

The Hardware/Software Interface

Memory & Caches III

Guest Instructor:

Naama Amiel

Instructors:

Justin Hsia, Amber Hu

Teaching Assistants:

Anthony Mangus

Divya Ramu

Grace Zhou

Jessie Sun

Jiuyang Lyu

Kanishka Singh

Kurt Gu

Liander Rainbolt

Mendel Carroll

Ming Yan

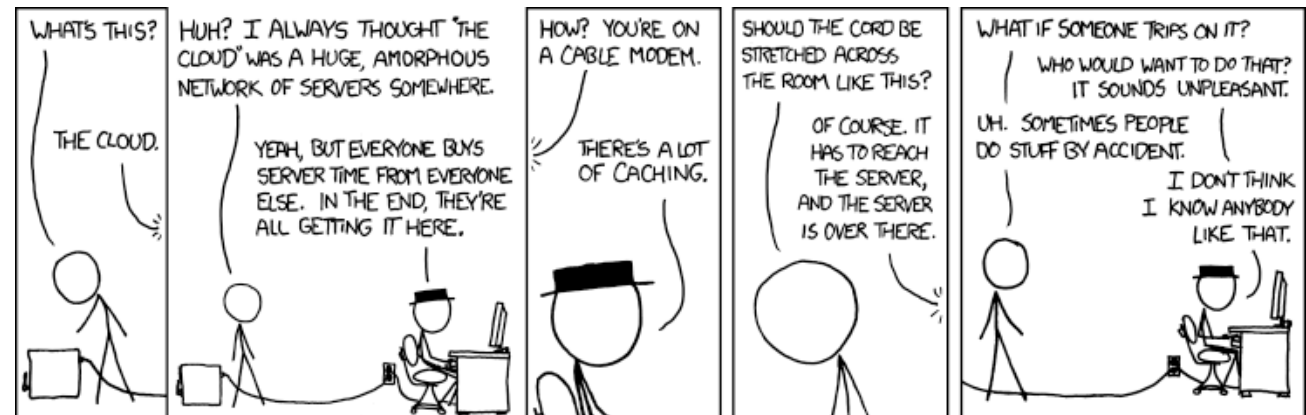
Naama Amiel

Pollux Chen

Rose Maresh

Soham Bhosale

Violet Monserate



<http://xkcd.com/908/>

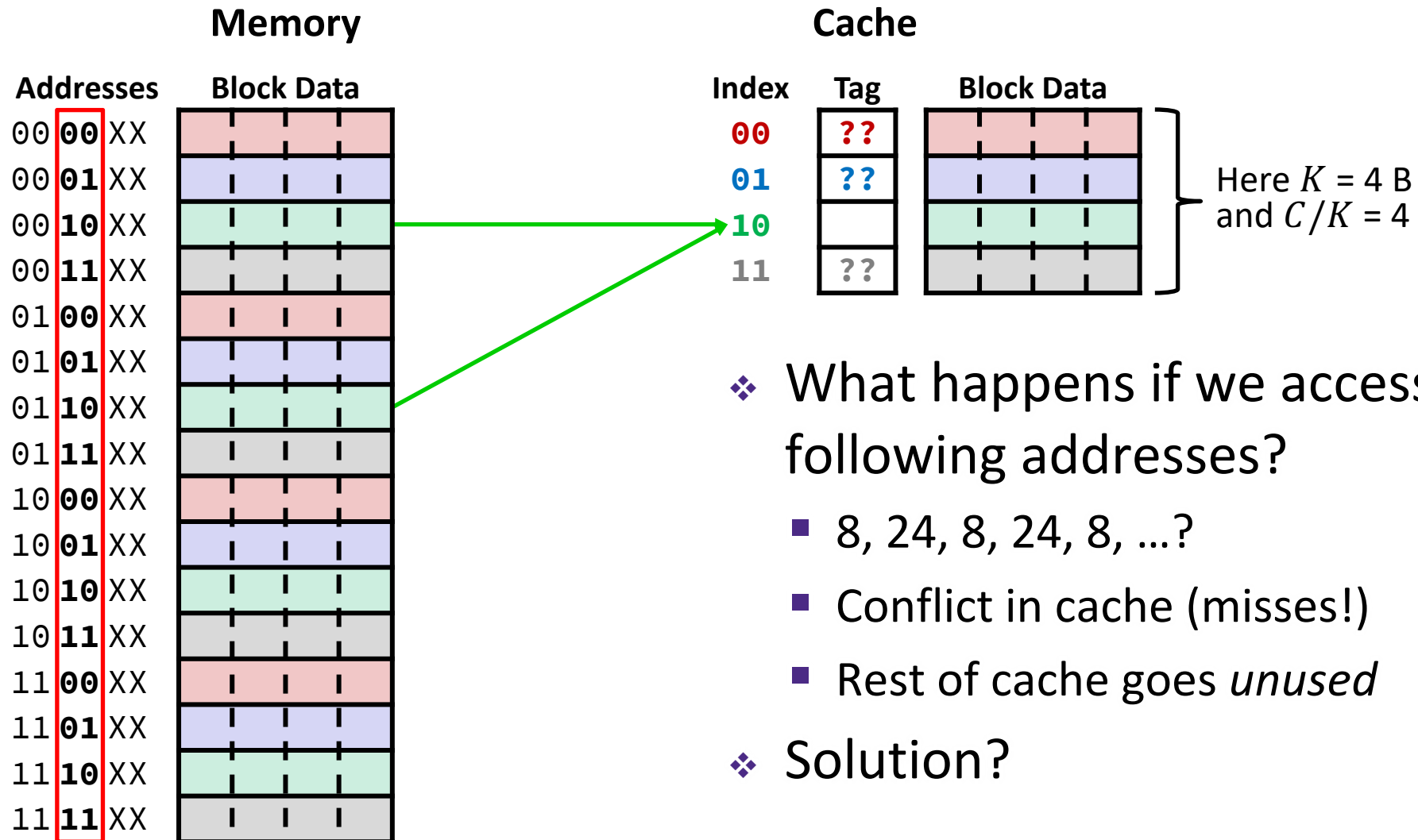
Relevant Course Information

- ❖ HW 17 due Friday (11/7)
 - Don't wait too long, this is a **big** HW
- ❖ HW 18 due Wednesday (11/12)
- ❖ Lab 3 due Friday (11/7)
 - Late days open until Monday (11/10)
 - No lecture on Monday, but we still have office hours!
- ❖ Lab 4 released on Friday (11/7)
 - Two-part lab about caching; you can do part 1 with content from this lecture!
- ❖ Midterm grades will be released when we can
 - Regrade requests will be available afterward

Lecture Outline (1/3)

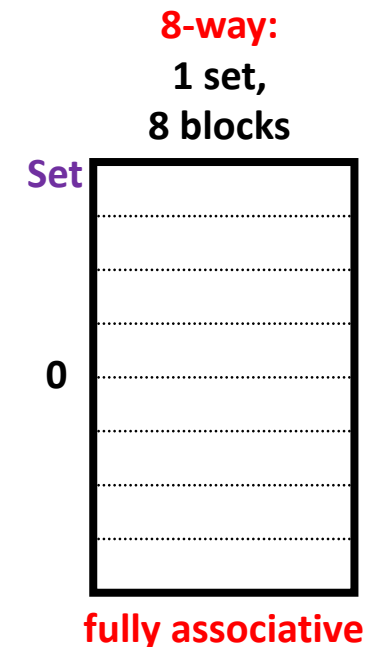
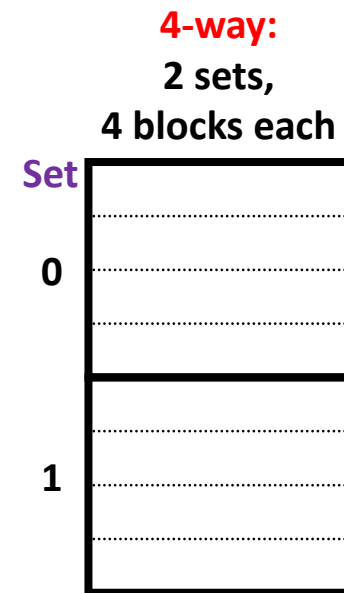
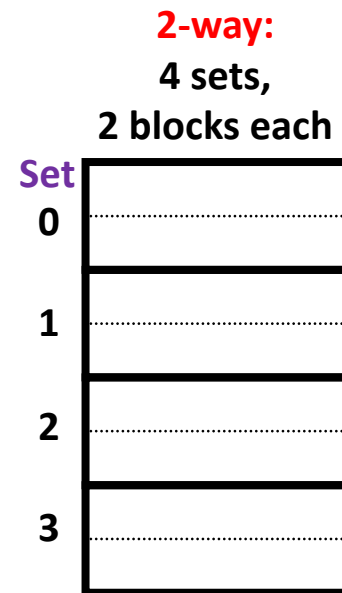
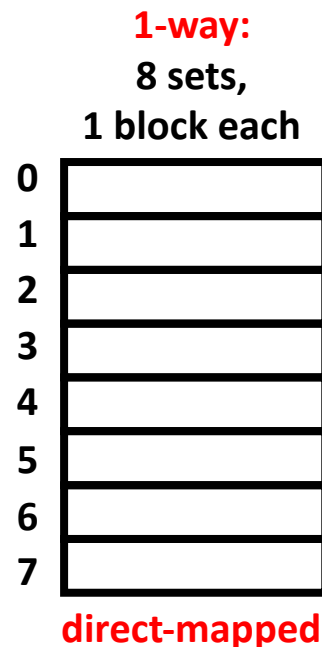
- ❖ **Associativity and Replacement**
- ❖ Cache Organization
- ❖ Example Cache Problems

Review: Direct-Mapped Cache Problem



Associativity Motivation

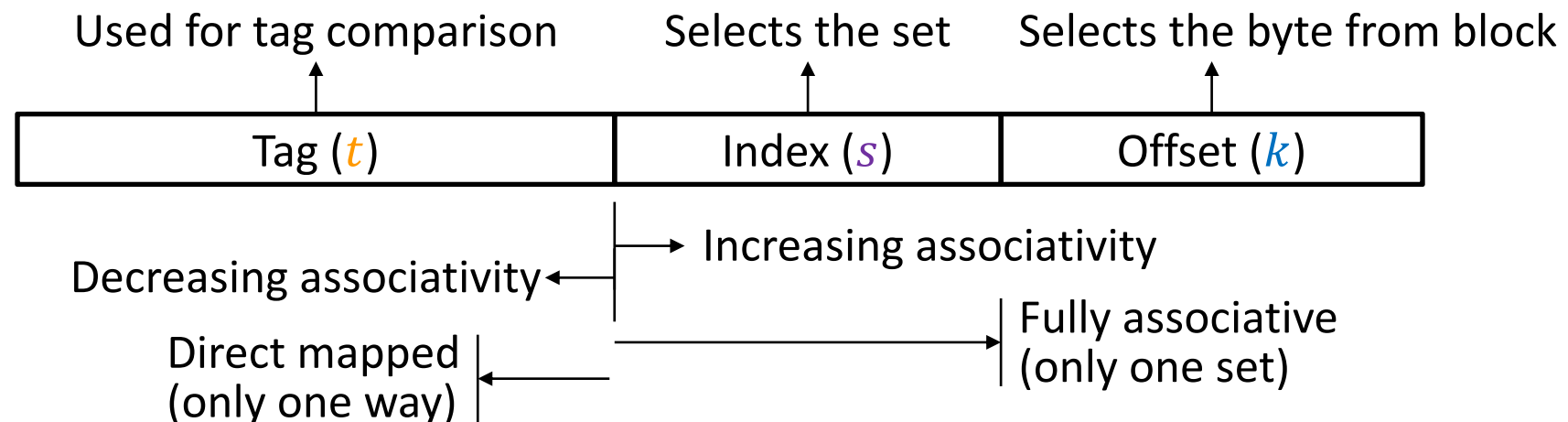
- ❖ What if we could store data in any place in the cache?
 - More complicated hardware = more power consumed, slower
- ❖ So we *combine* the two ideas:
 - Each address maps to exactly one **set**, each set can store block in some # of **ways**



Associativity (Review)

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 index into cache *sets*, of which there are $S = (C/K)/E$
 - Use lowest $\log_2(S) = \mathbf{s}$ bits of block address
 - Direct-mapped: $E = 1$, so $\mathbf{s} = \log_2(C/K)$ as we saw previously
 - Fully associative: $E = C/K$, so $\mathbf{s} = 0$ 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

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

m -bit address:	Tag (t)	Index (s)	Offset (k)
-------------------	-------------	---------------	----------------

$s = \underline{\hspace{1cm}}$
Direct-mapped

Set	Tag	Data
0		
1		
2		
3		
4		
5		
6		
7		

$s = \underline{\hspace{1cm}}$
2-way set associative

Set	Tag	Data
0		
1		
2		
3		

$s = \underline{\hspace{1cm}}$
4-way set associative

Set	Tag	Data
0		
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?
 - A. 2
 - B. 4
 - C. 8
 - D. 16
- ❖ If addresses are 16 bits wide, how wide is the Tag field?

Block Placement and Replacement (Review)

- ❖ 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?
 - 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			

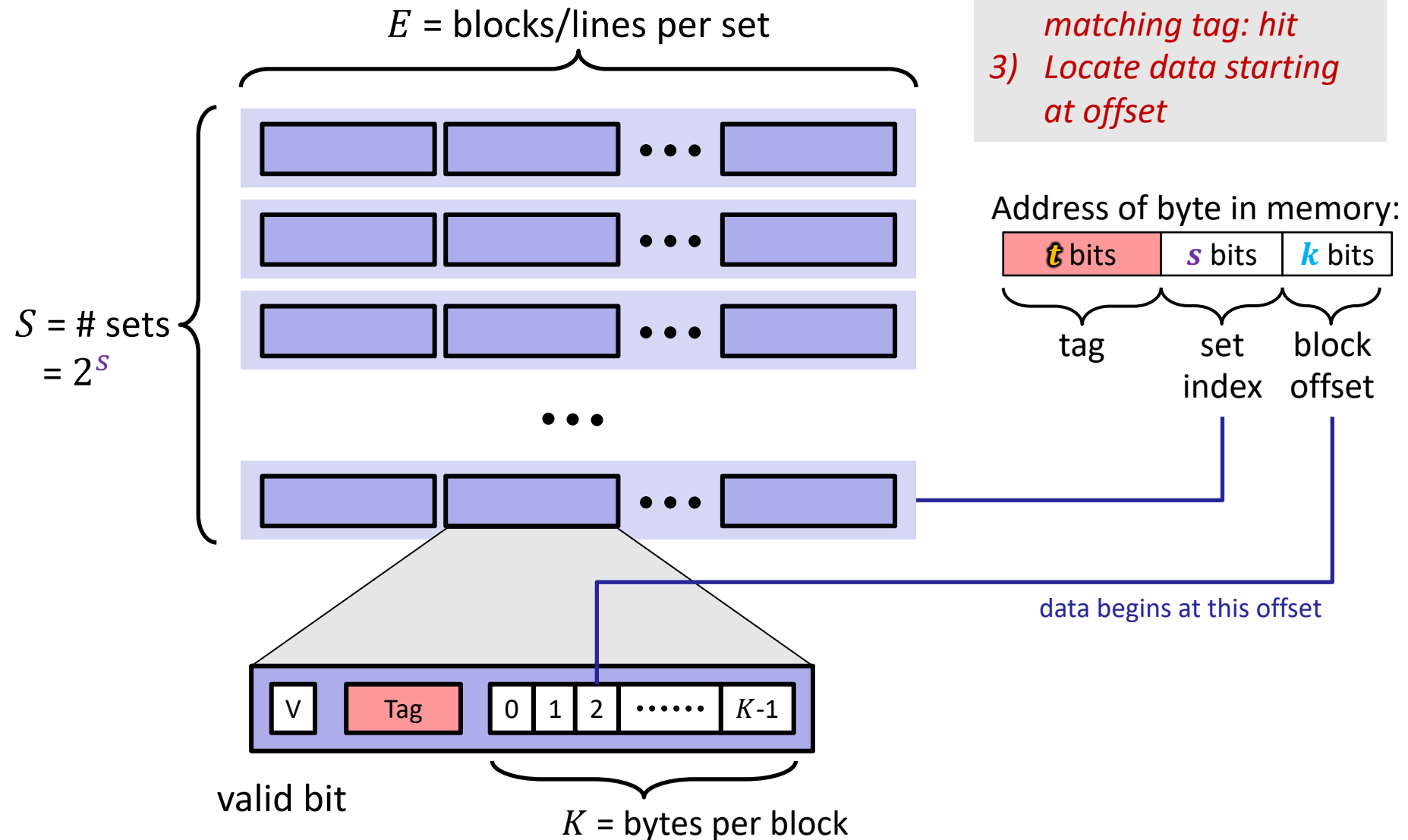
2-way set associative

Set	V	Tag	Data
0			
1			
2			
3			

4-way set associative

Set	V	Tag	Data
0			
1			

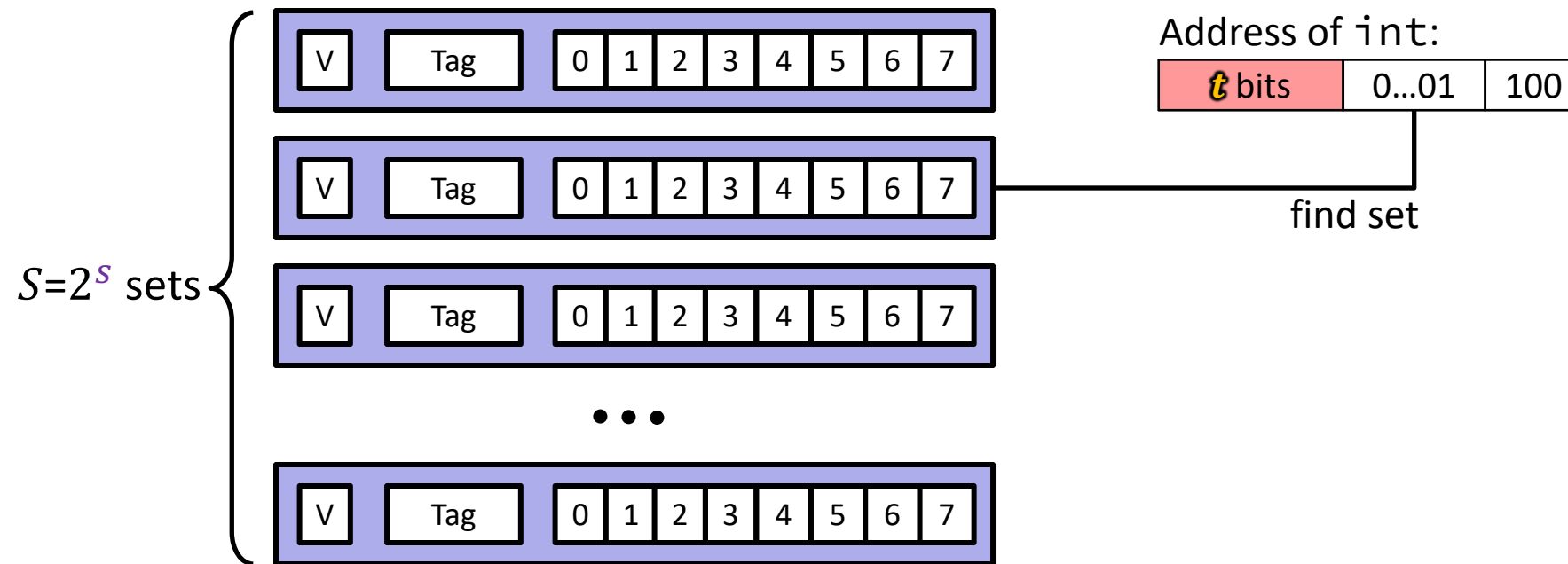
Cache Read Revisited



Example: Direct-Mapped ($E = 1$) Cache (1/3)

Direct-mapped: One line per set

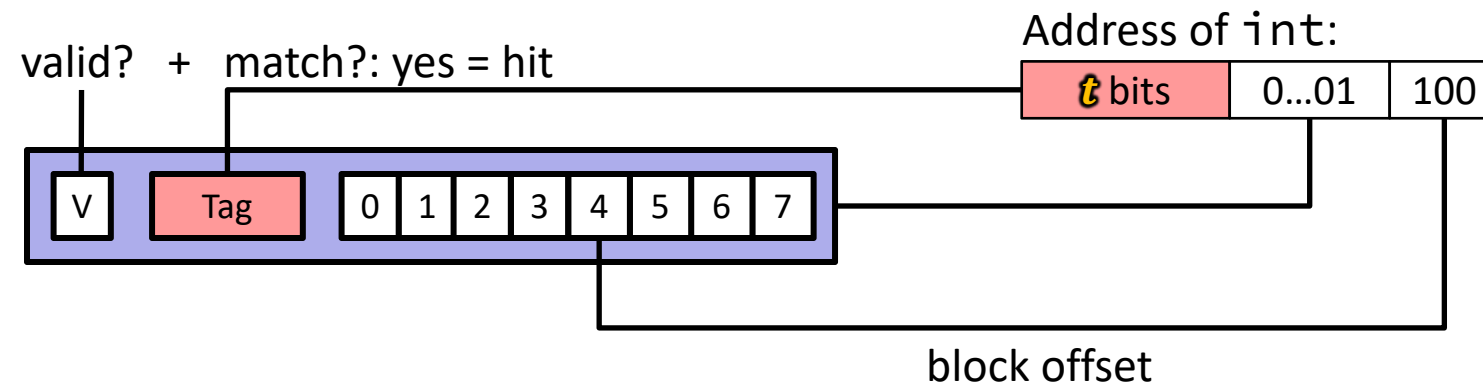
Block Size $K = 8$ B



Example: Direct-Mapped ($E = 1$) Cache (2/3)

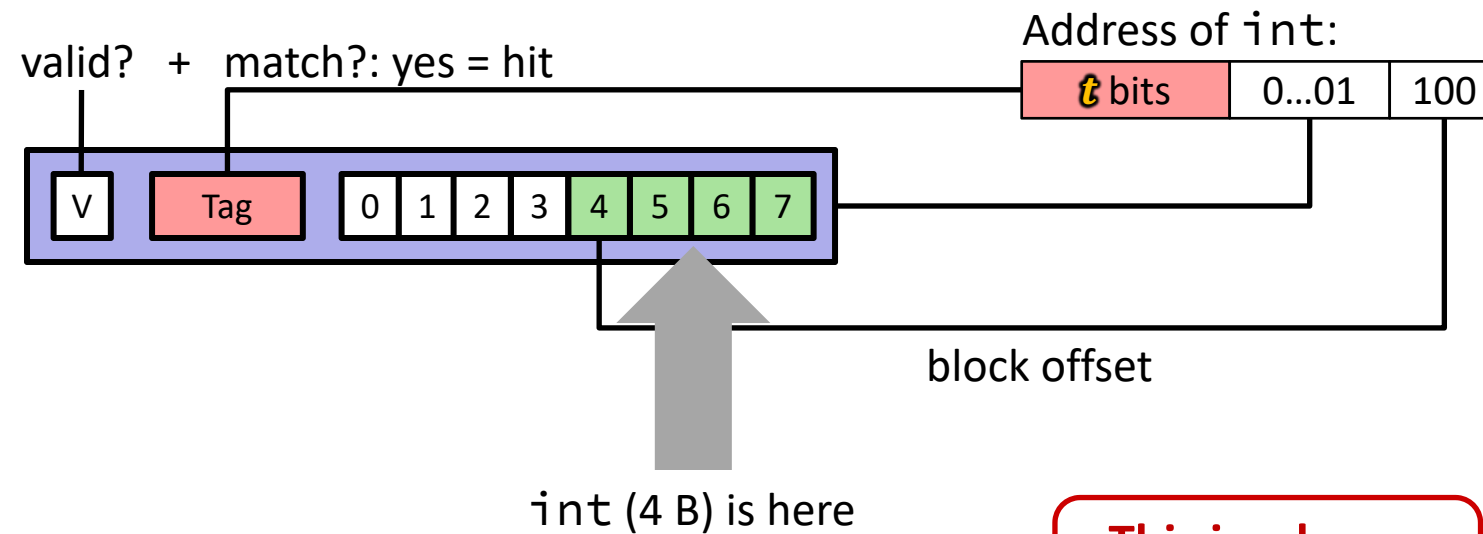
Direct-mapped: One line per set

Block Size $K = 8$ B



Example: Direct-Mapped ($E = 1$) Cache (3/3)

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



**This is why we
want alignment!**

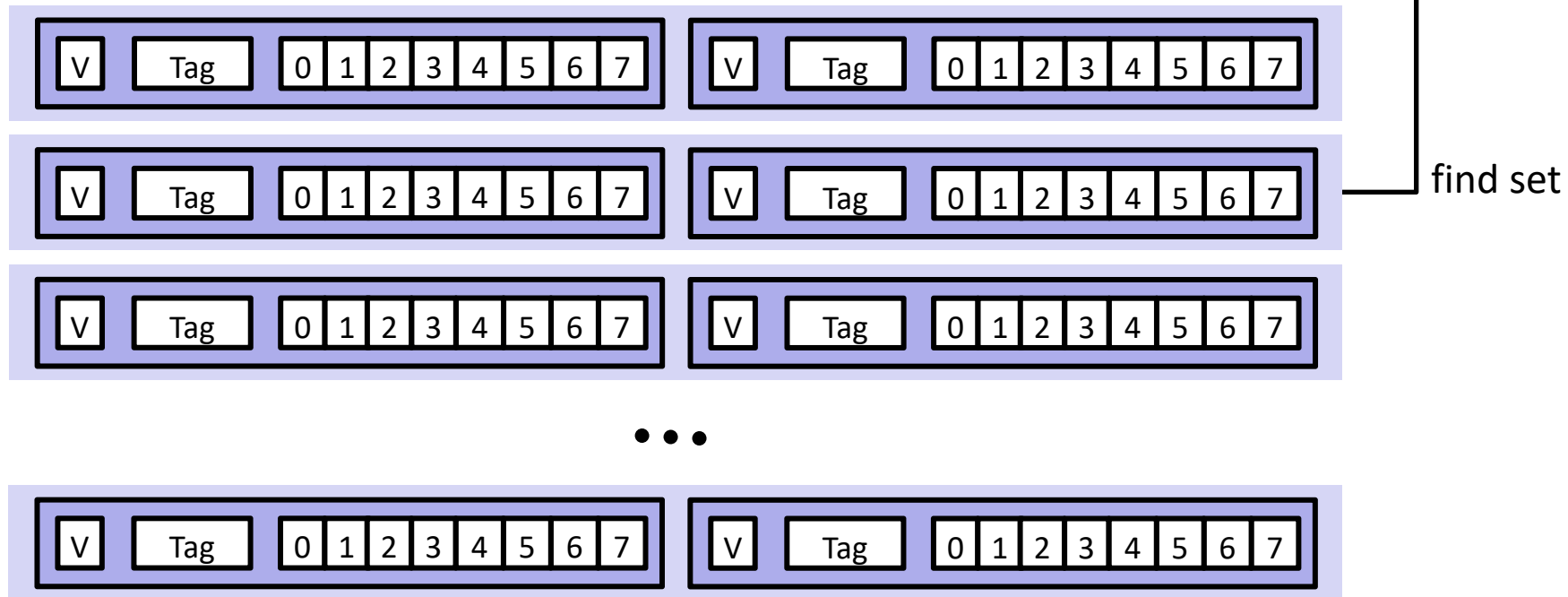
No match? Then old line gets evicted and replaced

Example: Set-Associative ($E = 2$) Cache (1/?)

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

Address of short:

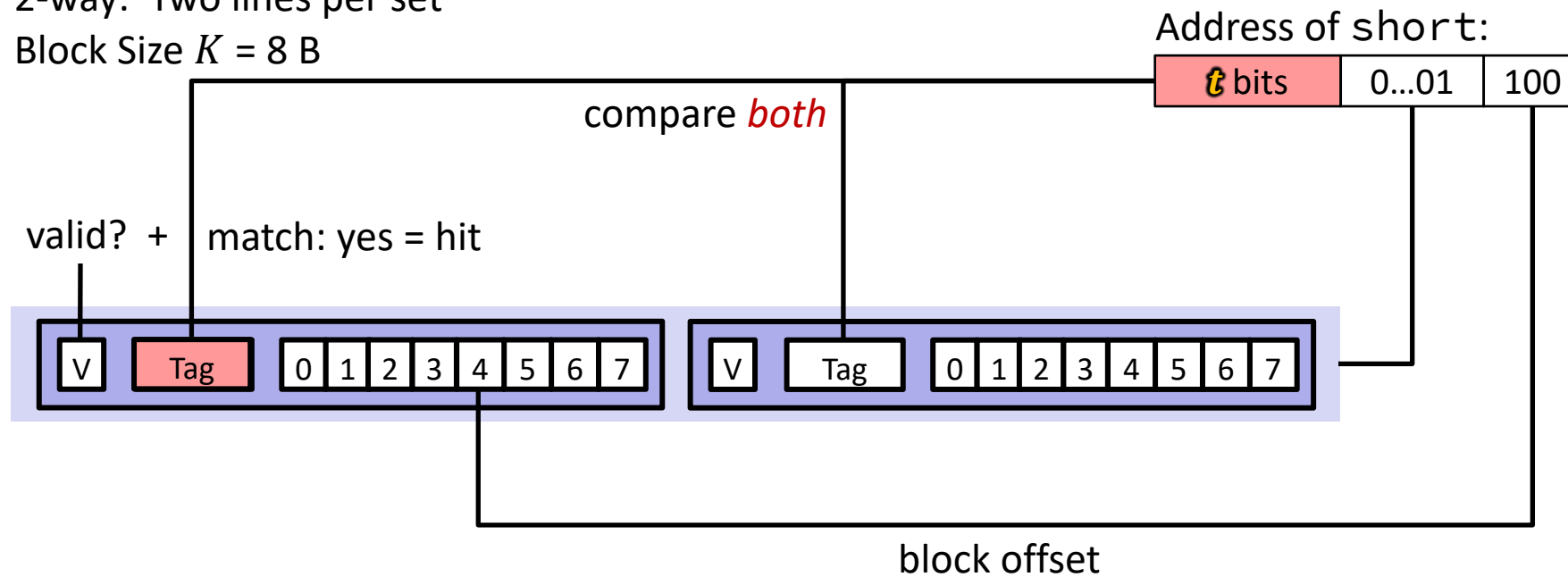
 bits	0...01	100
--	--------	-----



Example: Set-Associative ($E = 2$) Cache (2/3)

2-way: Two lines per set

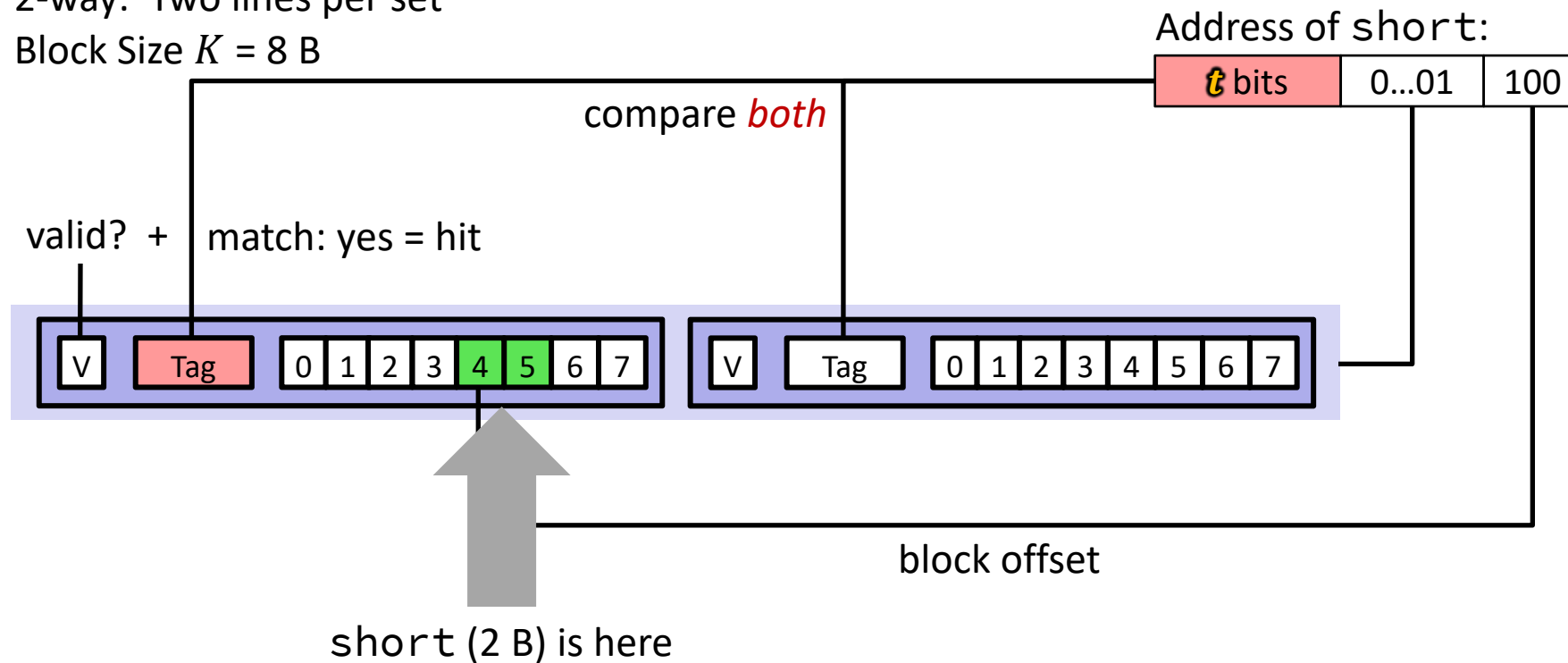
Block Size $K = 8$ B



Example: Set-Associative ($E = 2$) Cache (3/3)

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), ...

Lecture Outline (2/3)

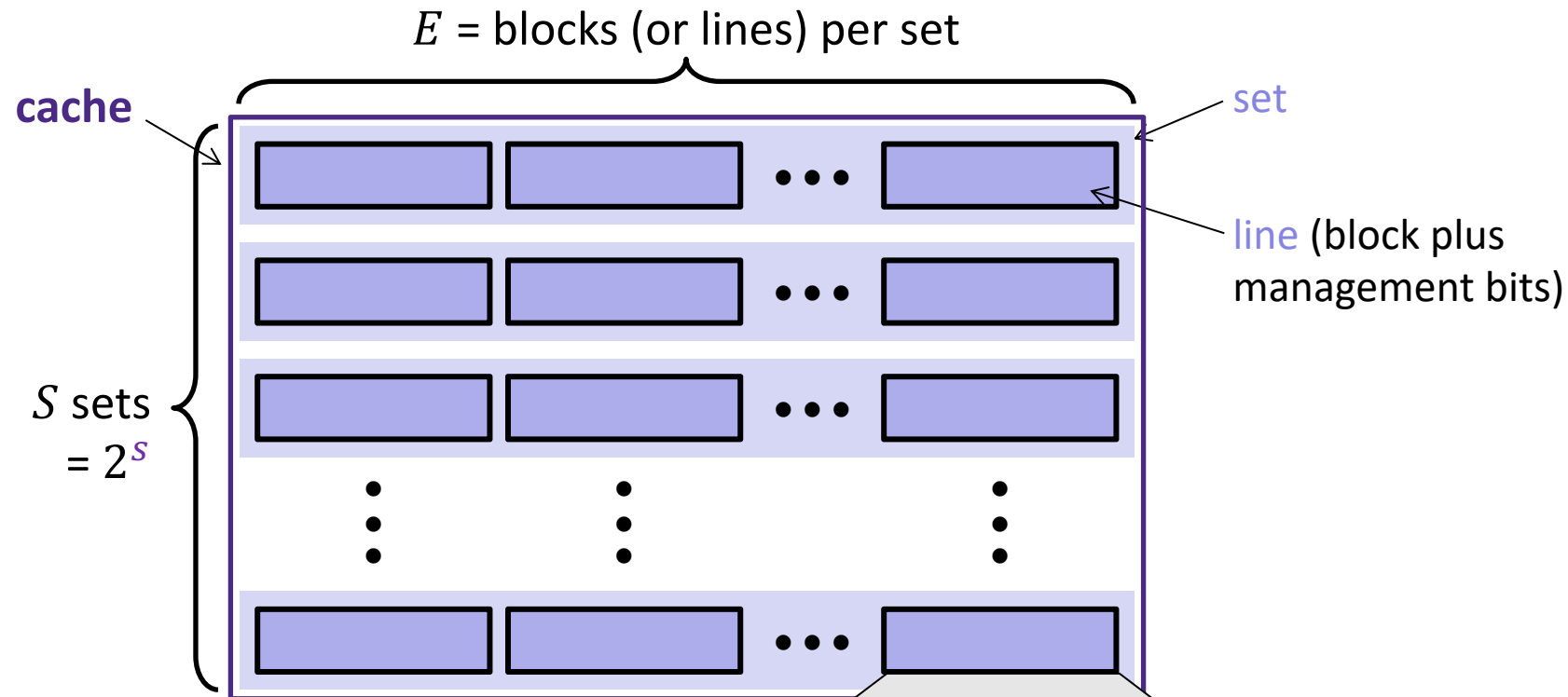
- ❖ Associativity and Replacement
- ❖ **Cache Organization**
- ❖ Example Cache Problems

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

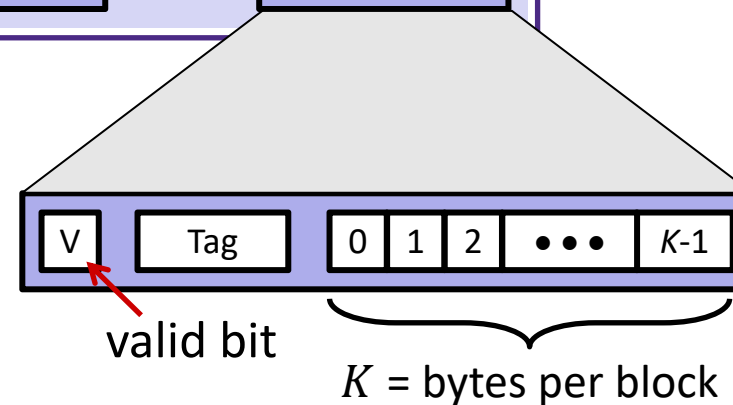
Parameter	Variable	Formulas
Block size	K (B in book)	$M = 2^m \leftrightarrow m = \log_2 M$ $S = 2^s \leftrightarrow s = \log_2 S$ $K = 2^k \leftrightarrow k = \log_2 K$ $C = K \times E \times S$ $s = \log_2(C/K/E)$ $m = t + s + k$
Cache size	C	
Associativity	E	
Number of Sets	S	
Address space	M	
Address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

General Cache Organization (S, E, K)



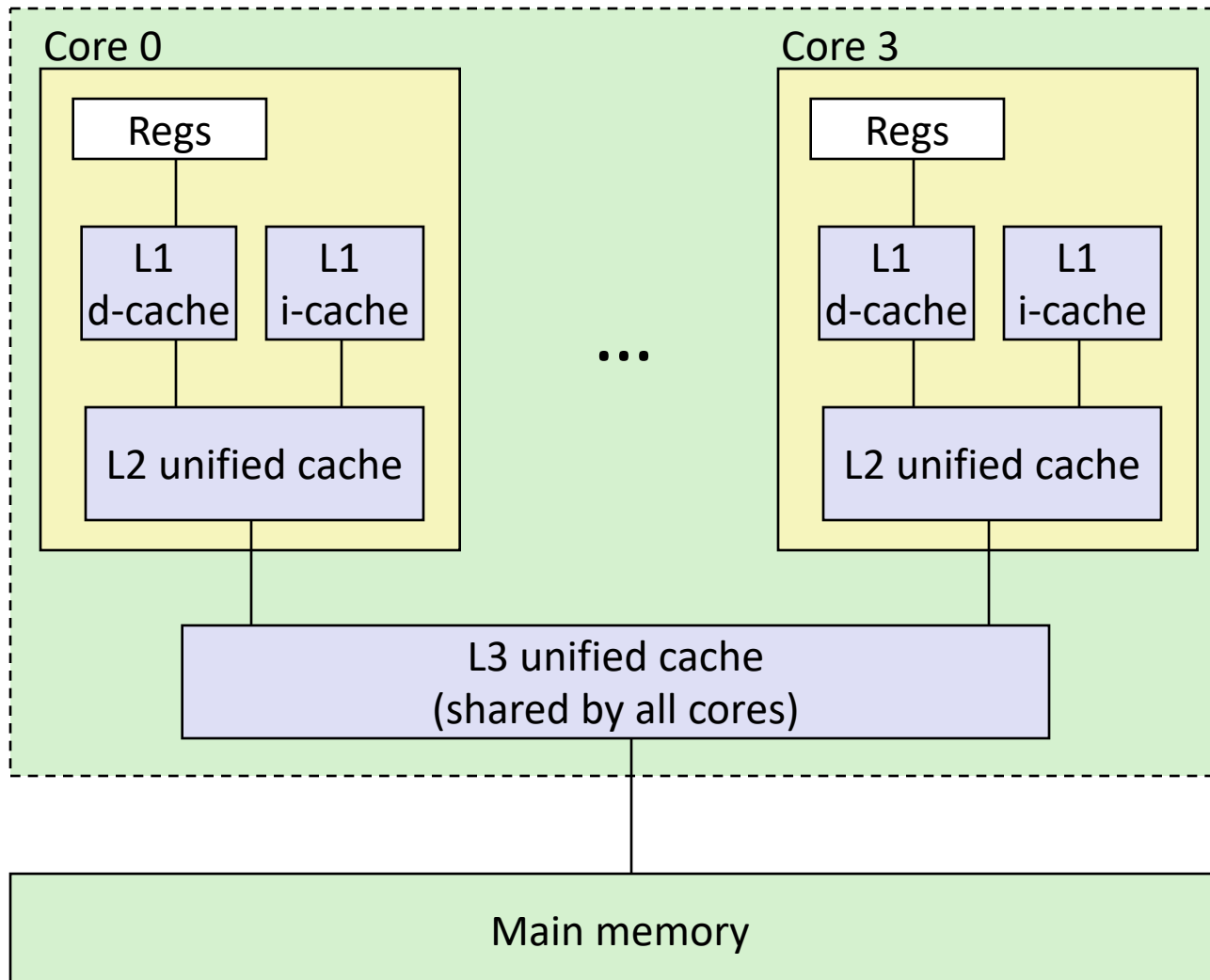
Cache size:

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



Intel Core i7 Cache Hierarchy

Processor package



Block size:

64 bytes for all caches

L1 i-cache and d-cache:

32 KiB, 8-way,
Access: 4 cycles

L2 unified cache:

256 KiB, 8-way,
Access: 11 cycles

L3 unified cache:

8 MiB, 16-way,
Access: 30-40 cycles

Learning About Your Machine

❖ Linux:

- `lscpu`
- `ls /sys/devices/system/cpu/cpu0/cache/index0/`
 - Example: `cat /sys/devices/system/cpu/cpu0/cache/index*/size`

❖ Windows:

- `wmic memcache get <query>` (all values in KB)
 - Example: `wmic memcache get MaxCacheSize`

❖ Mac:

- `sysctl` retrieves (⚠ and writes! ⚠) system information
 - Example: `sysctl hw.l1dcachesize` gets the size of the L1 d-cache in bytes

❖ Modern processor specs: <http://www.7-cpu.com/>

Lecture Outline (3/3)

- ❖ Associativity and Replacement
- ❖ Cache Organization
- ❖ **Example Cache Problems**

Example Cache Parameters Problem

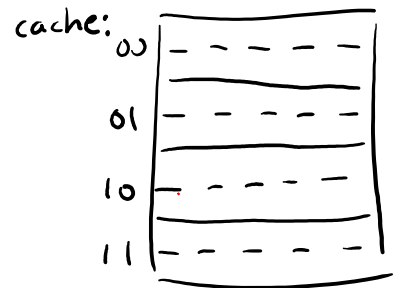
- ❖ 1 KiB address space, 125 cycles to go to memory.
Fill in the following table:

Cache Size	64 B
Block Size	8 B
Associativity	2-way
Hit Time	3 cycles
Miss Rate	20%
Tag Bits	
Index Bits	
Offset Bits	
AMAT	

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**:
 - $m = 10$ bits, $C = 64$ B, $K = 8$ B, $E = 2$

```
#define SIZE 8
short ar[SIZE][SIZE], sum = 0; // &ar=0x200
for (int i = 0; i < SIZE; i++)
    for (int j = 0; j < SIZE; j++)
        sum += ar[j][i];
```



Cache Simulator

- ❖ <https://courses.cs.washington.edu/courses/cse351/cachesim/>
 - From course website: Simulators → Cache Simulator
 - Allows you to play around with the effects of cache parameters and policies
 - Lots of neat features like highlighting, hover text, ability to rewind and replay accesses, and copy-and-paste access patterns

- ❖ Ways to use:
 - Take advantage of “explain mode” and navigable history to test your own hypotheses and answer your own questions
 - A [tutorial Ed lesson](#) is available
 - Will be used in HW19 – Lab 4 Preparation

Cache Simulator Demo

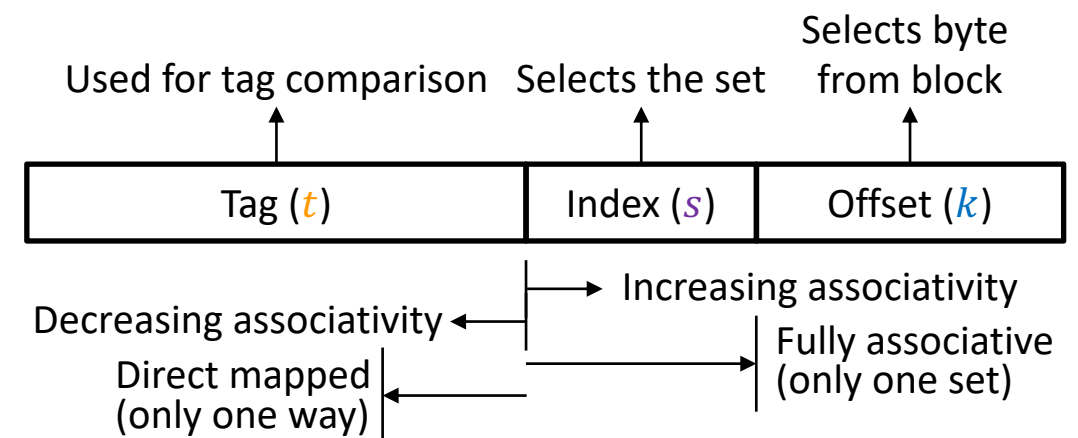
- ❖ <https://courses.cs.washington.edu/courses/cse351/cachesim/>
 - From course website: Simulators → Cache Simulator
 - Allows you to play around with the effects of cache parameters and policies
 - Lots of neat features like highlighting, hover text, ability to rewind and replay accesses, and copy-and-paste access patterns
- ❖ Let's simulate the example code analysis problem:

```
#define SIZE 8
short ar[SIZE][SIZE], sum = 0; // &ar=0x200
for (int i = 0; i < SIZE; i++)
    for (int j = 0; j < SIZE; j++)
        sum += ar[j][i];
```

Summary (1/2)

- ❖ **Associativity** gives us flexibility in where to place blocks in the cache
 - Group E slots into **sets**, means there are E **ways** to place block within each set
 - Direct-mapped is $E = 1$, **fully associative** is $E = \#$ of slots in cache (*i.e.*, $S = 1$)
 - Helps avoid conflicts in each set at the expense of slightly longer and more complex searching & placing in the cache
 - By default, we will replace the **least-recently used** block in a set
 - Index $\rightarrow$ Set, $S = (C/K)/E$, still $s = \log_2(S)$

Direct-mapped			2-way set associative			4-way set associative		
Set	Tag	Data	Set	Tag	Data	Set	Tag	Data
0			0			0		
1			1					
2								
3								
4			2			1		
5								
6								
7			3					



Summary (2/2)

❖ Management bits

- Information needed for proper management of the cache & its data, but not counted in the cache size
- **Valid bit** for validity of data
- **Tag bits** for identifying which block

