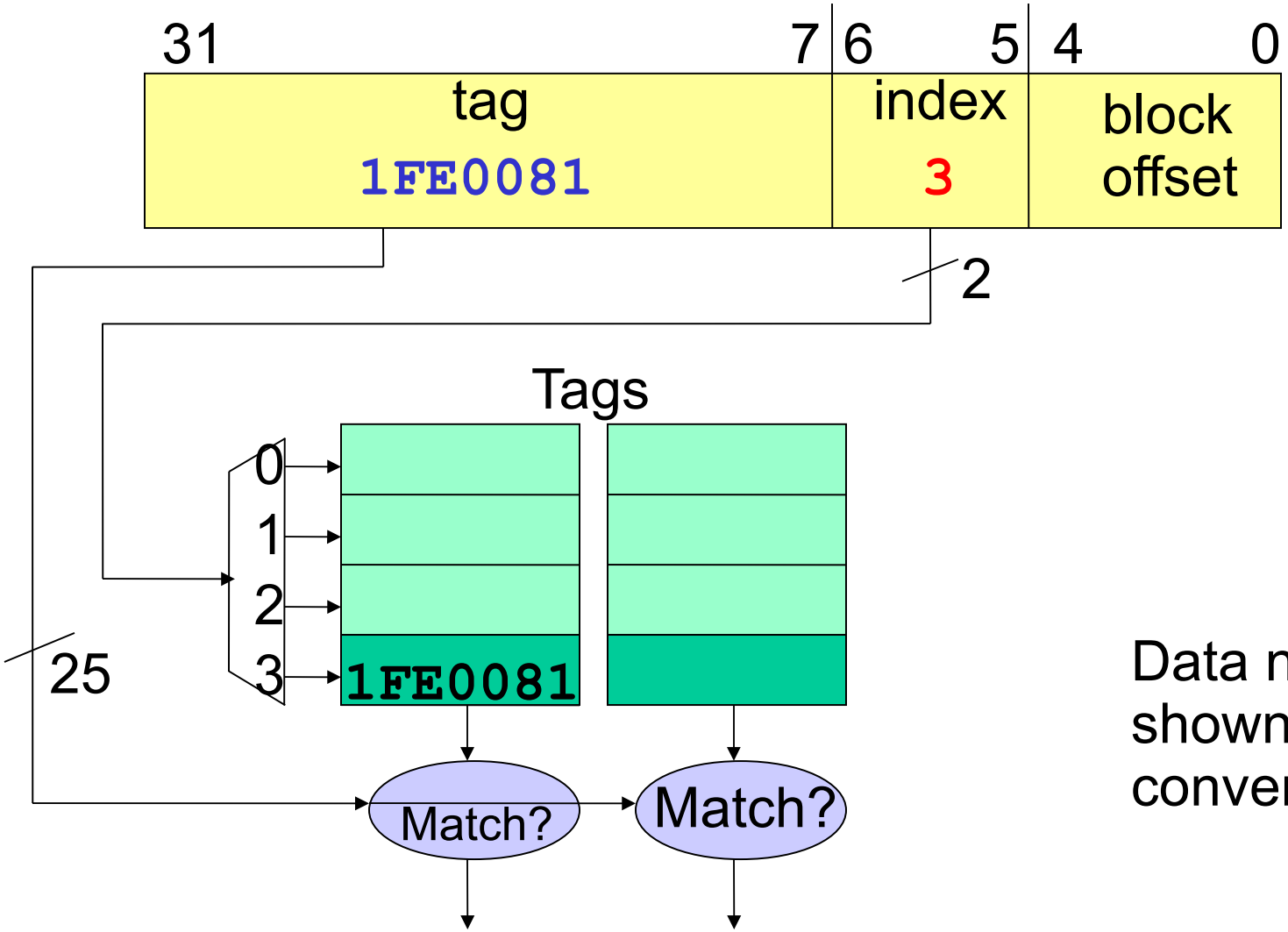
- Show contents of cache after each access.
- Count # of hits and # of replacements.

index bits = $\log_2$(# *sets*) = $\log_2$(4) = 2

block offset bits = $\log_2$(*block size*) = $\log_2$(32 bytes) = 5

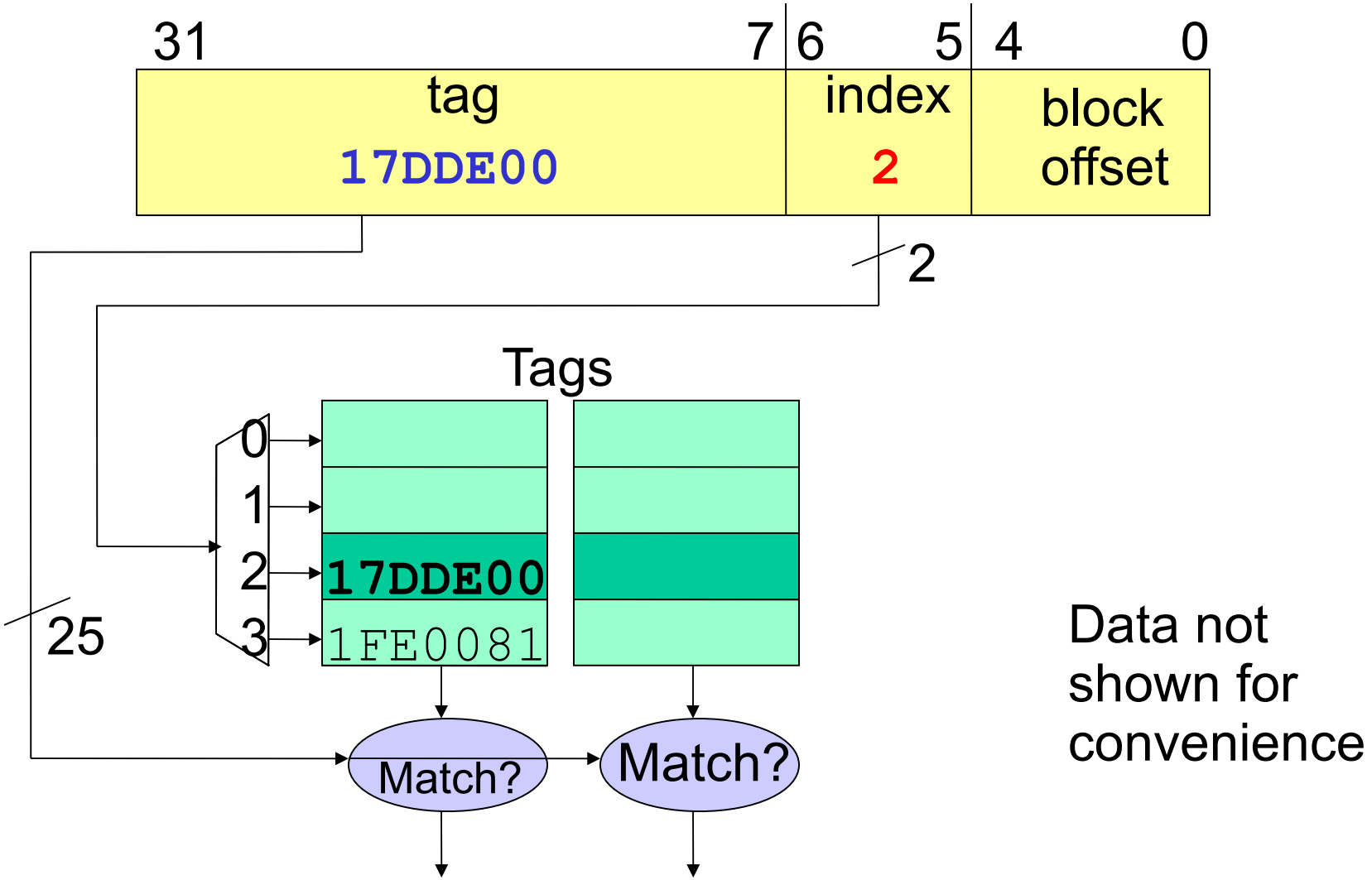
tag bits = *total # address bits* – # *index bits* – # *block offset bits* = 32 bits – 2 bits – 5 bits = 25

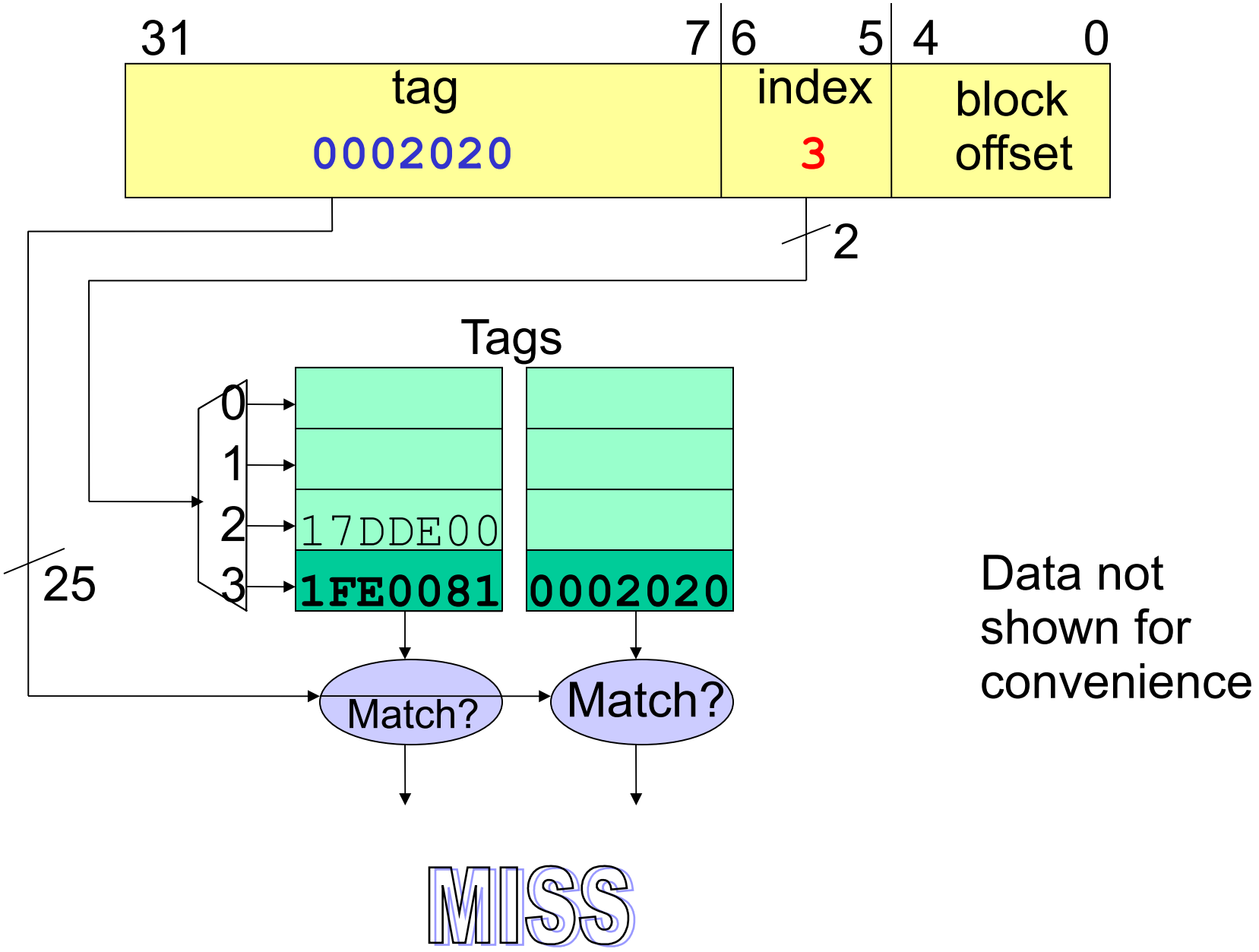
Address (hex)	Tag (hex)	Index & offset bits (binary)	Index (decimal)	Comment
0xFF0040E0	0x1FE0081	110 0000	3	Miss
0xBEEF005C	0x17DDE00	101 1100	2	Miss
0x00101078	0x0002020	111 1000	3	Miss
0xFF0040E2	0x1FE0081	110 0010	3	Hit
0x00101078	0x0002020	111 1000	3	Hit
0x002183E0	0x0004307	110 0000	3	Miss/Replace
0x00101064	0x0002020	110 0100	3	Hit



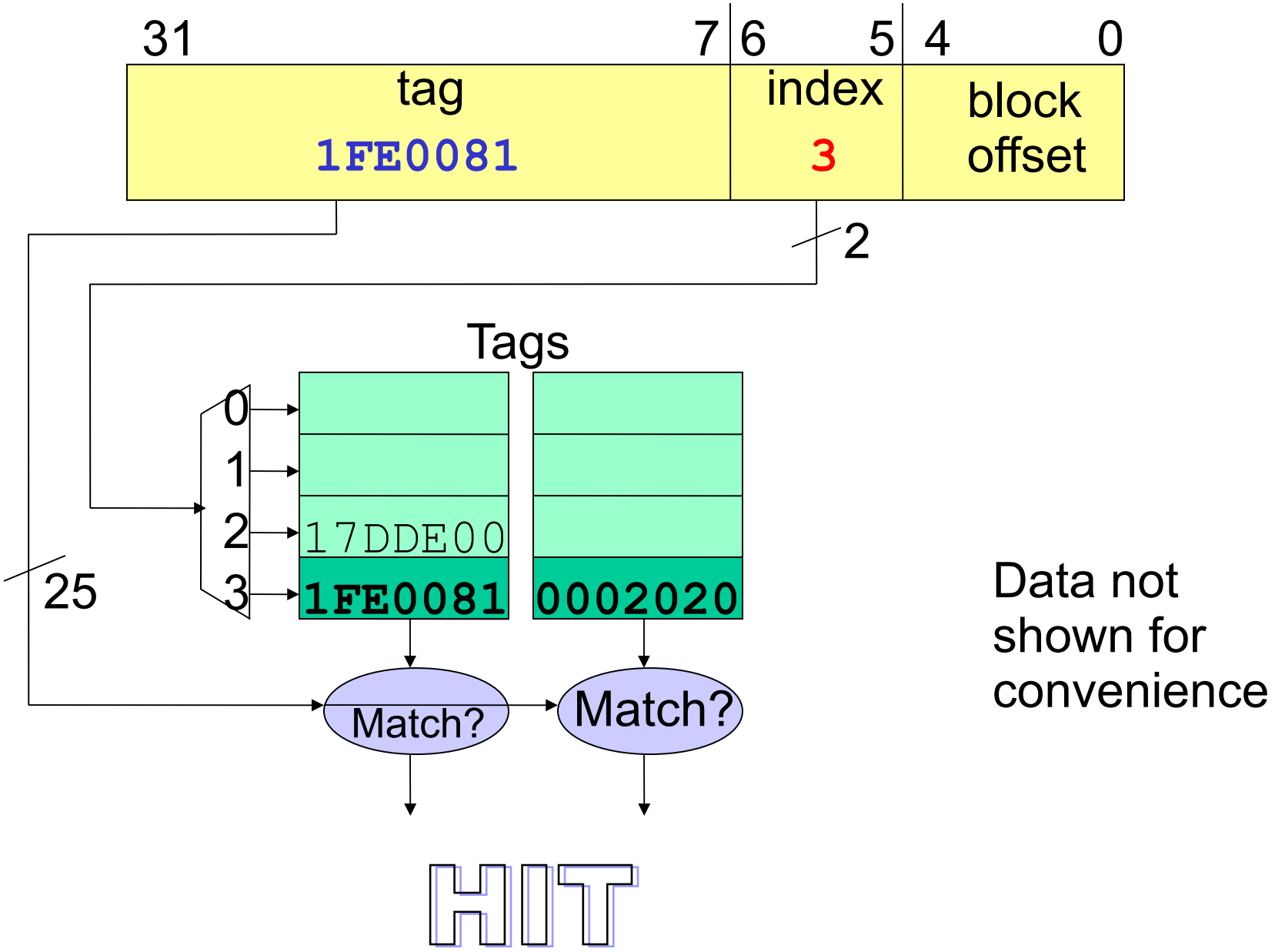
Data not shown for convenience

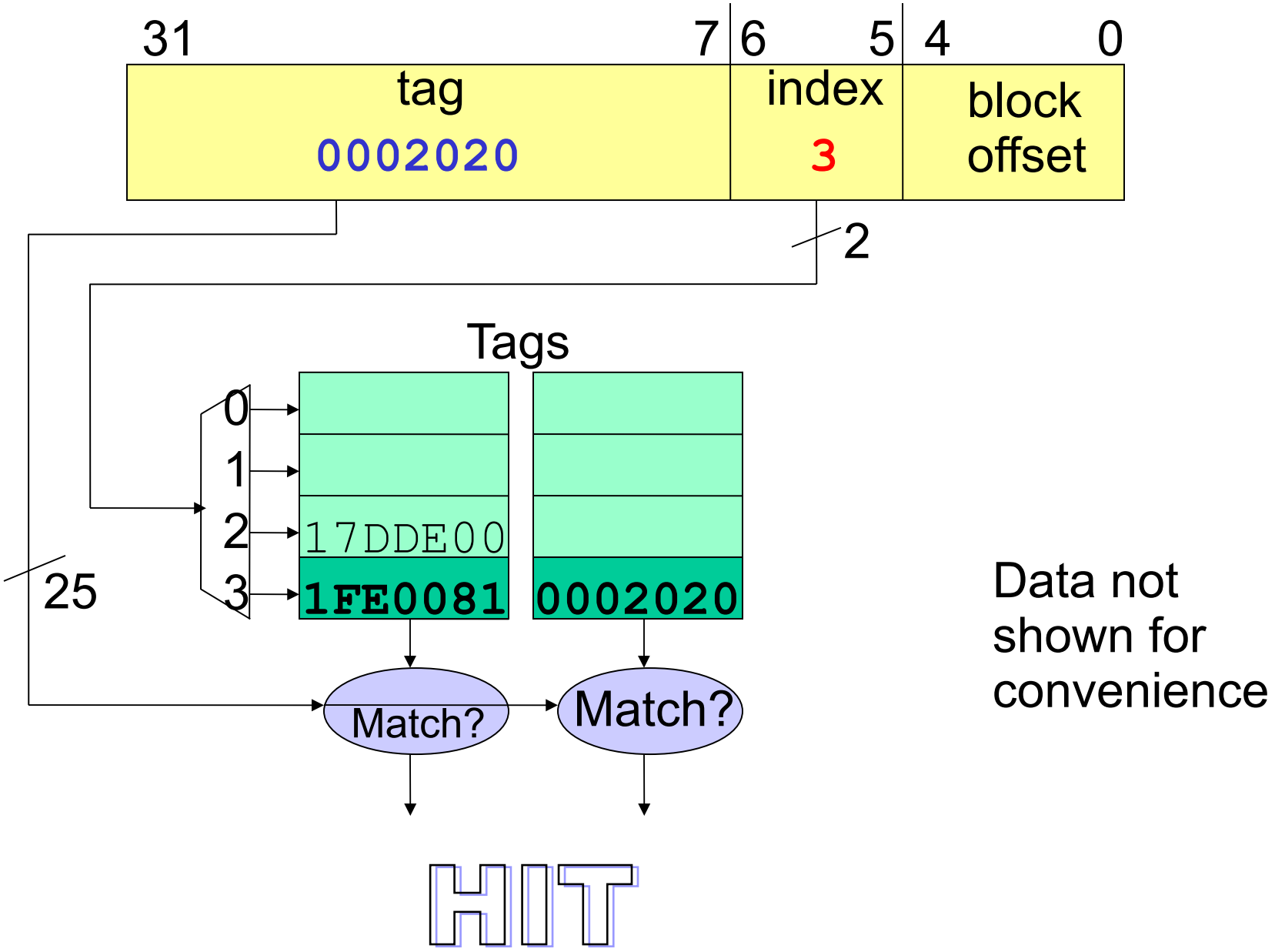
MISS


31
7
6
5
4
0
tag
17DDE00
index
2
block
offset
2
Tags
0
1
2
3
17DDE00
1FE0081
25
Match?
Match?
MISS
Data not
shown for
convenience

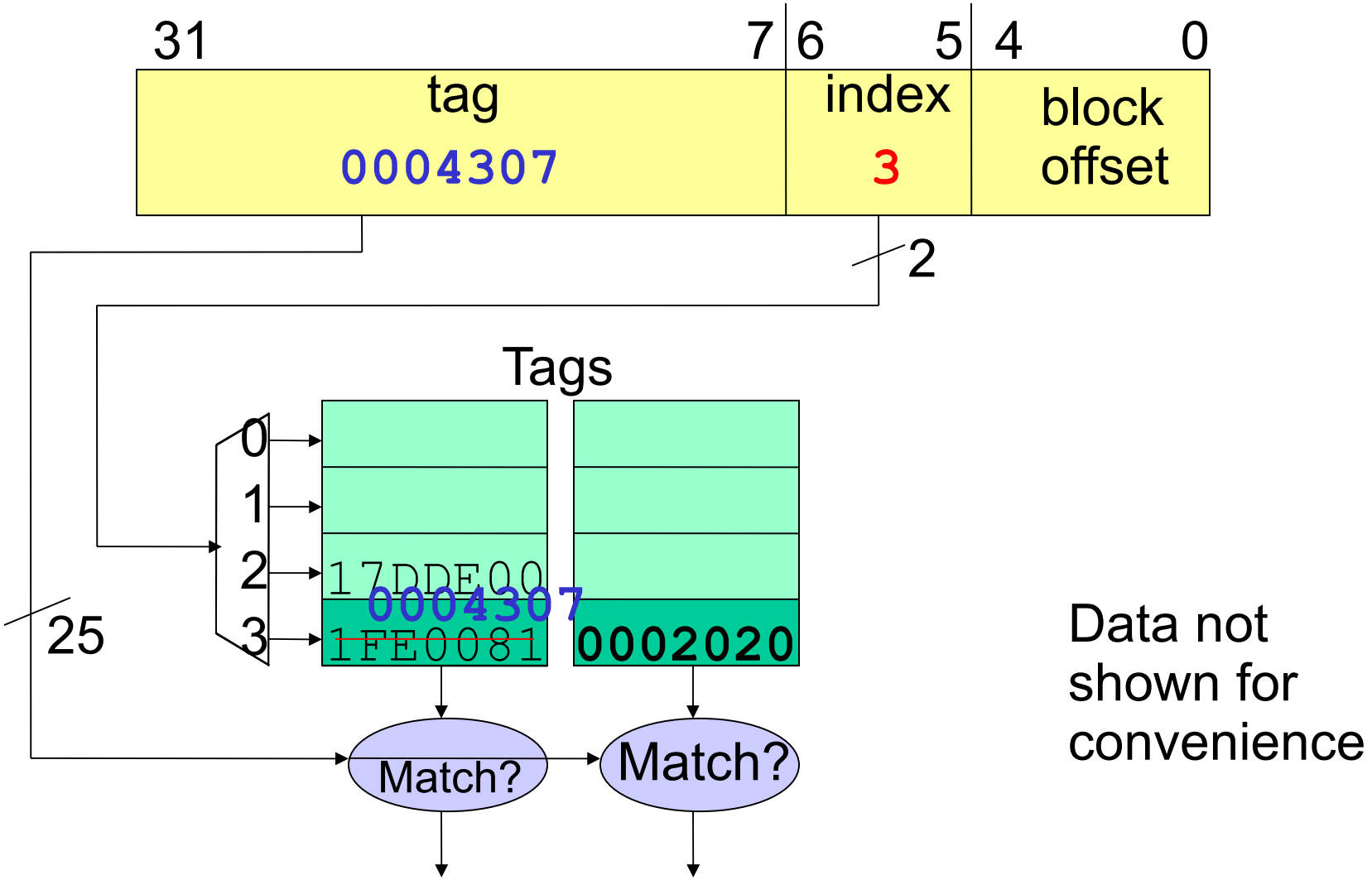


Data not shown for convenience


31
7
6
5
4
0
tag
1FE0081
index
3
block
offset
2
Tags
0
1
2
3
17DDE00
1FE0081
0002020
25
Match?
Match?
HIT
Data not
shown for
convenience

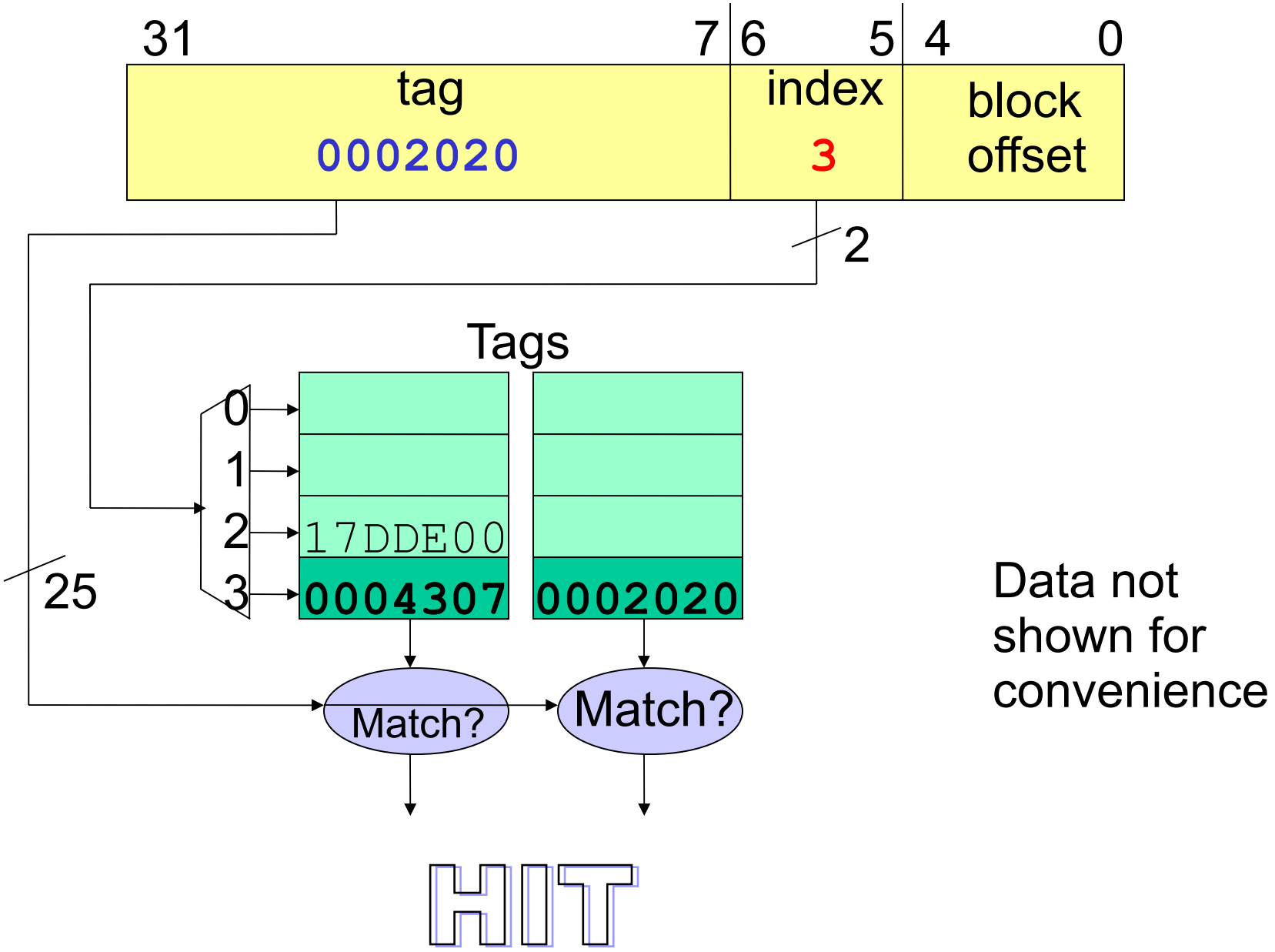


Data not shown for convenience



Data not shown for convenience

MISS & REPLACE LRU BLOCK



Data not shown for convenience

PROBLEM 5

→ Total cache storage (capacity)

- (64-byte blocks) * (512 sets) * (# of ways)

→ Total number of bits

- Tag size = 41 bits - # index bits - # offset bits
- Block size = valid + dirty + LRU + tag + data

PROBLEMS 1 and 2

- Sample FSM presentation
- Understanding the requirements of the problem