

Caches III

CSE 351 Spring 2021

Instructor:

Ruth Anderson

Teaching Assistants:

Allen Aby

Joy Dang

Alena Dickmann

Catherine Guevara

Corinne Herzog

Ian Hsiao

Diya Joy

Jim Limprasert

Armin Magness

Aman Mohammed

Monty Nitschke

Allie Pflieger

Neil Ryan

Alex Saveau

Sanjana Sridhar

Amy Xu



<https://what-if.xkcd.com/111/>

Administrivia

- ❖ Unit Summary #2 due Friday (5/07)
 - Task 1 & 2 identical for all unit summaries
 - Task 3 posted
- ❖ hw16 due ~~Friday (5/07)~~ Monday (5/10)
- ❖ hw17 due *next* Wednesday (5/12)
 - Don't wait too long, this is a BIG hw
- ❖ Lab 3 due Wednesday (5/12)

- ❖ **Questions Docs:** Use @uw google account to access!!
 - <https://tinyurl.com/CSE351-21sp-Questions>

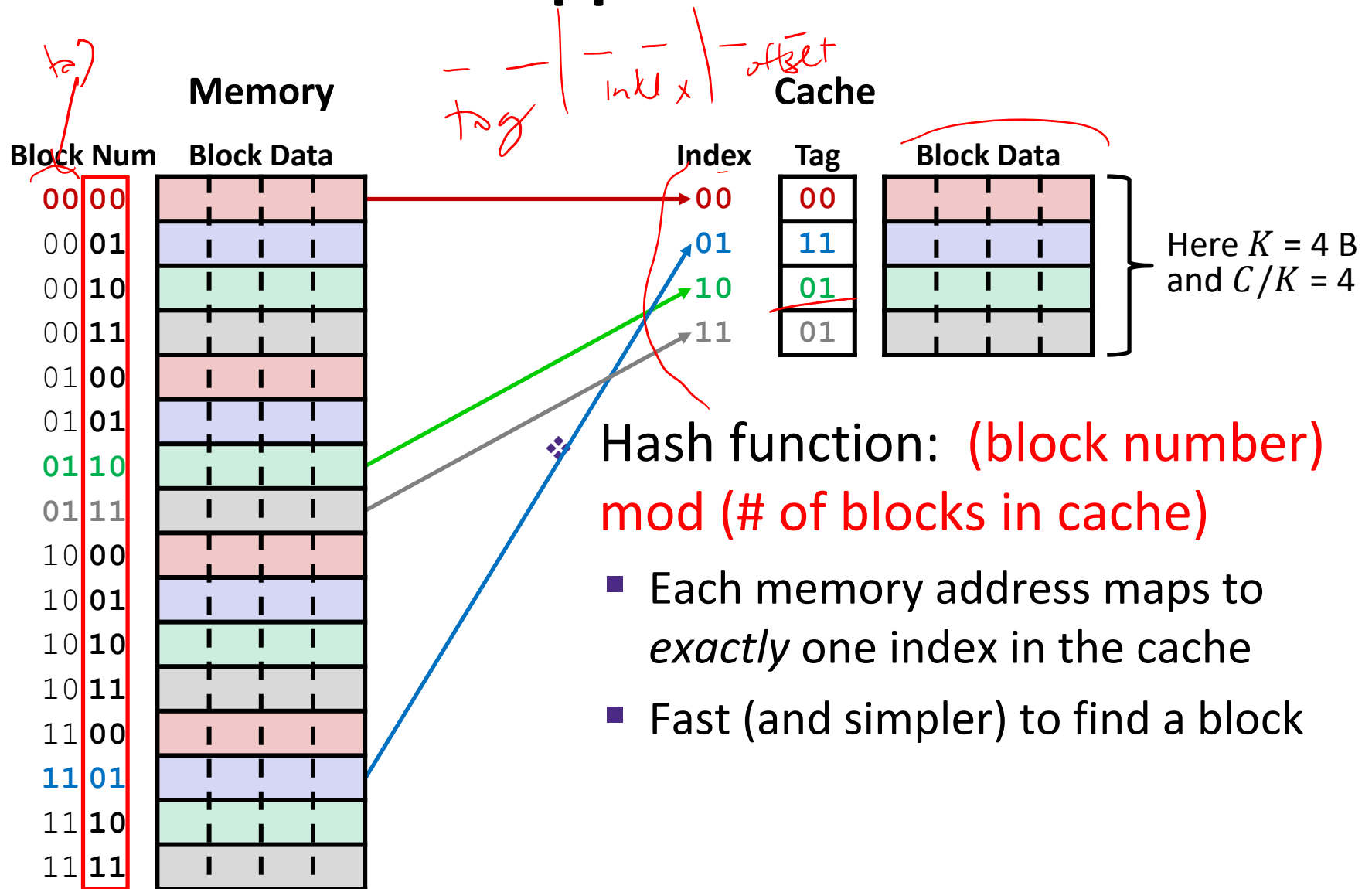
Making memory accesses fast!

- ❖ Cache basics
- ❖ Principle of locality
- ❖ Memory hierarchies
- ❖ Cache organization
 - Direct-mapped (*sets*; index + tag)
 - **Associativity (*ways*)**
 - **Replacement policy**
 - Handling writes
- ❖ Program optimizations that consider caches

Reading Review

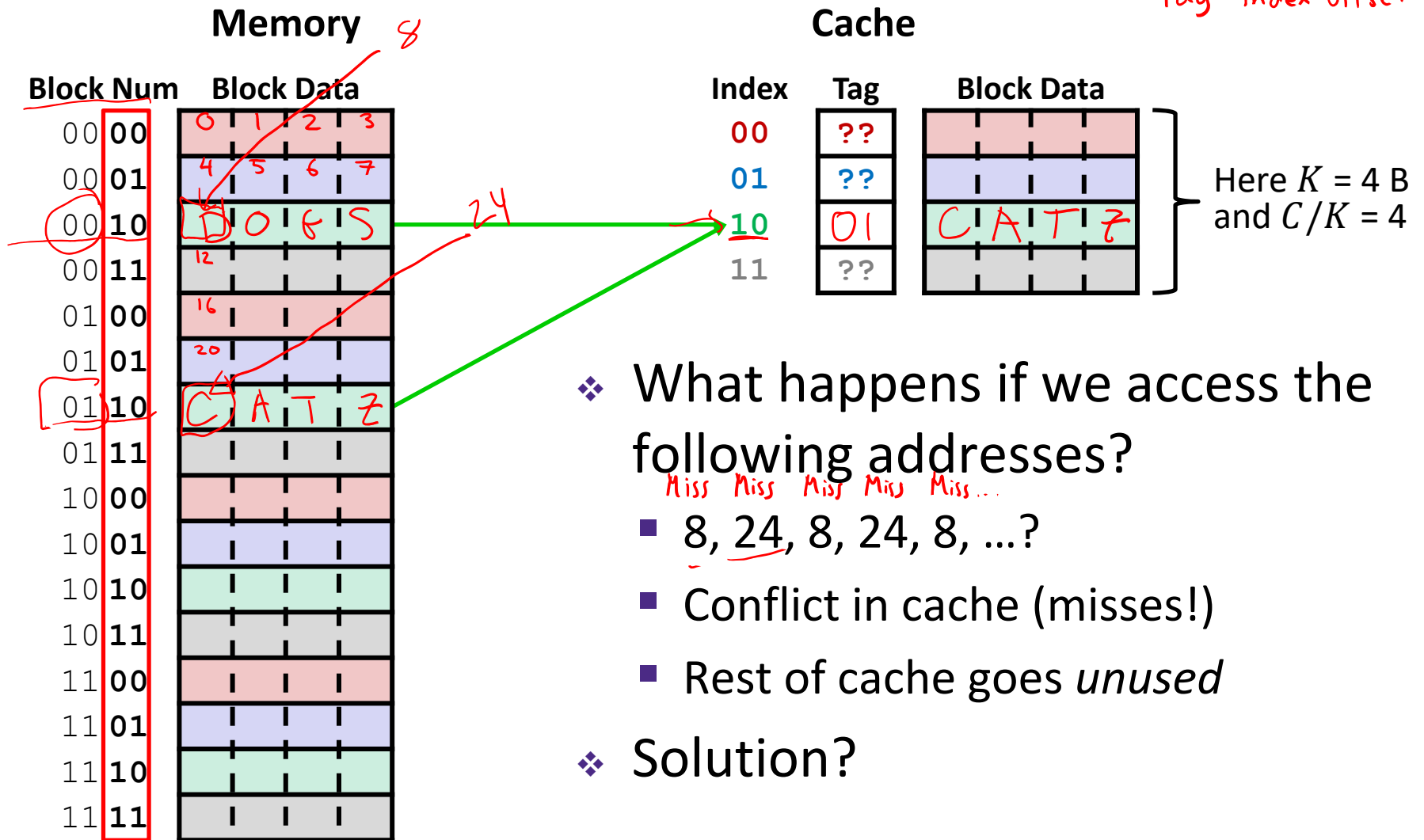
- ❖ Terminology:
 - Associativity: sets, fully-associative cache
 - Replacement policies: least recently used (LRU)
 - Cache line: cache block + management bits (valid, tag)
 - Cache misses: compulsory, conflict, capacity

Review: Direct-Mapped Cache



Direct-Mapped Cache Problem

8: 0b 00|10|00
 24: 0b 01|10|00
 tag | index | offset



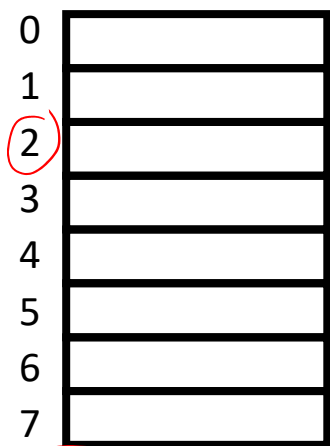
- ❖ What happens if we access the following addresses?
Miss Miss Miss Miss Miss...
 - 8, 24, 8, 24, 8, ...?
 - Conflict in cache (misses!)
 - Rest of cache goes *unused*
- ❖ Solution?

Associativity

- ❖ What if we could store data in any place in the cache?
 - More complicated hardware = more power consumed, slower
- ❖ So we *combine* the two ideas:
 - Each address maps to exactly one **set**
 - Each set can store block in more than one **way**

$E=1$
1-way:
 8 sets,

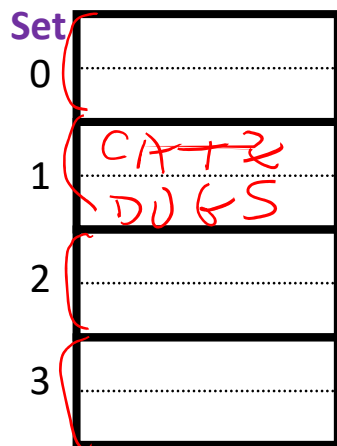
1 block each



direct-mapped

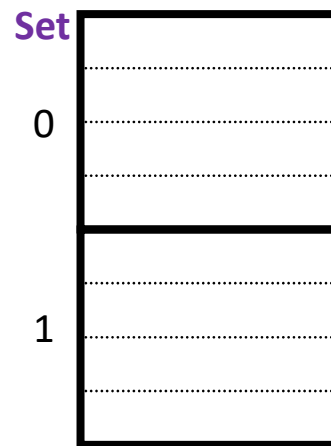
$E=2$
2-way:
 4 sets,

2 blocks each



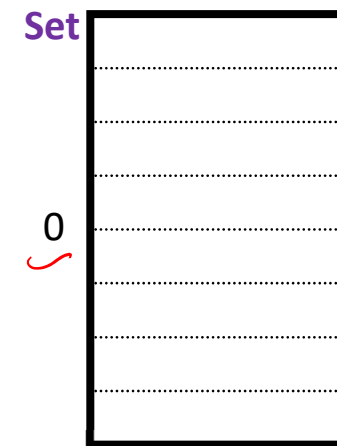
$E=4$
4-way:

2 sets,
 4 blocks each



$E=8$
8-way:

1 set,
 8 blocks

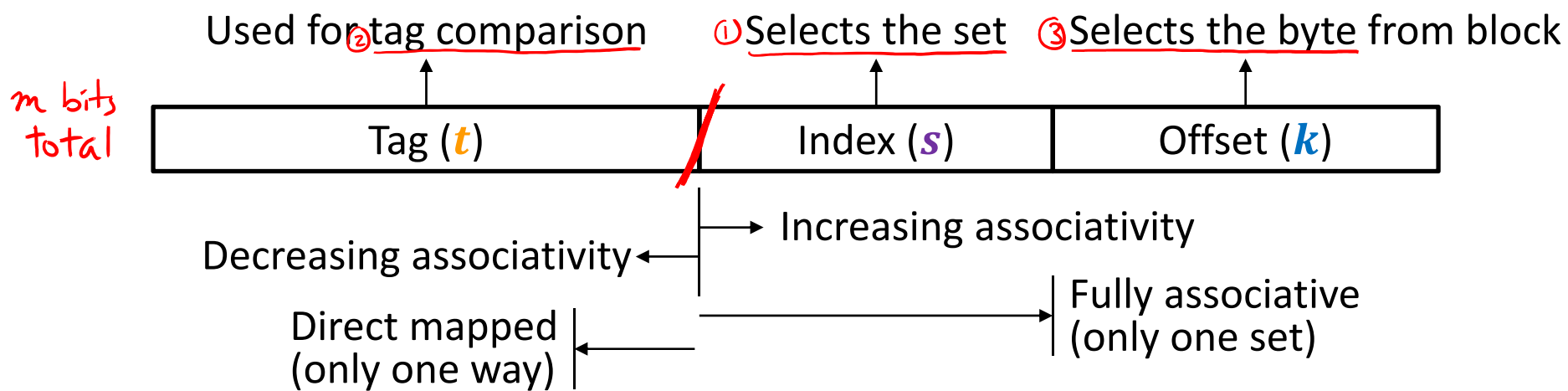


fully associative

Cache Organization (3)

Note: The textbook uses “b” for offset bits

- ❖ **Associativity (E):** # of ways for each set
 - Such a cache is called an “ E -way set associative cache”
 - We now index into cache sets, of which there are $S = C/K/E$
 - Use lowest $\log_2(C/K/E) = s$ bits of block address
 - Direct-mapped: $E = 1$, so $s = \log_2(C/K)$ as we saw previously
 - Fully associative: $E = C/K$, so $s = 0$ bits

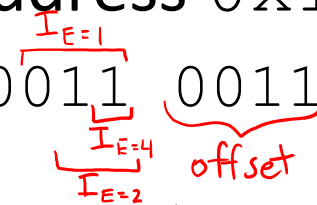


Example Placement

block size:	16 B
capacity:	8 blocks
address:	16 bits

❖ Where would data from address 0x1833 be placed?

■ Binary: 0b 0001 1000 0011 0011



$$t = m - s - k \quad s = \log_2(C/K/E) \quad k = \log_2(K)$$

m-bit address:

Tag (<i>t</i>)	Index (<i>s</i>)	Offset (<i>k</i>)
------------------	--------------------	---------------------

E = 1
s = ?

Direct-mapped

Set	Tag	Data
(000)	0	
(001)	1	
(010)	2	
(011)	3	✓
(100)	4	
(101)	5	
(110)	6	
(111)	7	

E = 2
s = ?

2-way set associative

Set	Tag	Data
(00)0		
(01)1		
(10)2		
(11)3	✓	✓

E = 4
s = ?

4-way set associative

Set	Tag	Data
(0)0		
(1)1	✓	✓

Block Placement and Replacement

- ❖ Any empty block in the correct set may be used to store block
 - **Valid bit** for each cache block indicates if data is valid (1) or garbage (0)
- ❖ If there are no empty blocks, which one should we replace?
 - No choice for direct-mapped caches
 - Caches typically use something close to least recently used (LRU) (hardware usually implements “not most recently used”)

Direct-mapped

Set	V	Tag	Data
0			
1			
2			
3			
4			
5			
6			
7			

2-way set associative

Set	V	Tag	Data
0			
1			
2			
3	1		
3	1		

(Handwritten red annotations: a circle around the '1' in the first row of set 3, a bracket around the two rows of set 3, and an arrow pointing to the second row of set 3)

4-way set associative

Set	V	Tag	Data
0			
1			
1			
1			
1			

(Handwritten red annotations: a large bracket on the right side of the four rows of set 1)

Polling Questions

❖ We have a cache of size 2 KiB with block size of 128 B. If our cache has 2 sets, what is its associativity?

▪ Vote in Ed Lessons

A. 2

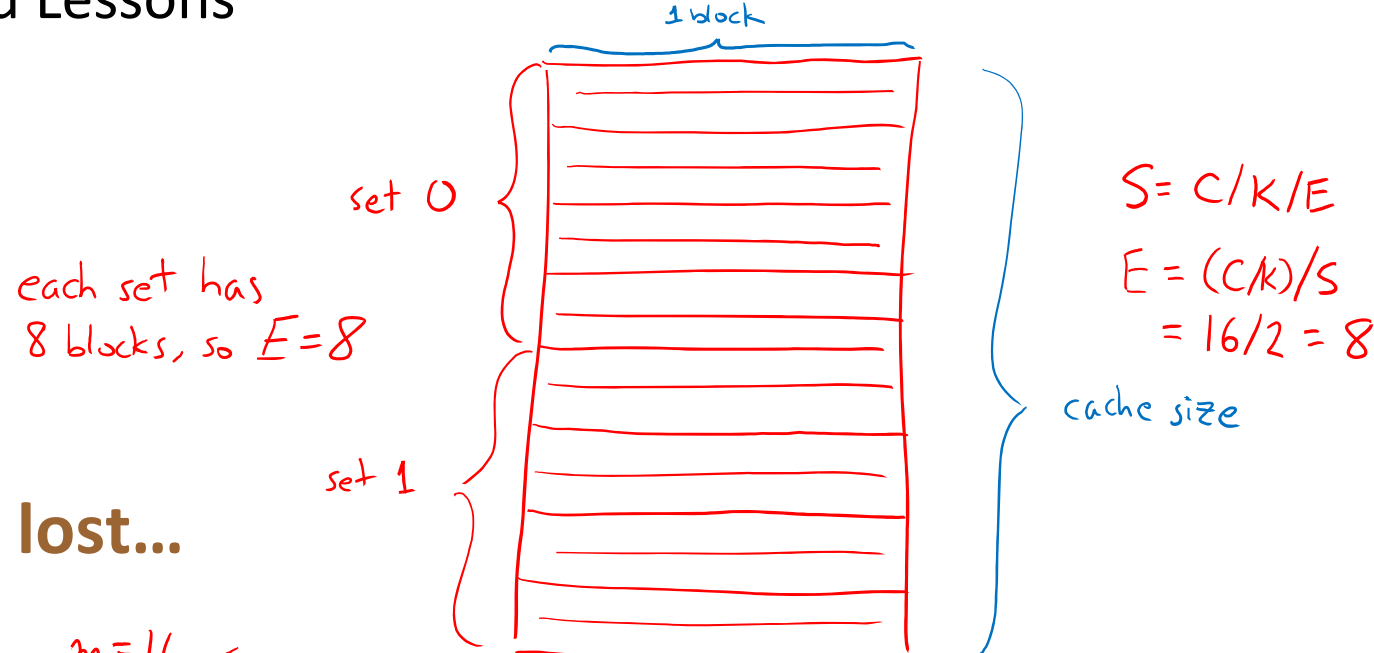
B. 4

C. 8

D. 16

E. We're lost...

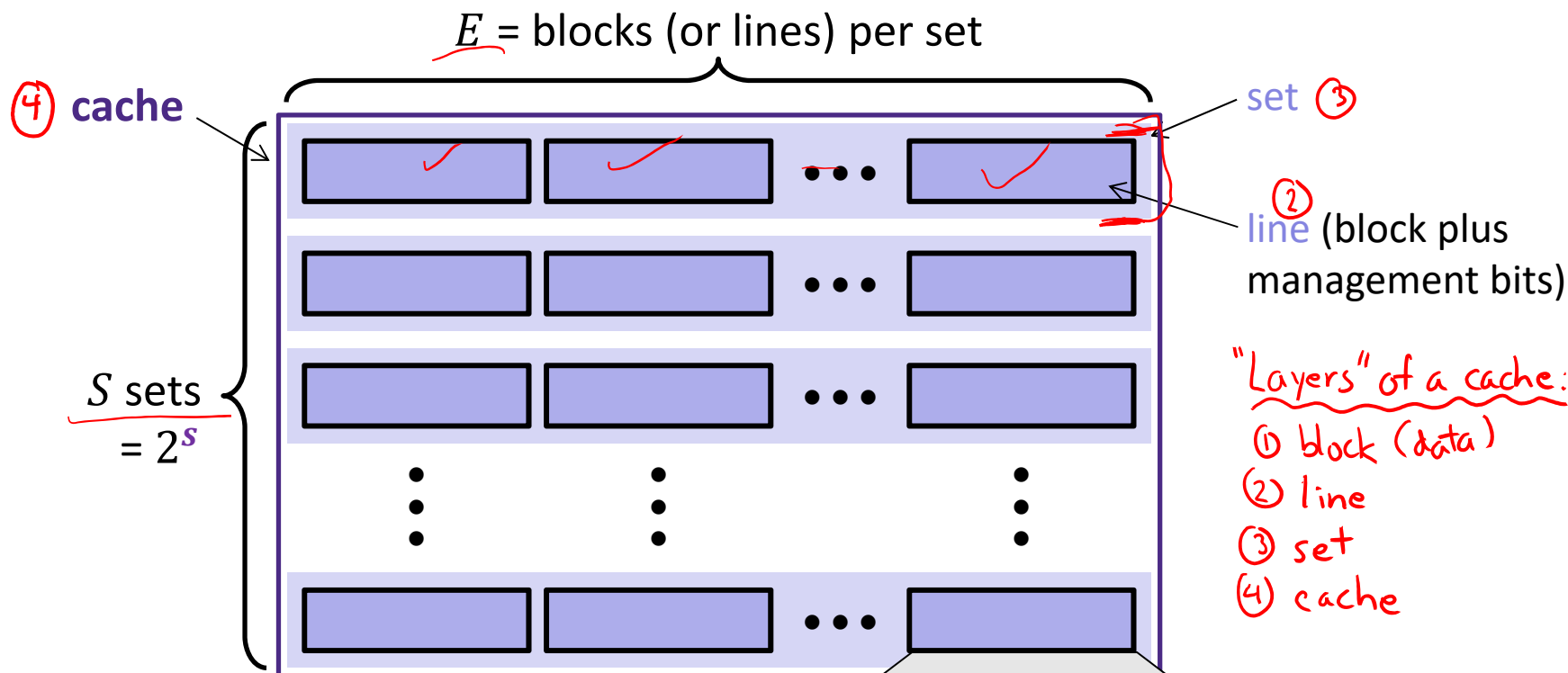
$C = 2^{11} B$
 $K = 2^7 B$
 cache holds $C/K = 2^{11-7} = 2^4 = 16$ blocks



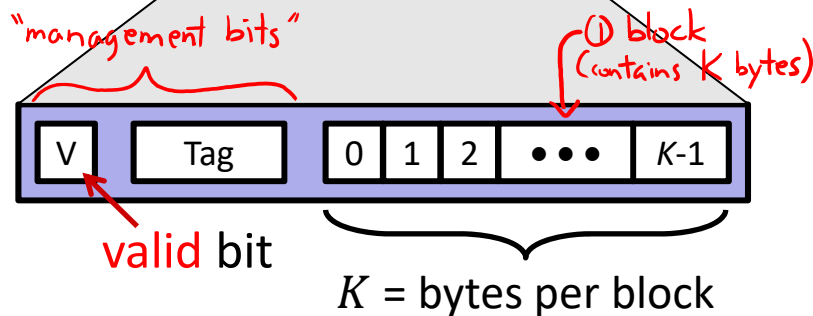
❖ If addresses are 16 bits wide, how wide is the Tag field? $k = \log_2(K) = 7$ bits, $s = \log_2(S) = 1$ bit, $t = m - s - k = 8$ bits

General Cache Organization (S, E, K)

associativity
sets *block size*



Cache size:
 $C = K \times E \times S$ data bytes
 (doesn't include V or Tag)



Notation Review

- ❖ We just introduced a lot of new variable names!
 - Please be mindful of block size notation when you look at past exam questions or are watching videos

Parameter	Variable	Formulas
Block size	K (B in book)	$M = 2^m \leftrightarrow m = \log_2 M$ $S = 2^s \leftrightarrow s = \log_2 S$ $K = 2^k \leftrightarrow k = \log_2 K$ $C = K \times E \times S$ $s = \log_2(C/K/E)$ $m = t + s + k$
Cache size	C	
Associativity	E	
Number of Sets	S	
Address space	M	
Address width	m	
Tag field width	t	
Index field width	s	
Offset field width	k (b in book)	

Example Cache Parameters Problem

- ❖ 2^{12} B $m = 12$ bits
 4 KiB address space, 125 cycles to go to memory.

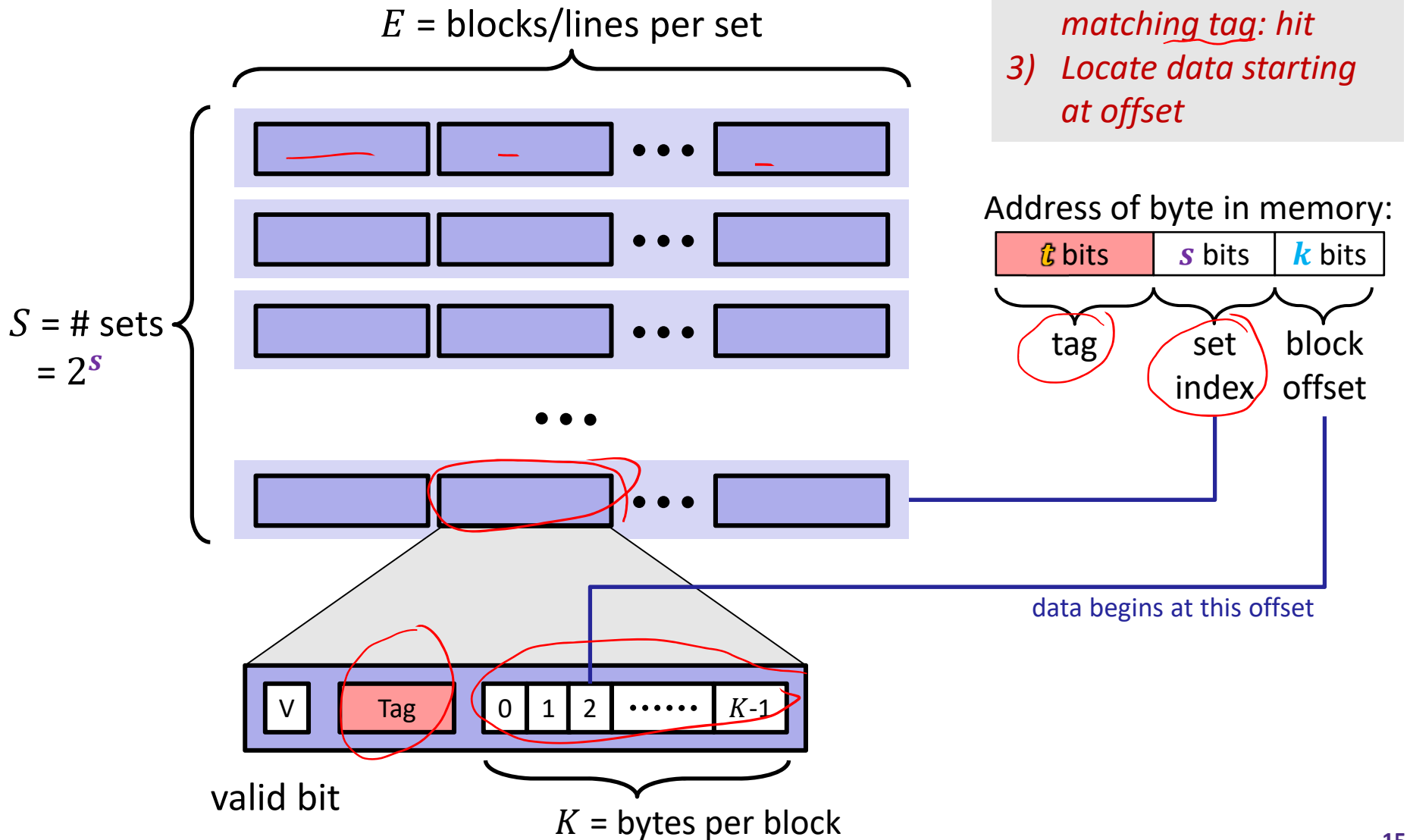
Fill in the following table:

C	Cache Size	256 B	2^8
K	Block Size	32 B	2^5
E	Associativity	2-way	$\frac{2^8}{2^5} = 2^3$ blocks
HT	Hit Time	3 cycles	8 blocks
MR	Miss Rate	20%	4 sets
	Tag Bits	5	
	Index Bits	2 ✓	
	Offset Bits	5 ✓	
	AMAT	$HT + (MR * MP)$ $3 + (0.2 * 125) = 28$ clock cycles 25	

$t = m - s - k$
 $s = \log_2(C/K/E)$
 $k = \log_2(B)$
 $AMAT = HT + MR * MP$

Cache Read

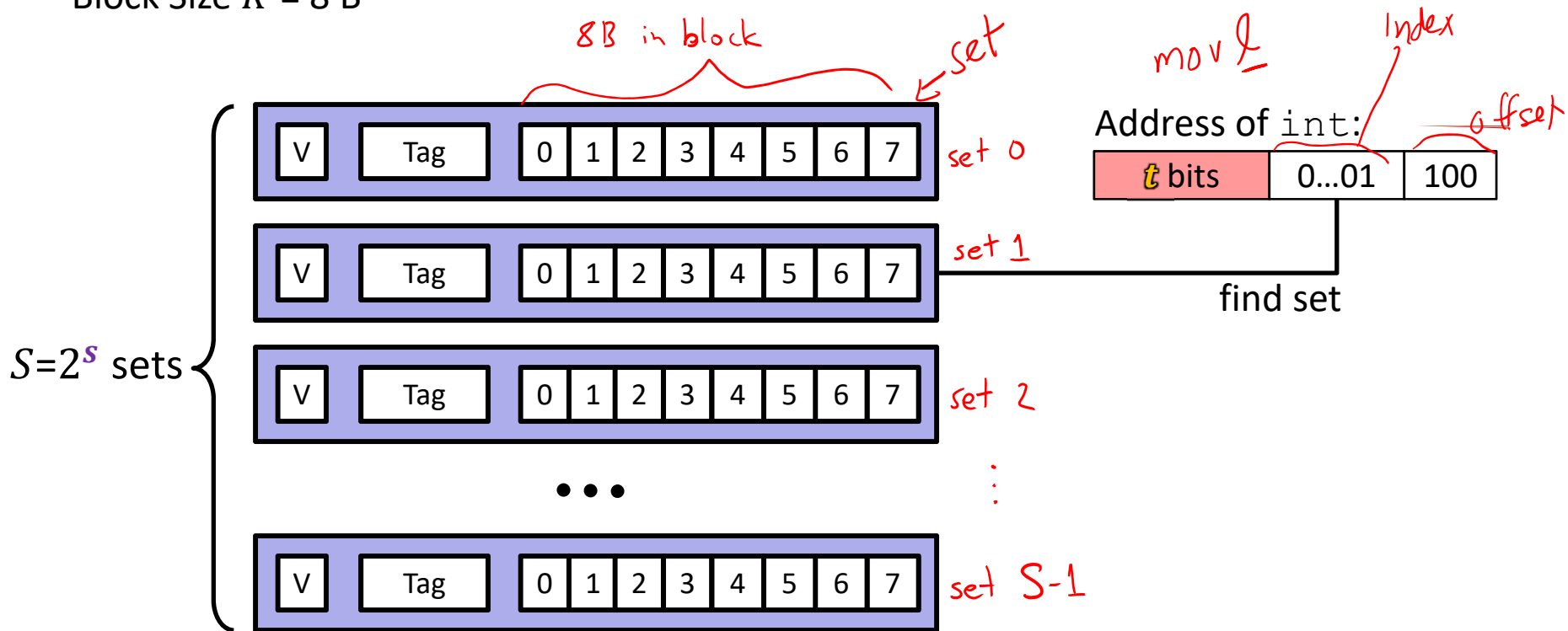
- 1) Locate set
- 2) Check if any line in set is valid and has matching tag: hit
- 3) Locate data starting at offset



Example: Direct-Mapped Cache ($E = 1$)

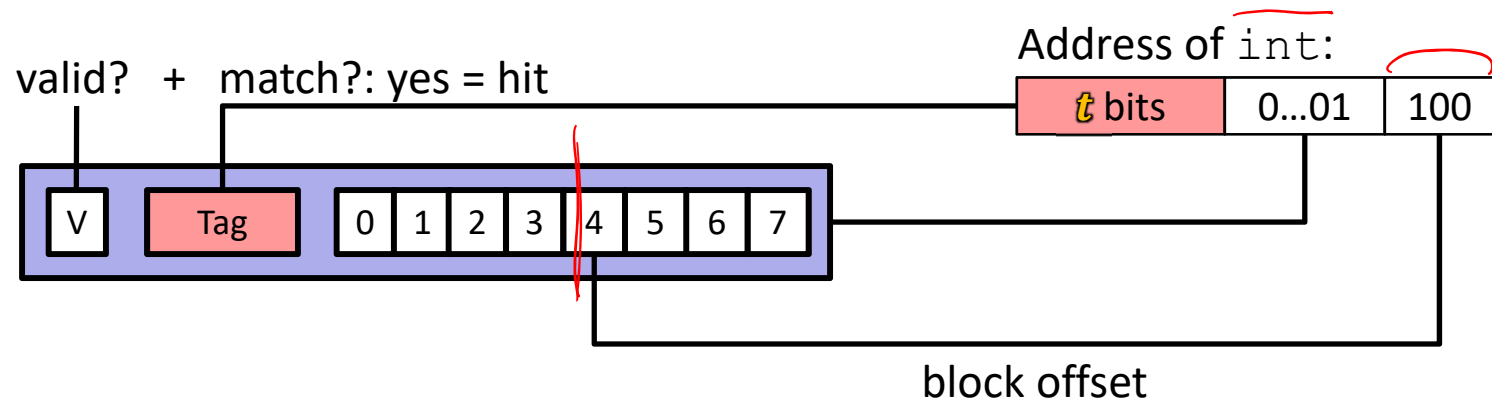
Direct-mapped: One line per set

Block Size $K = 8$ B



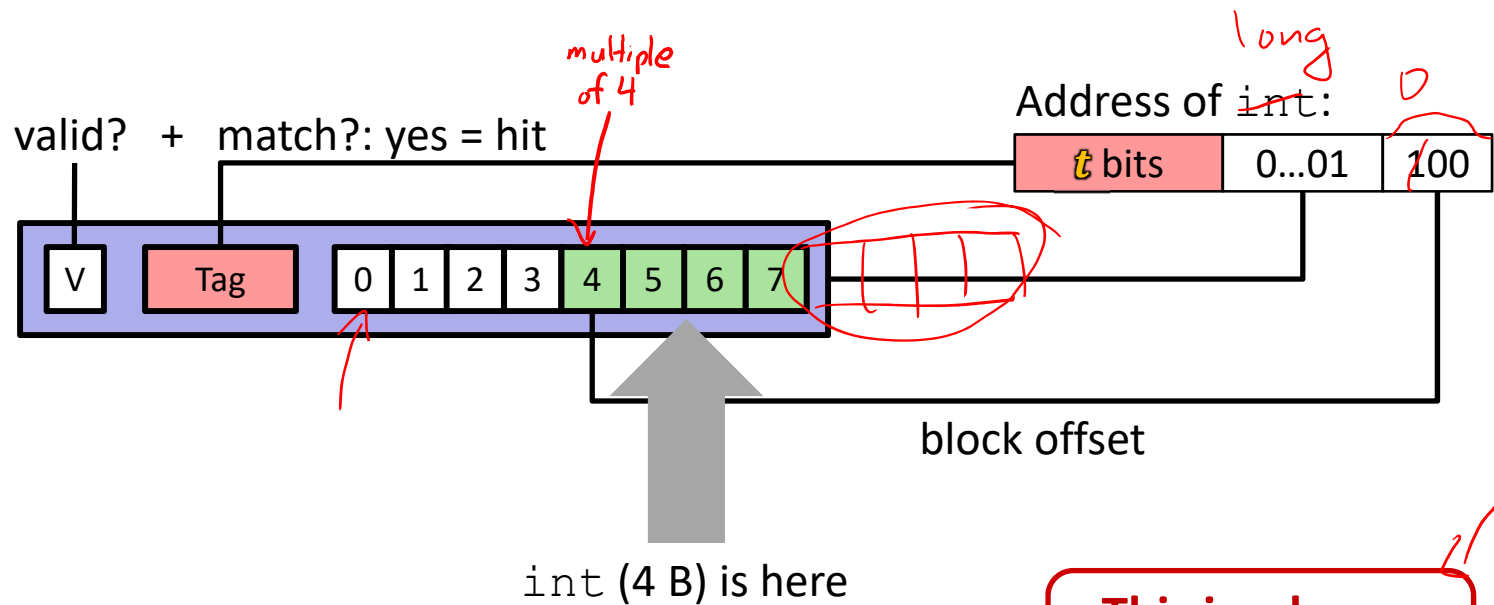
Example: Direct-Mapped Cache ($E = 1$)

Direct-mapped: One line per set
 Block Size $K = 8$ B



Example: Direct-Mapped Cache ($E = 1$)

Direct-mapped: One line per set
 Block Size $K = 8$ B

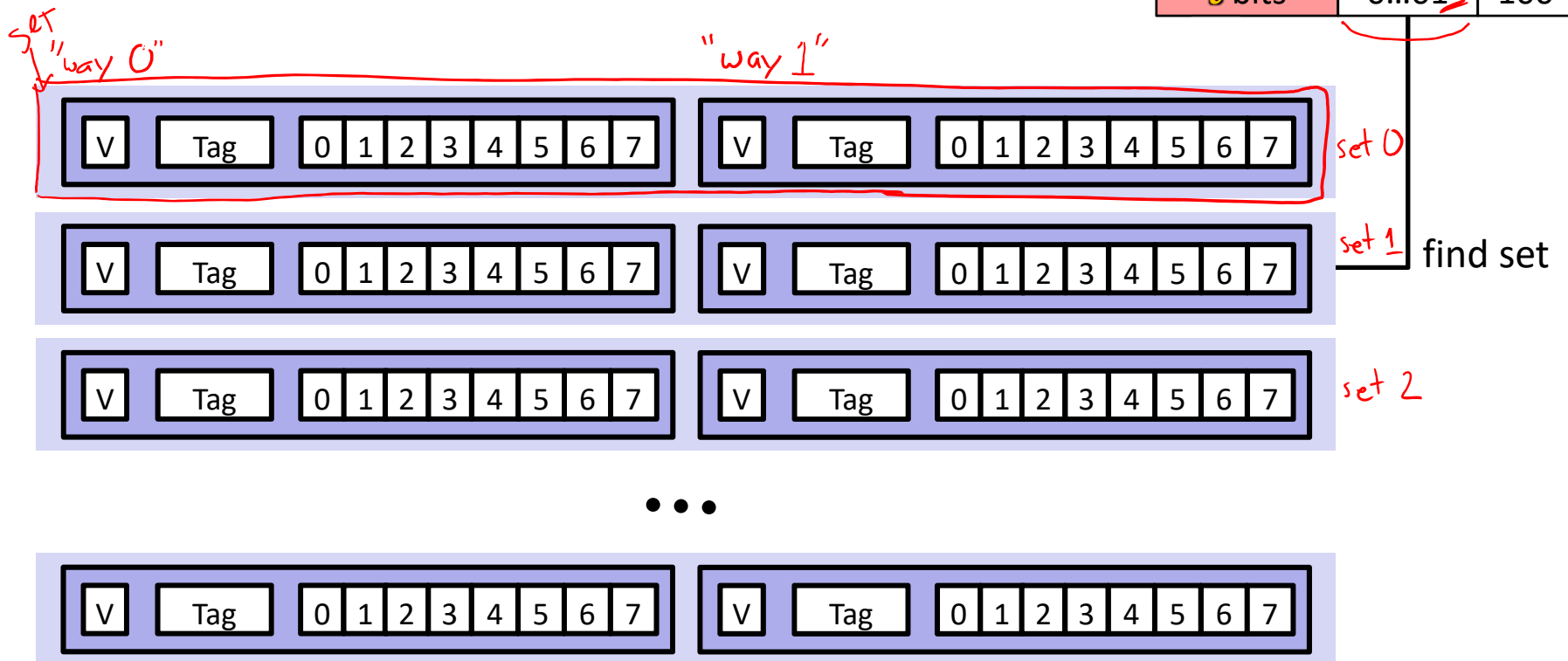
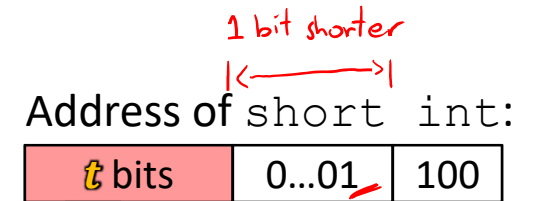


No match? Then old line gets evicted and replaced

no unnecessary extra cache accesses across block boundaries

Example: Set-Associative Cache ($E = 2$)

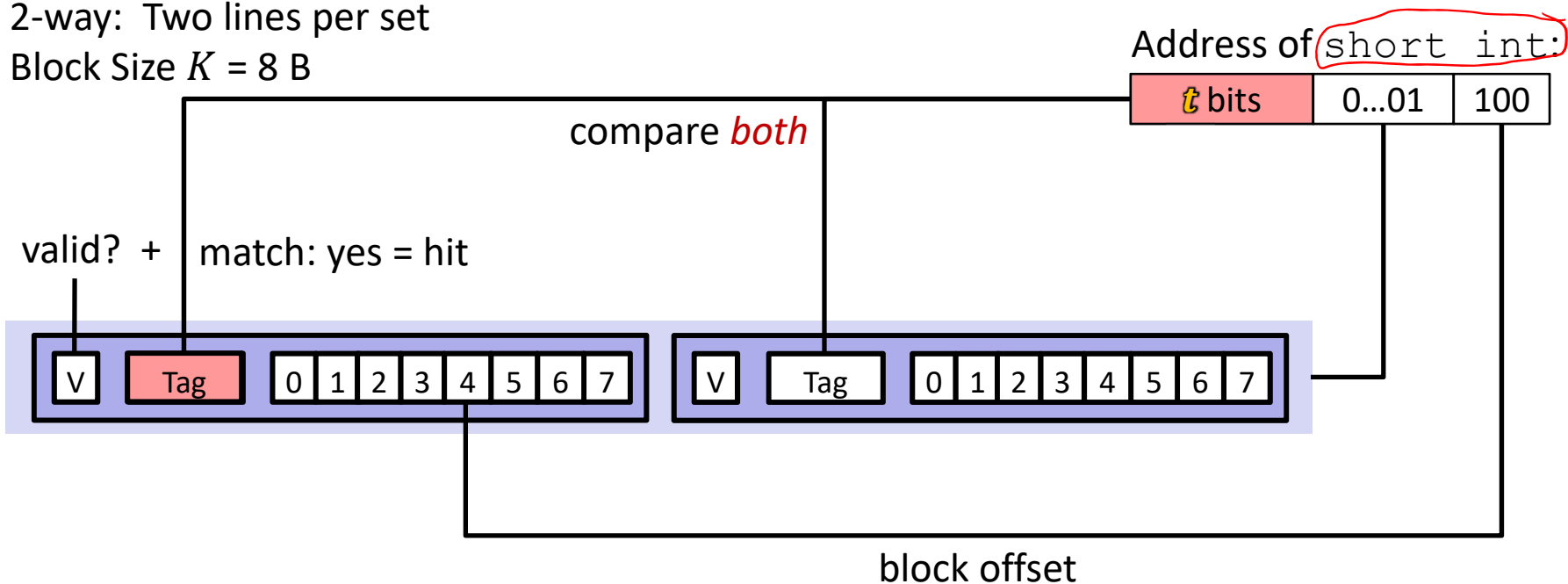
2-way: Two lines per set
 Block Size $K = 8$ B



Example: Set-Associative Cache ($E = 2$)

2-way: Two lines per set

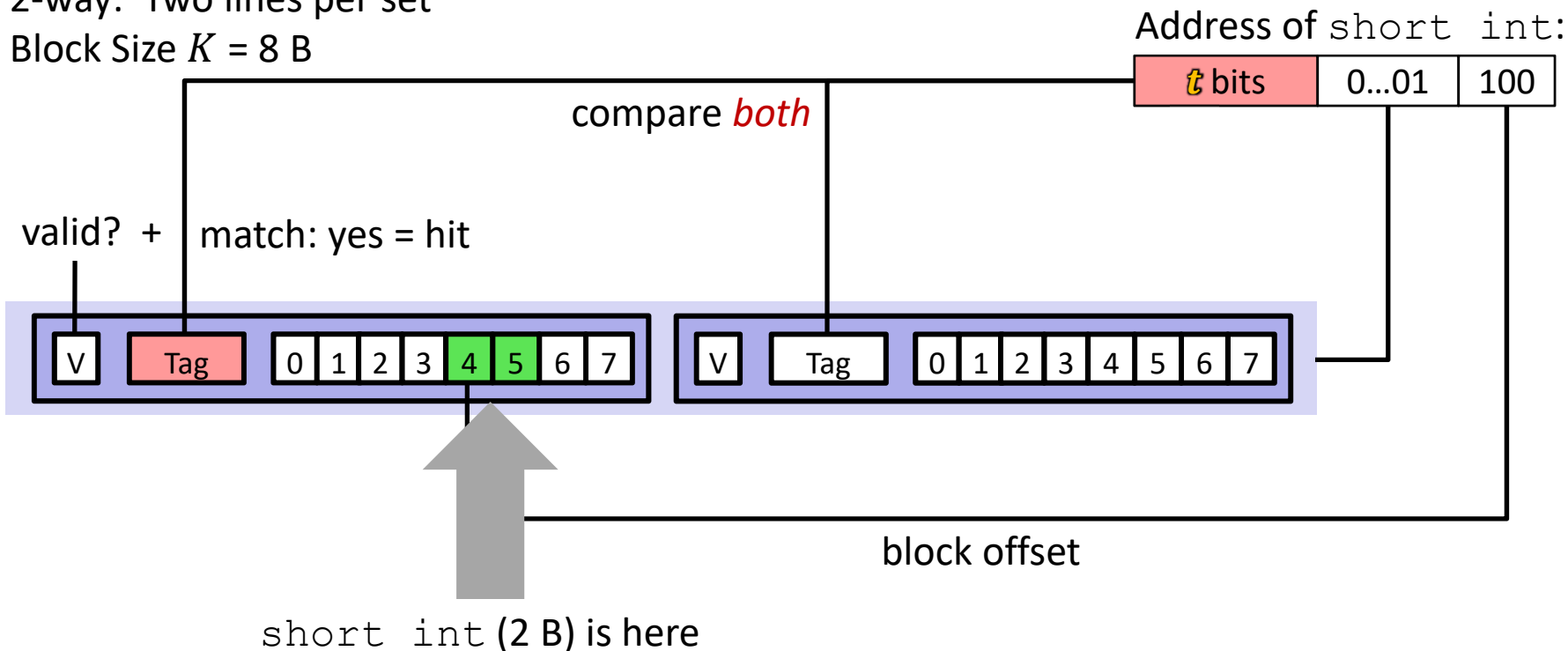
Block Size $K = 8$ B



Example: Set-Associative Cache ($E = 2$)

2-way: Two lines per set

Block Size $K = 8$ B



No match?

- One line in set is selected for eviction and replacement
- Replacement policies: random, least recently used (LRU), ...

Types of Cache Misses: 3 C's!

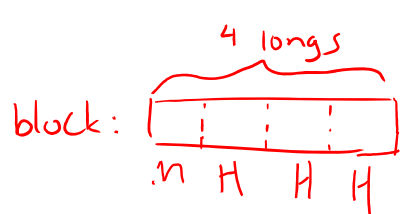
- ❖ **Compulsory** (cold) miss
 - Occurs on first access to a block
- ❖ **Conflict** miss
 - Conflict misses occur when the cache is large enough, but multiple data objects all map to the same slot
 - e.g. referencing blocks 0, 8, 0, 8, ... could miss every time
 - Direct-mapped caches have more conflict misses than E -way set-associative (where $E > 1$)
- ❖ **Capacity** miss
 - Occurs when the set of active cache blocks (the *working set*) is larger than the cache (just won't fit, even if cache was *fully-associative*)
 - **Note:** Fully-associative only has Compulsory and Capacity misses

Example Code Analysis Problem

- Assuming the cache starts cold (all blocks invalid) and `sum`, `i`, and `j` are stored in registers, calculate the **miss rate**: *25%*
 - t = 5 bits, s = 2 bits, k = 5 bits*
 - $m = 12$ bits, $C = 256$ B, $K = 32$ B, $E = 2$

8 bytes per element

```
#define SIZE 8
long ar[SIZE][SIZE], sum = 0; // &ar=0x800
for (int i = 0; i < SIZE; i++)
    for (int j = 0; j < SIZE; j++)
        sum += ar[i][j];
```



	address	tag	index	offset	
<code>ar[0][0]</code>	0b 1000	0	0 0 0	0000	→ M
<code>ar[0][1]</code>	0b 1000	0	0 0 0	1000	→ H
<code>ar[0][2]</code>	0b 1000	0	0 0 1	0000	→ H
<code>ar[0][3]</code>	0b 1000	0	0 0 1	1000	→ H
<code>ar[0][4]</code>	0b 1000	0	0 1 0	0000	→ M
⋮			⋮		⋮
					H
					H
					H