Memory & Caches III

CSE 351 Spring 2024

Instructor:

Elba Garza

Teaching Assistants:

Ellis Haker
Adithi Raghavan
Aman Mohammed
Brenden Page
Celestine Buendia
Chloe Fong
Claire Wang
Hamsa Shankar
Maggie Jiang
Malak Zaki
Naama Amiel
Nikolas McNamee
Shananda Dokka
Stephen Ying
Will Robertson

Playlist: CSE 351 24Sp Lecture Tunes!

Relevant Course Information

- HW 15 due tonight! HW16 due Monday
- HW 17/18 due following Friday (10 May)
  - Covers the major cache mechanics–big homework, start soon!
- Take-home Midterm, May 6th to May 7th
  - 48 hours, but should take 1-3 hours to complete
  - No in-person lecture on Monday the 6th—I will post a new recording instead
- Mid-Course Canvas Survey due May 6th by 11:59 PM
- Lab 3 due Wednesday, May 8th
- Lab 4 releasing soon afterward!
  - Can do Part 1 after today; will need Lecture 19 to do Part 2

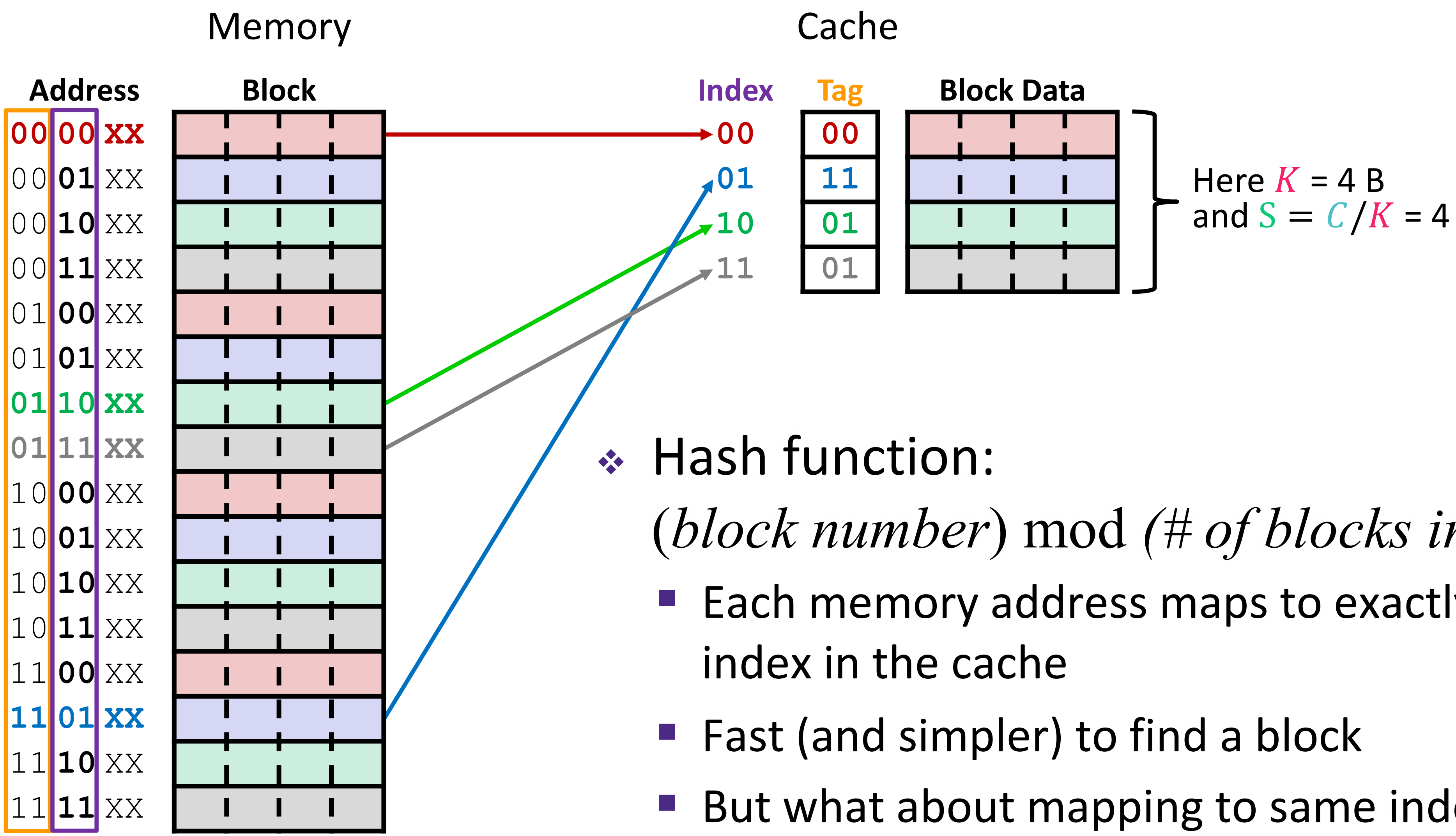
Making memory accesses fast!

- 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

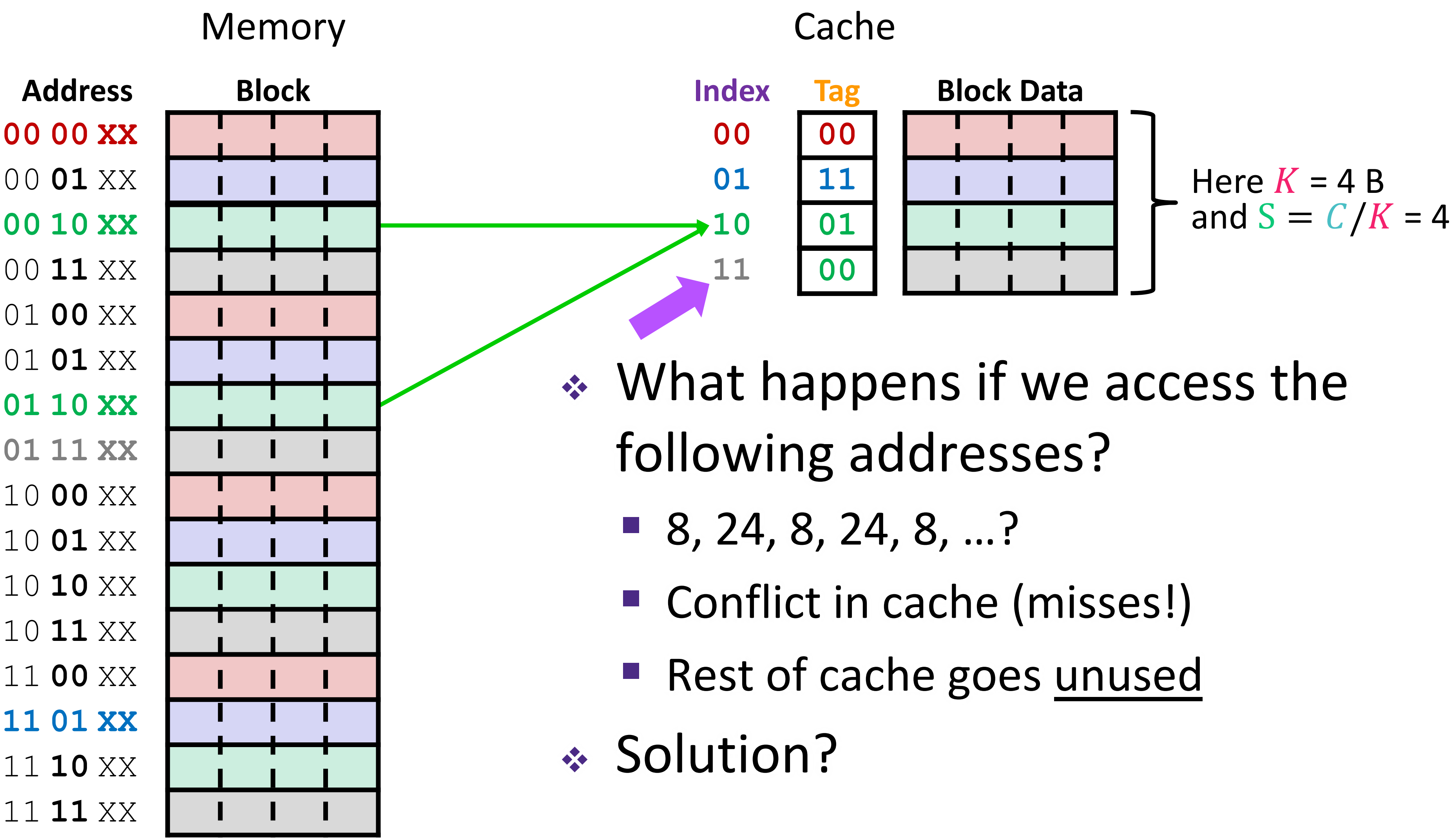
Review: Direct-Mapped Cache



Here K = 4 B
and $S = C/K$ = 4

- Hash function:
  (*block number*) mod *(# of blocks in cache)*
  - Each memory address maps to exactly <u>one</u> index in the cache
  - Fast (and simpler) to find a block
  - But what about mapping to same index?...

Direct-Mapped: A Problem!



Memory

Address	Block
00 00 **XX**	
00 **01** XX	
00 10 **XX**	
00 **11** XX	
01 **00** XX	
01 **01** XX	
01 10 **XX**	
01 11 **XX**	
10 **00** XX	
10 **01** XX	
10 **10** XX	
10 **11** XX	
11 **00** XX	
11 01 **XX**	
11 **10** XX	
11 **11** XX	

Cache

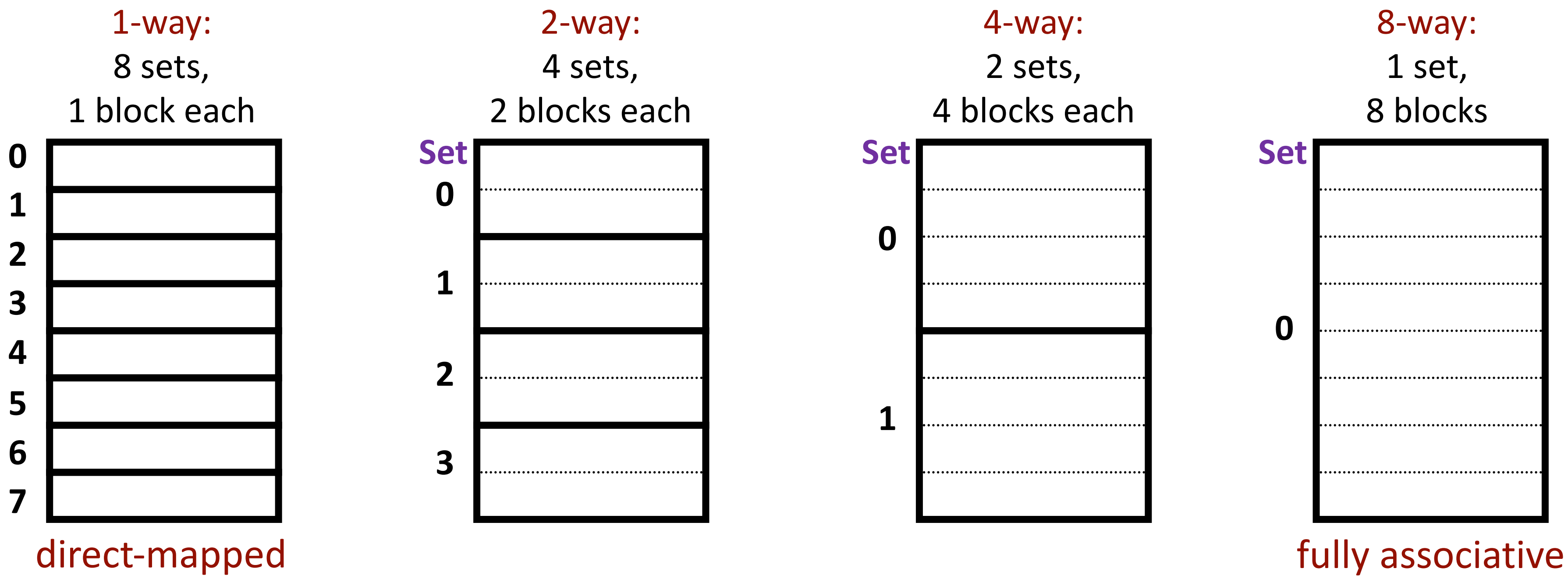
Index	Tag	Block Data
00	00	
01	11	
10	01	
11	00	

Here K = 4 B and $S = C/K$ = 4

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

Associativity: A Solution!

- What if we could store **any** data in **any** place in the cache?
  - But: requires more complicated hardware ⟹ more power consumed, slower
- Let's combine the two ideas:
  - Each address maps to exactly one **set**, but each set can store block in more than one **way** in the set!



1-way:
8 sets,
1 block each

2-way:
4 sets,
2 blocks each

4-way:
2 sets,
4 blocks each

8-way:
1 set,
8 blocks

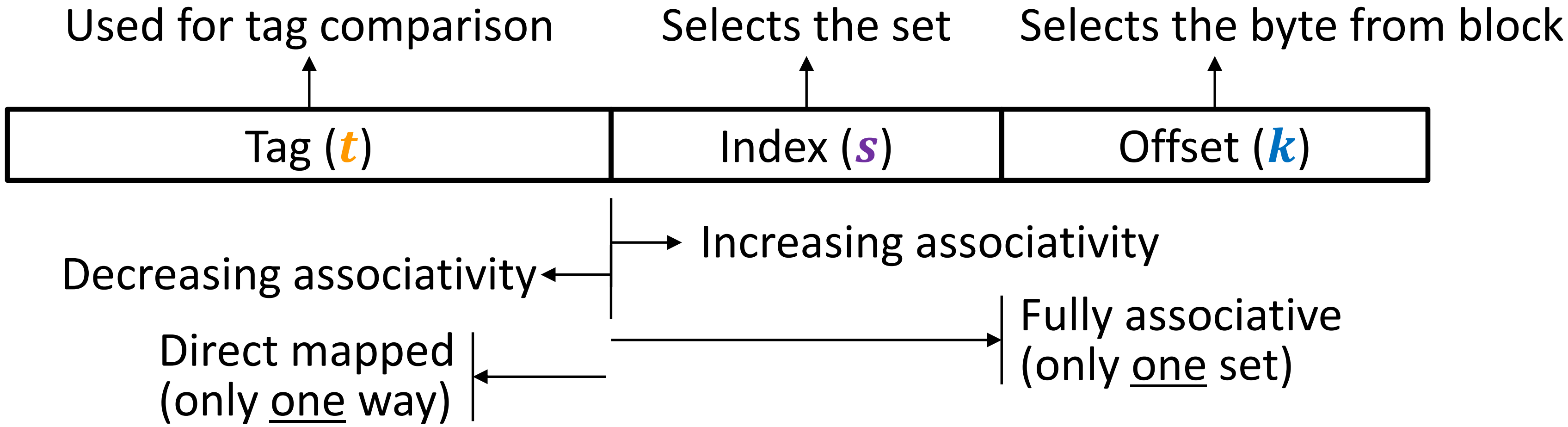
direct-mapped

fully associative

Cache Organization (3)

Note: The textbook uses "b" for offset bits

- Associativity (E): number of **ways** to store in 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 $\log_2(C/K/E)$ = $\boldsymbol{s}$ bits of block address
    - Direct-mapped: E = 1, so $\boldsymbol{s}$ = $\log_2(C/K)$ as we saw previously
    - Fully associative: E = C/K, so $\boldsymbol{s}$ = 0 bits



Example Placement

block size K:	16 B
Capacity C/K:	8 blocks
Address m:	16 bits

- Where would data from address `0x1833` be placed?
  - Binary: `0b 0001 1000 0011 0011`

m-bit address:

t	s	k
Tag (t)	Index (s)	Offset (k)

$t = m - s - k$

$s = \log_2(C/K/E)$

$k = \log_2(K)$

$E = 1$

$s =$

Direct-mapped

Set	Tag	Data
0		
1		
2		
3		
4		
5		
6		
7		

$E = 2$

$s =$

2-way set associative

Set	Tag	Data
0		
1		
2		
3		

$E = 4$

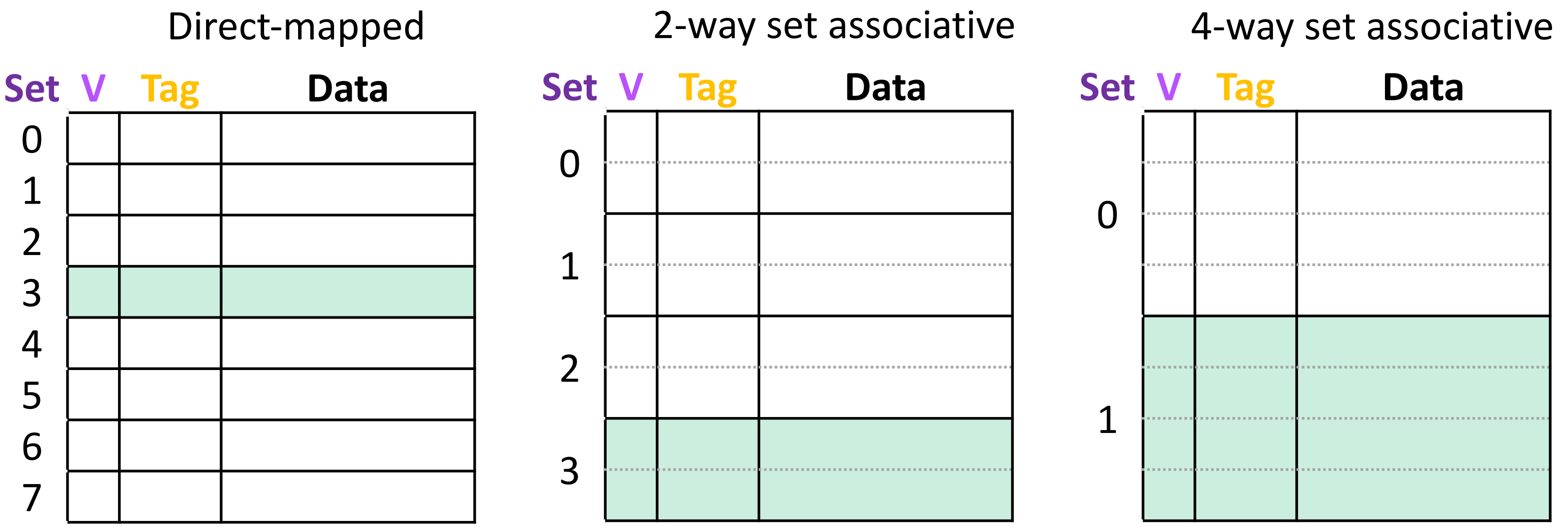
$s =$

4-way set associative

Set	Tag	Data
0		
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 valid (`1`) or mystery (`0`) data
- If there are no empty blocks, which one should we replace? i.e. replacement policy
  - No choice for direct-mapped caches—gotta replace what's there. Super easy.
  - Otherwise, caches typically use something close to **least recently used (LRU)** (hardware usually implements "*not most recently used*")

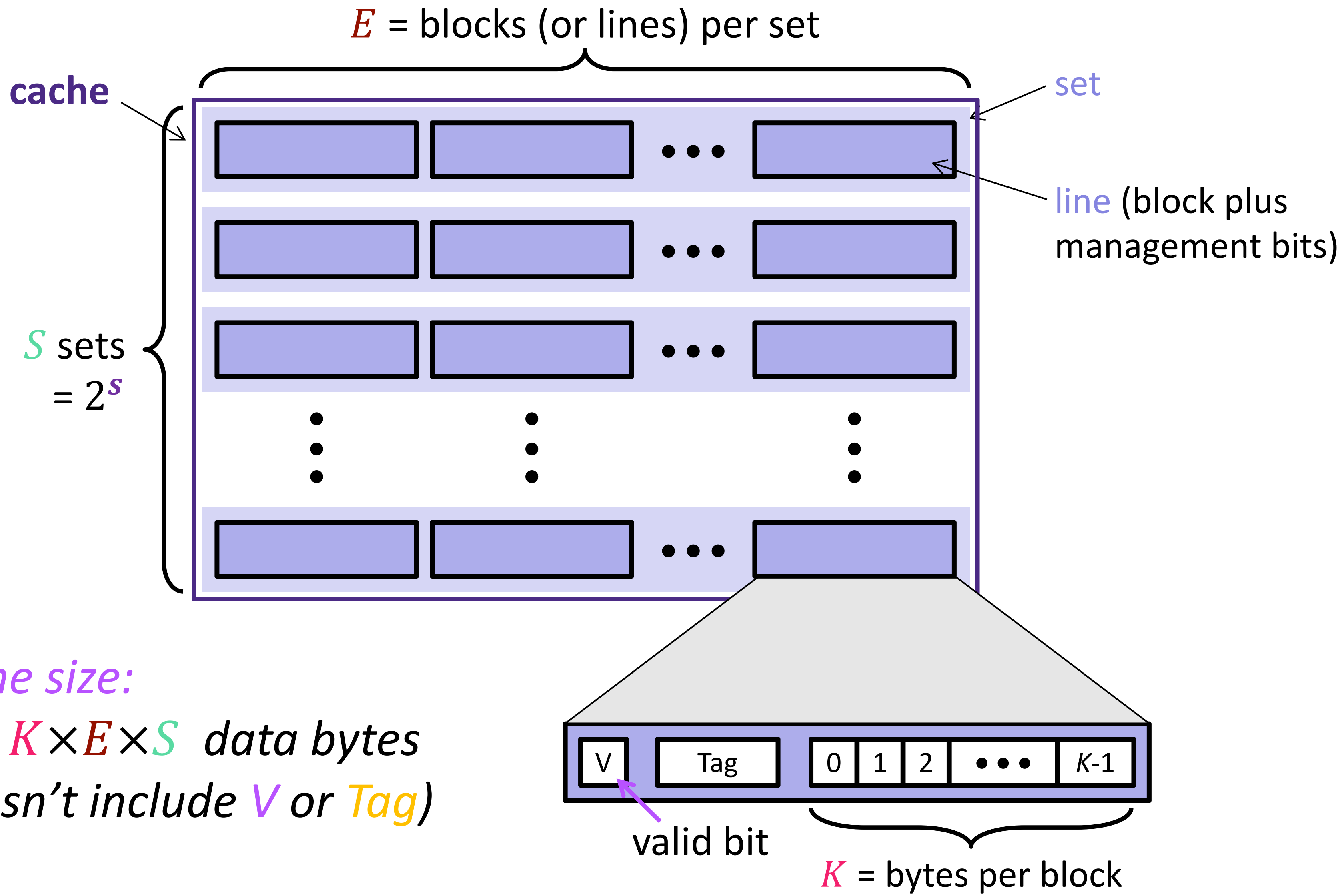


Polling Questions

- We have a cache of size 2 KiB with block size of 128 B. If our cache has 2 sets, what is its associativity?
  - A. 2
  - B. 4
  - C. 8
  - D. 16
  - E. We're lost…

- If addresses are 16 bits wide, how wide is the Tag field?

General Cache Organization (S, E, K)



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

Example Cache Parameters Problem

- 1 KiB address space, 125 cycles to go to memory.
  Fill in the following table:

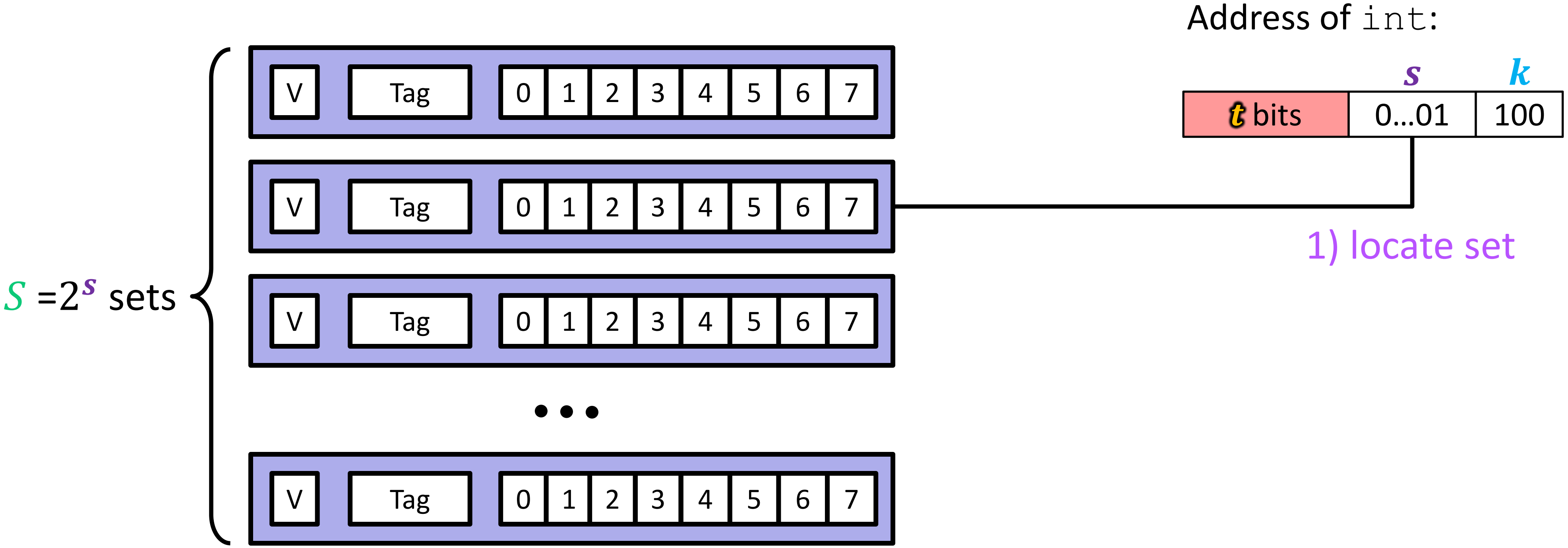
Cache Size C	64 B
Block Size K	8 B
Associativity E	2-way
Hit Time	3 cycles
Miss Rate	20%
Address width (m)	
Tag Bits (t)	
Index Bits (s)	
Offset Bits (k)	
AMAT	

Read: Direct-Mapped Cache (E = 1)

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

Direct-mapped: One line per set

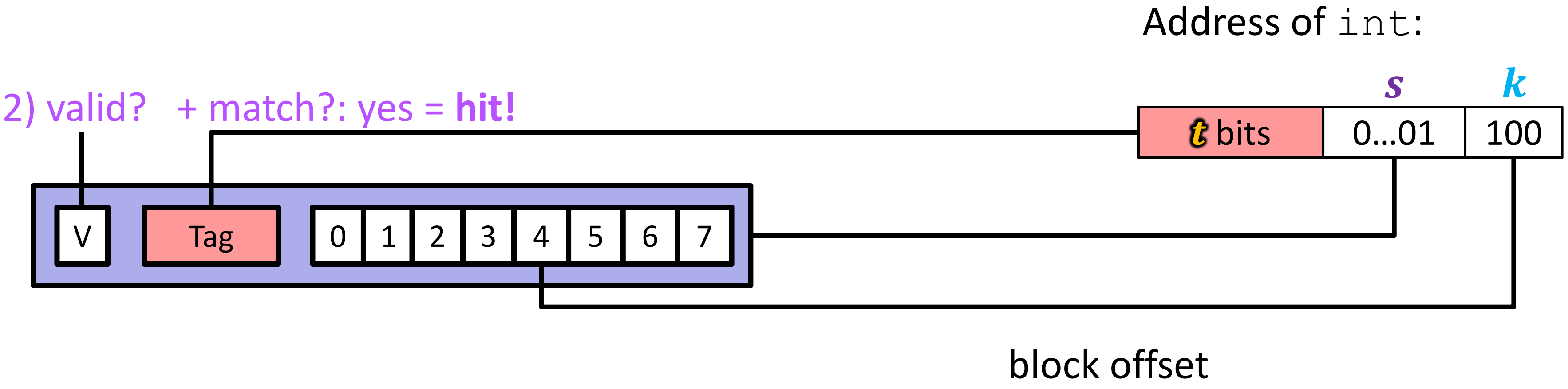
Block Size K = 8 B



Read: Direct-Mapped Cache (E = 1)

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

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

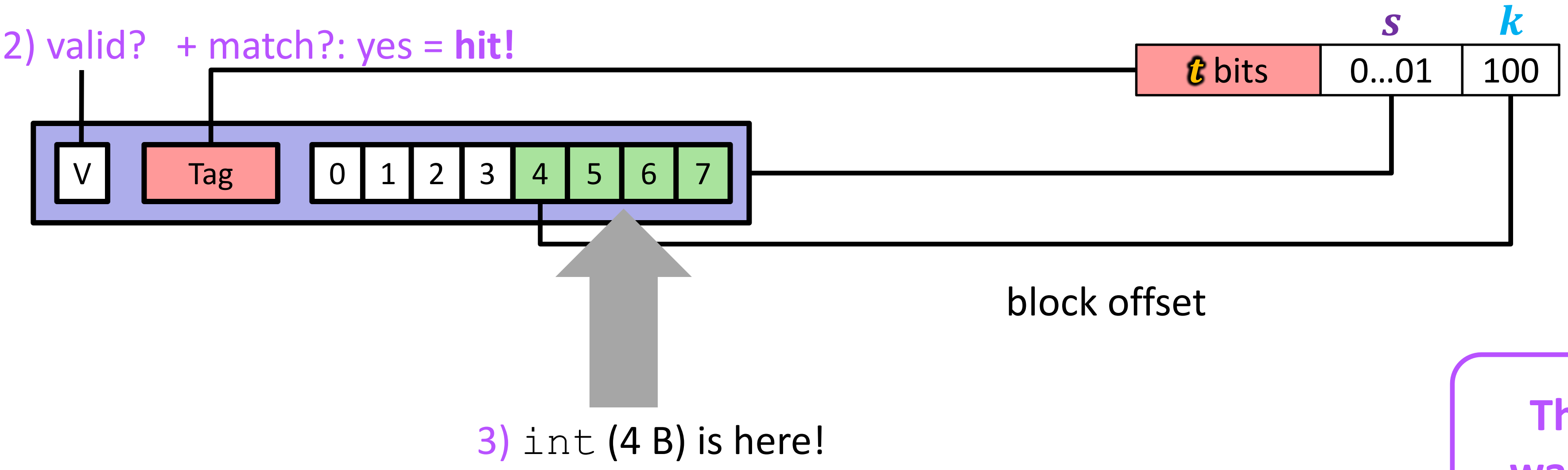


Read: Direct-Mapped Cache (E = 1)

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

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

Address of `int`:



This is why we want alignment!

No match? Then old line/block gets evicted and replaced!

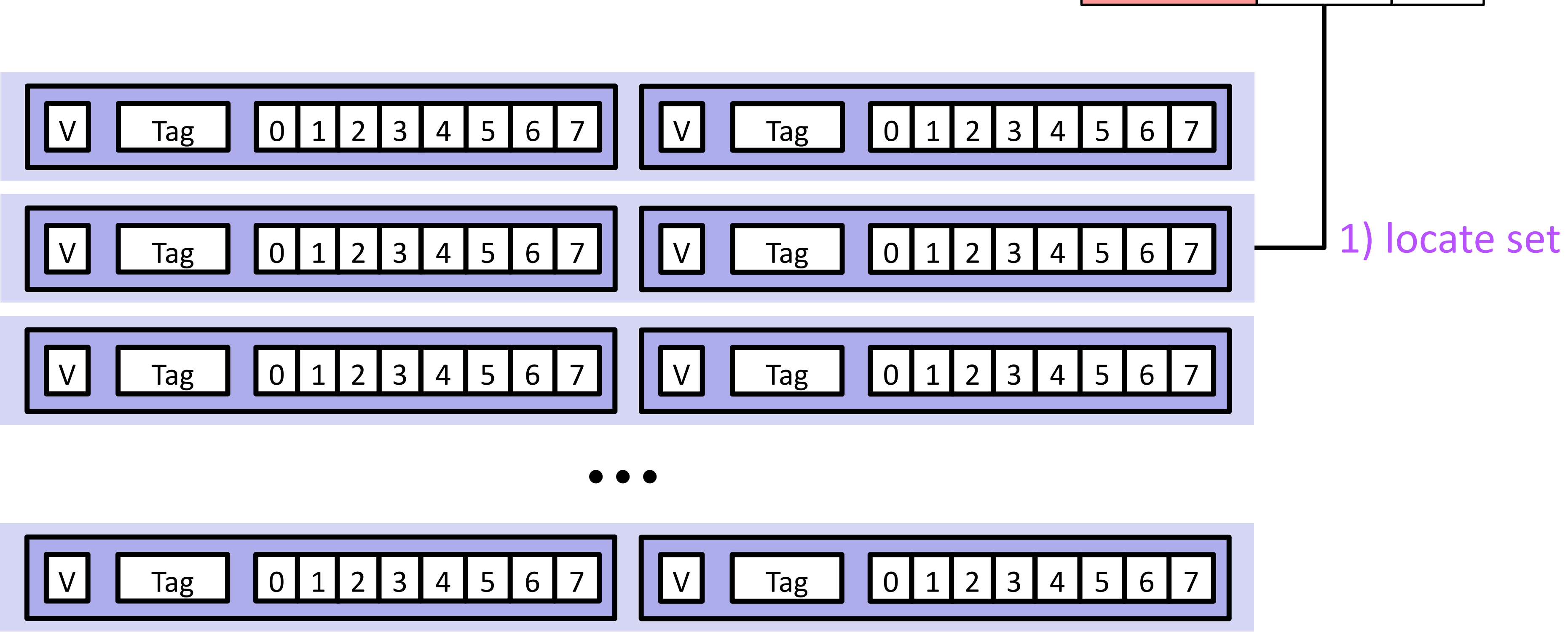
Read: Set-Associative Cache (E = 2)

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

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

Address of `short int`:

t bits	0…01	100

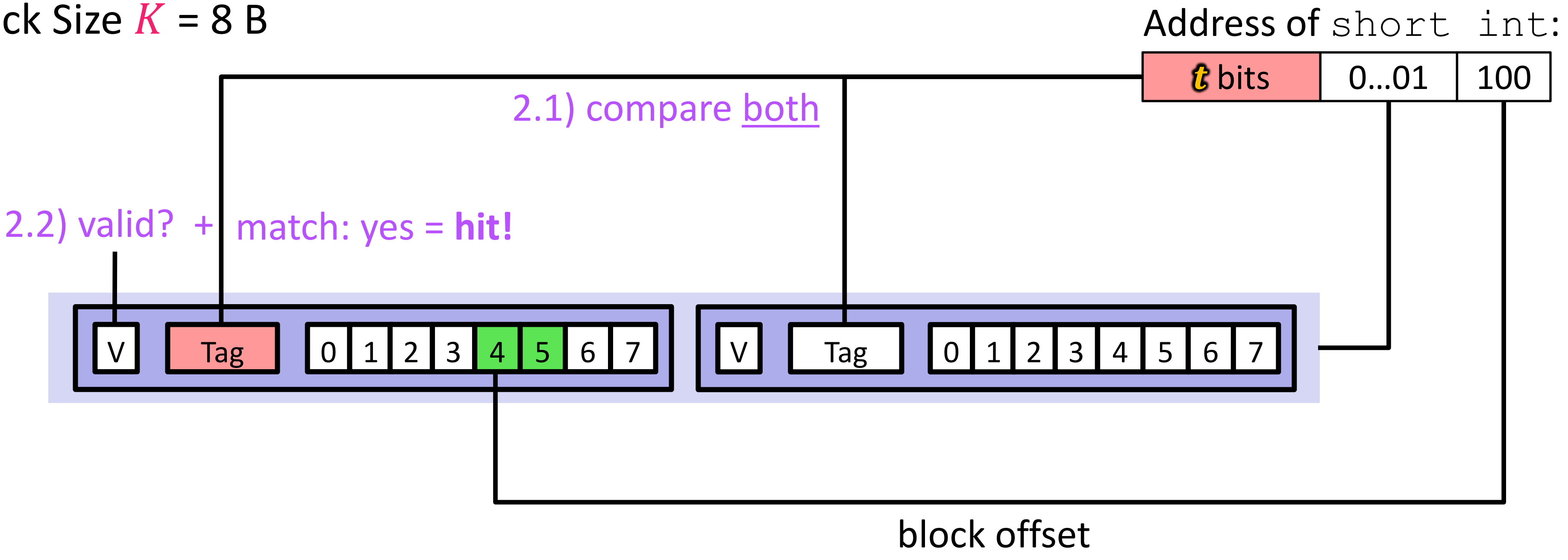


Read: Set-Associative Cache (E = 2)

1) Locate set
2) Check if <u>any line</u> in set is valid and has matching tag: **hit!**
3) Locate data starting at offset

2-way: Two lines per set

Block Size K = 8 B

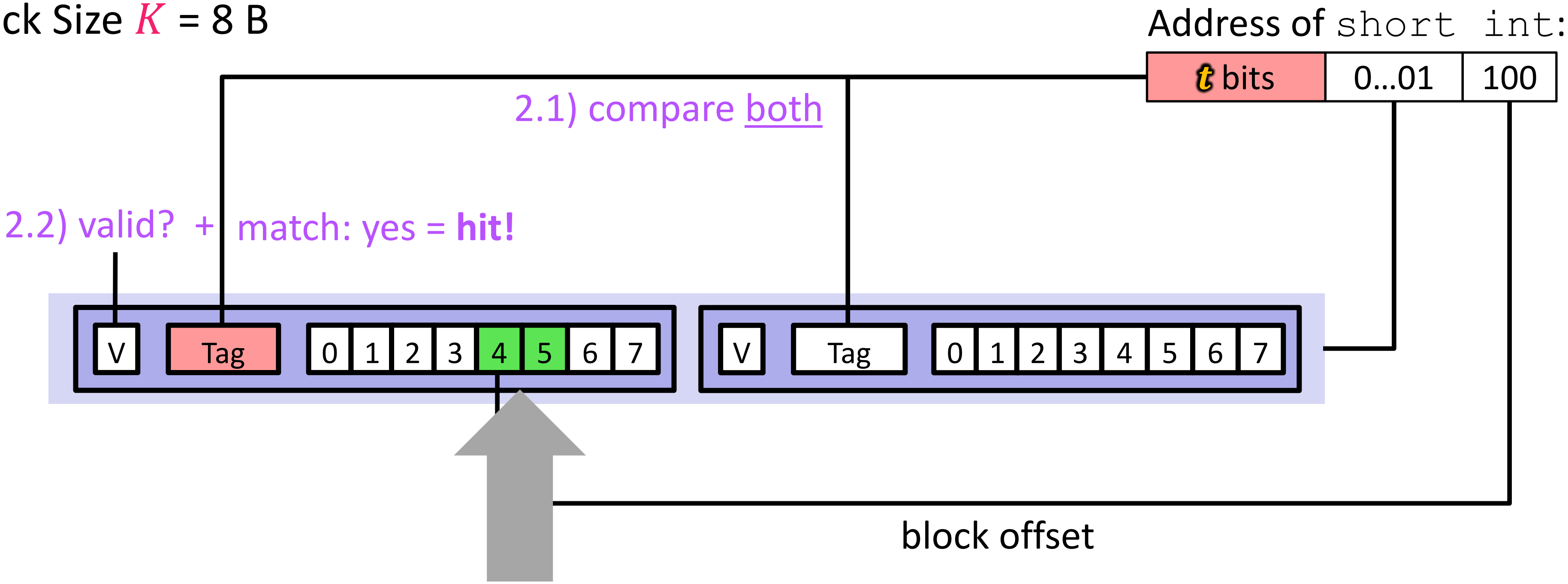


Read: Set-Associative Cache (E = 2)

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

2-way: Two lines per set

Block Size K = 8 B



Address of `short int`:

t bits	0…01	100

2.1) compare both

2.2) valid? + match: yes = **hit!**

V	Tag	0	1	2	3	4	5	6	7

V	Tag	0	1	2	3	4	5	6	7

block offset

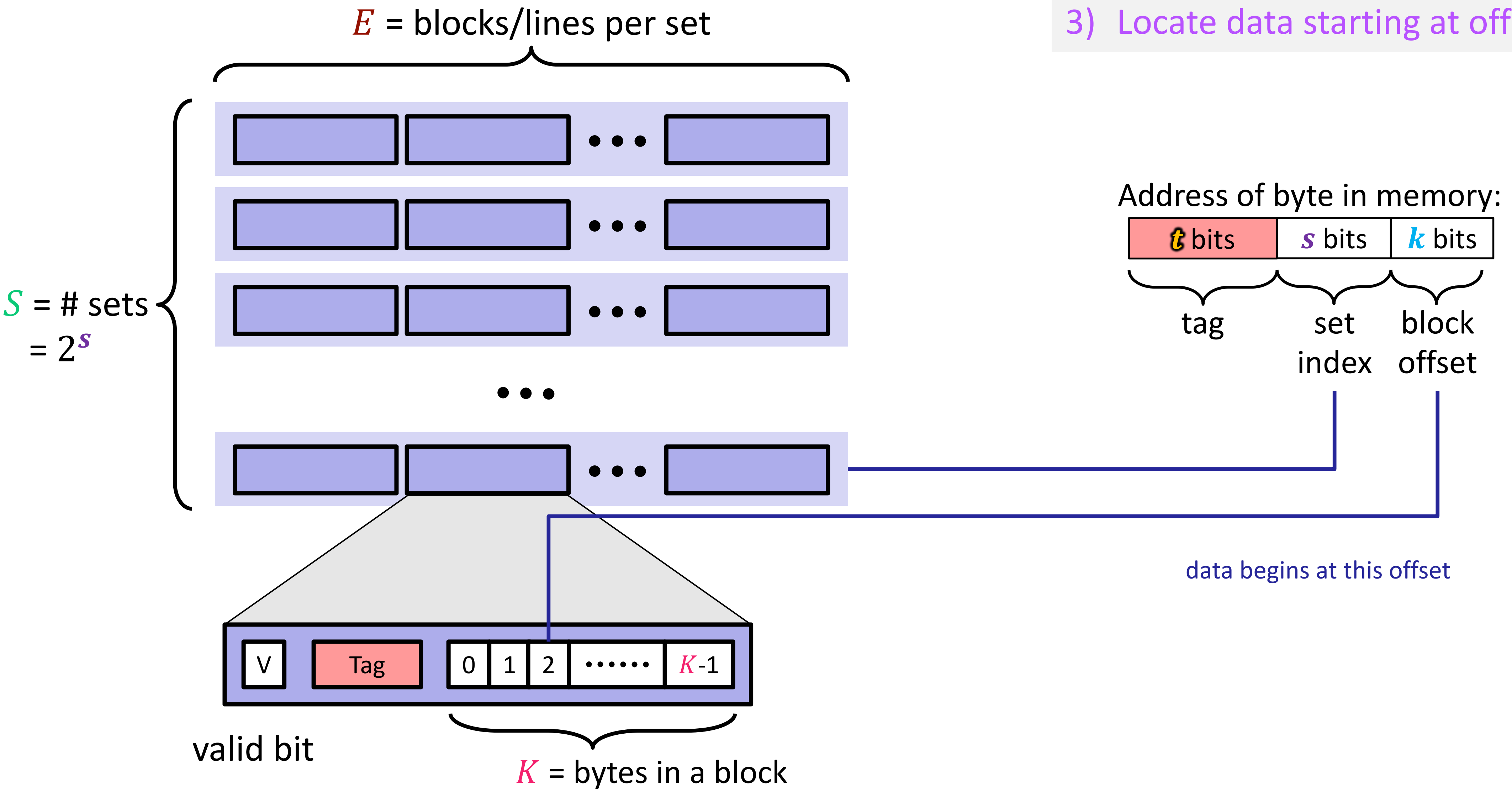
3) `short int` (2 B) is here!

No match?

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

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



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