Memory & Caches II

CSE 351 Winter 2021

Instructor:

Mark Wyse

Teaching Assistants:

Kyrie Dowling

Catherine Guevara

Ian Hsiao

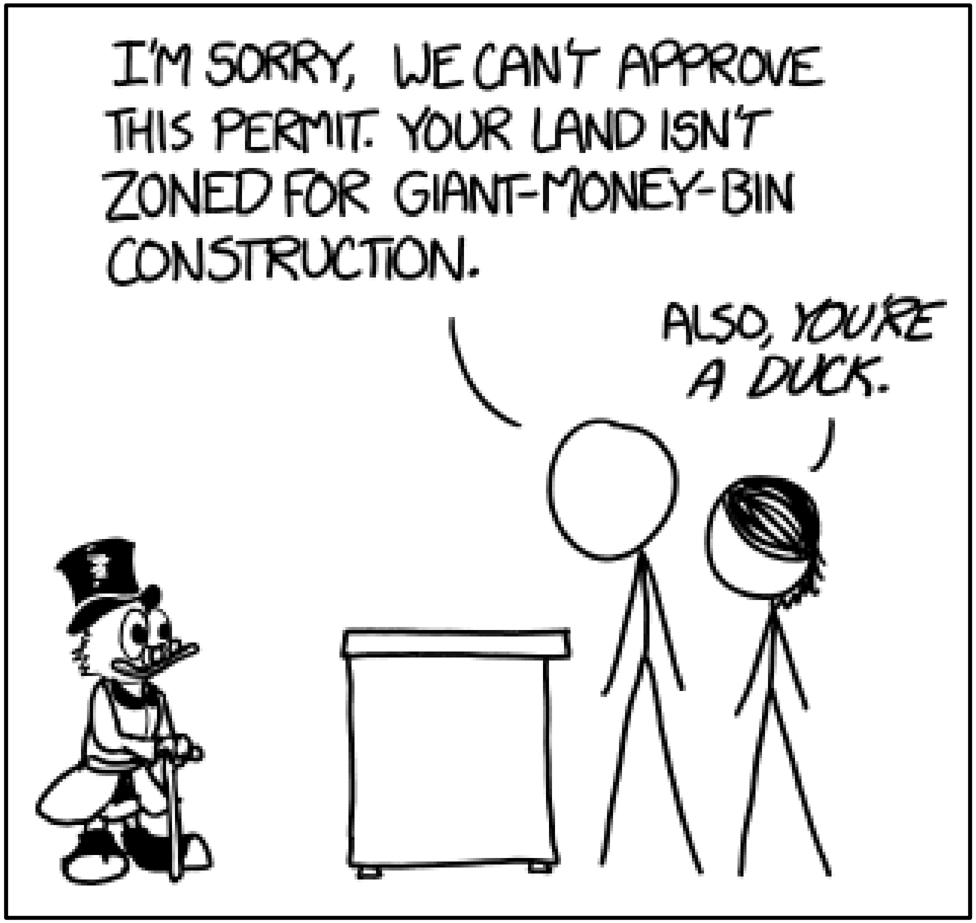
Jim Limprasert

Armin Magness

Allie Pfleger

Cosmo Wang

Ronald Widjaja


I'M SORRY, WE CAN'T APPROVE THIS PERMIT. YOUR LAND ISN'T ZONED FOR GIANT-MONEY-BIN CONSTRUCTION.
ALSO, YOU'RE A DUCK.

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

Administrivia

- No class on Monday! (2/15)

- Mid-quarter Survey due Friday (2/19)

- hw15 due Wednesday (2/17)
- hw16 due Monday (2/22)
  - Don’t wait too long, this is a BIG hw

- Lab 3 due Monday (2/22)

Growth vs. Fixed Mindset

- Students can be thought of as having either a “growth” mindset or a “fixed” mindset (based on research by Prof. Carol Dweck)
  - “In a **fixed mindset** students believe their basic abilities, their intelligence, their talents, are just fixed traits. They have a certain amount and that's that, and then their goal becomes to look smart all the time and never look dumb.”
  - “In a **growth mindset** students understand that their talents and abilities can be developed through effort, good teaching and persistence. They don't necessarily think everyone's the same or anyone can be Einstein, but they believe everyone can get smarter if they work at it.”

An Example Memory Hierarchy

↑ **Smaller, faster, costlier per byte**

↓ **Larger, slower, cheaper per byte**

<1 ns **registers** 8-32 5-10 s

1 ns **on-chip L1 cache (SRAM)**

5-10 ns on **~~off~~-chip L2 cache (SRAM)** 1-2 min

L3

100 ns **main memory (DRAM)** GB-TB 15-30 min

150,000 ns **SSD** 31 days

10,000,000 ns *(10 ms)* **Disk** 66 months = 5.5 years

local secondary storage (local disks) TB-PB

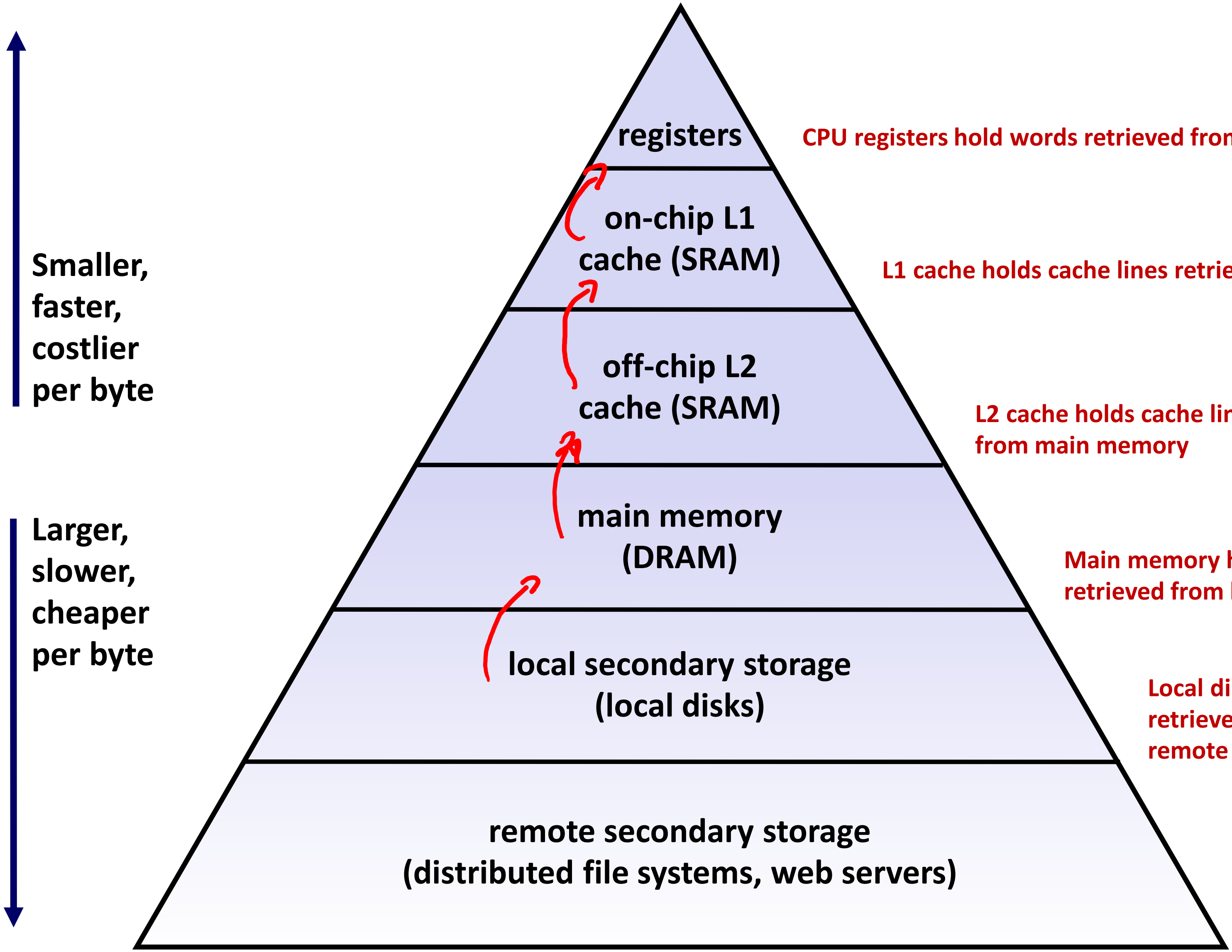
1-150 ms **remote secondary storage (distributed file systems, web servers)** 1 - 15 years

Memory Hierarchies

- Some fundamental and enduring properties of hardware and software systems:
  - Faster storage technologies almost always cost more per byte and have lower capacity
  - The gaps between memory technology speeds are widening
    - True for: registers ↔ cache, cache ↔ DRAM, DRAM ↔ disk, etc.
  - Well-written programs tend to exhibit good locality

- These properties complement each other beautifully
  - They suggest an approach for organizing memory and storage systems known as a memory hierarchy
    - For each level k, the faster, smaller device at level k serves as a cache for the larger, slower device at level k+1

An Example Memory Hierarchy



CPU registers hold words retrieved from L1 cache

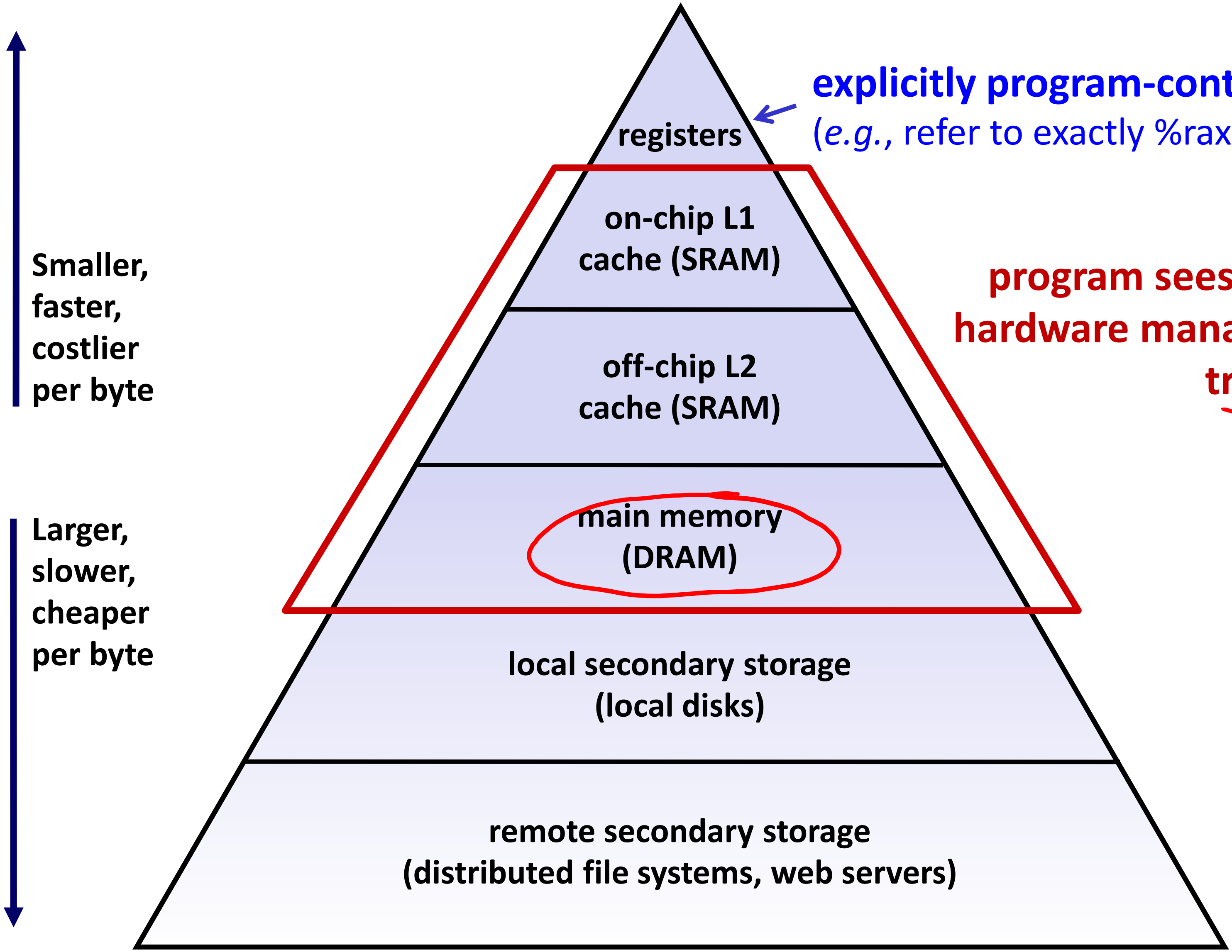
L1 cache holds cache lines retrieved from L2 cache

L2 cache holds cache lines retrieved from main memory

Main memory holds disk blocks retrieved from local disks

Local disks hold files retrieved from disks on remote network servers

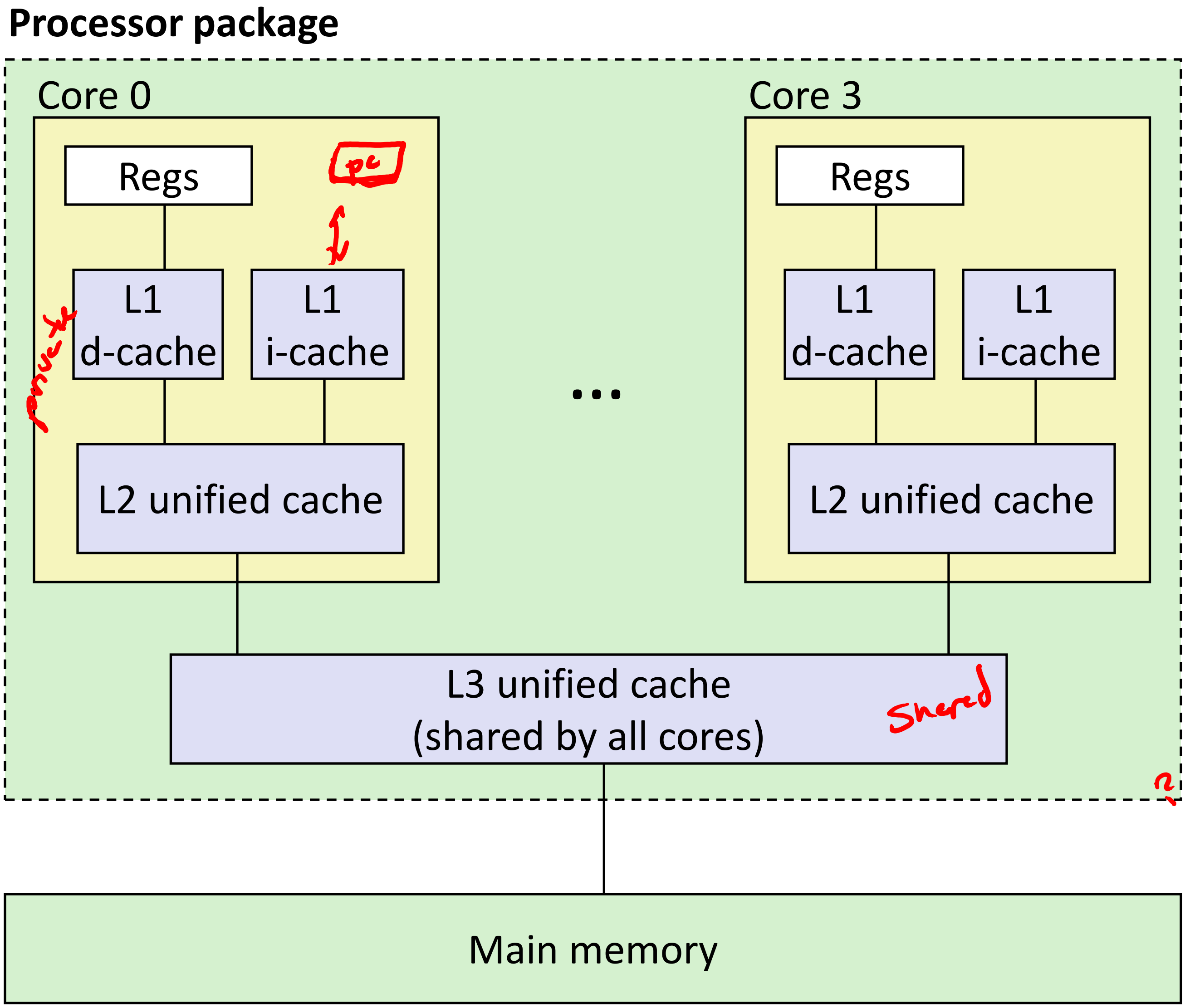
An Example Memory Hierarchy



explicitly program-controlled
(*e.g.*, refer to exactly %rax, %rbx)

**program sees "memory";
hardware manages caching
transparently**

Intel Core i7 Cache Hierarchy



Block size:
64 bytes for all caches

L1 i-cache and d-cache:
32 KiB, 8-way,
Access: 4 cycles

L2 unified cache:
x8
256 KiB, 8-way,
Access: 11 cycles

L3 unified cache:
? x32
8 MiB, 16-way,
Access: 30-40 cycles

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:
  - Memory hierarchy
  - Cache parameters: block size (K), cache size (C)
  - Addresses: block offset field (k bits wide)
  - Cache organization: direct-mapped cache, index field

- Questions from the Reading?

Review Questions

- We have a direct-mapped cache with the following parameters:
  - Block size of 8 bytes $= 2^3$ B $k = \log_2 K = \log_2 2^3 = 3$
  - Cache size of 4 KiB $2^2 2^{10} = 2^{12}$ B
- How many blocks can the cache hold? $= \frac{C}{K} = \frac{2^{12}}{2^3} = 2^9 = 512$
- How many bits wide is the block offset field? 3 bits
- Which of the following addresses would fall under block number 3?

A. 0x3 $\lfloor \frac{3}{8} \rfloor = \emptyset$
0b 000/011
block Ø

$\boxed{\text{B. 0x1F}}$ $\lfloor \frac{31}{8} \rfloor = 3$
0b 01 1/111
block = 3

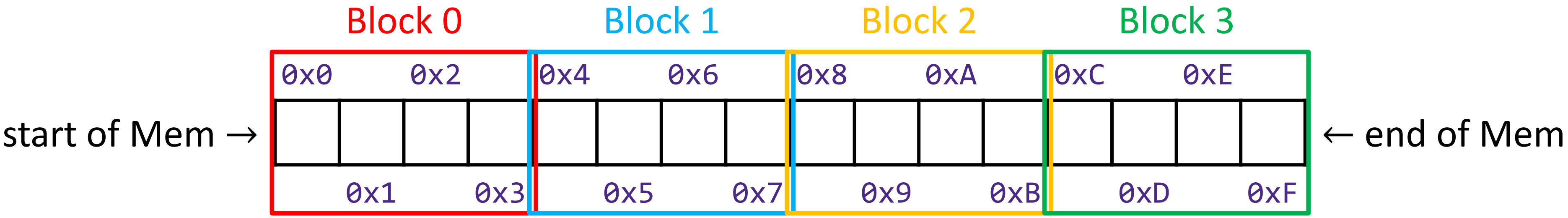
C. 0x30
0b 110/000
block 6

D. 0x38
0b 111/000
block 7

Cache Organization (1)

Note: The textbook uses “B” for block size

- Block Size (K): unit of transfer between \$ and Mem
  - Given in bytes and always a power of 2 (*e.g.*, 64 B)
  - Blocks consist of adjacent bytes (differ in address by 1)
    - Spatial locality!
  - Small example ($K = 4$ B):



Block 0 Block 1 Block 2 Block 3

start of Mem → 0x0 0x1 0x2 0x3 | 0x4 0x5 0x6 0x7 | 0x8 0x9 0xA 0xB | 0xC 0xD 0xE 0xF ← end of Mem

addr: 0-3 4-7 8-11 12-15

Cache Organization (1)

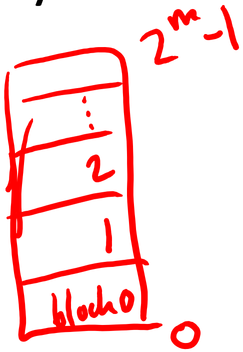
Note: The textbook uses "B" for block size

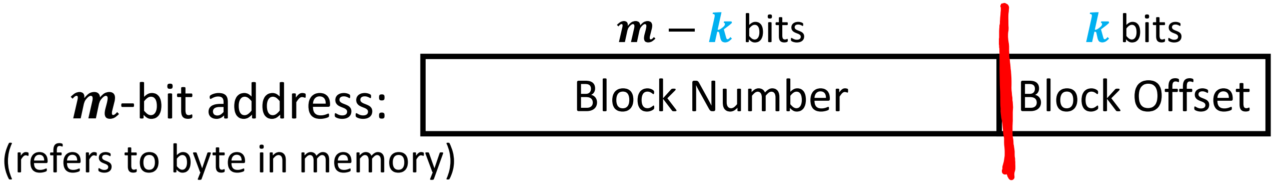
- Block Size (K): unit of transfer between $ and Mem
  - Given in bytes and always a power of 2 (*e.g.*, 64 B)
  - Blocks consist of adjacent bytes (differ in address by 1)
    - Spatial locality!

Cache Organization (1)

Note: The textbook uses "b" for offset bits

- Block Size (K): unit of transfer between \$ and Mem
  - Given in bytes and always a power of 2 (*e.g.*, 64 B)
  - Blocks consist of adjacent bytes (differ in address by 1)
    - Spatial locality!
- Offset field
  - Low-order $\boxed{\log_2(K) = \boldsymbol{k}}$ bits of address tell you which byte within a block
    - (address) mod 2^n = n lowest bits of address
  - (address) modulo (# of bytes in a block)




$\boldsymbol{m}-\boldsymbol{k}$ bits

$\boldsymbol{k}$ bits

$\boldsymbol{m}$-bit address:
(refers to byte in memory)

Block Number	Block Offset

Cache Organization (1)

Note: The textbook uses “b” for offset bits

- Block Size (K): unit of transfer between $ and Mem
  - Given in bytes and always a power of 2 (*e.g.*, 64 B)
  - Blocks consist of adjacent bytes (differ in address by 1)
    - Spatial locality!

- Example:
  - If we have 6-bit addresses and block size K = 4 B, which block and byte does `0x15` refer to?

$m=6$

$K=4 \rightarrow k=\log_2 4=2$

m-k	k
4	2

0x15

0b 01 01 | 01

m-k | k

block = 5

byte = 1

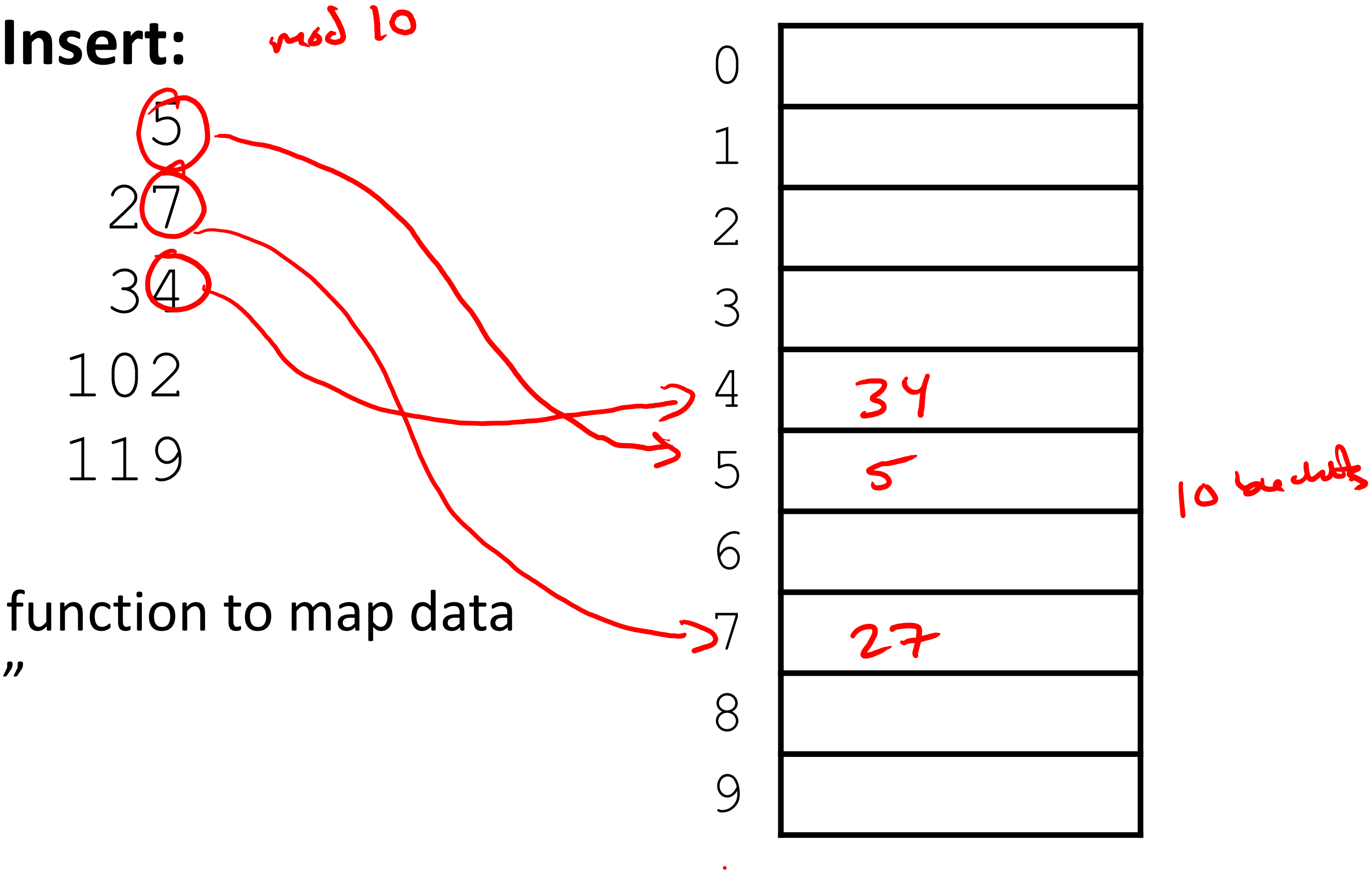
Cache Organization (2)

- Cache Size (C): amount of *data* the $ can store
  - Cache can only hold so much data (subset of next level)
  - Given in bytes (C) or number of blocks (C/K)
  - Example: C = 32 KiB = 512 blocks if using 64-B blocks

$2^5 2^{10} = 2^{15} / 2^6 = \frac{C}{K} = 2^9$ blocks = 512

- Where should data go in the cache?
  - We need a mapping from memory addresses to specific locations in the cache to make checking the cache for an address **fast**
- What is a data structure that provides fast lookup?
  - Hash table!

Review: Hash Tables for Fast Lookup

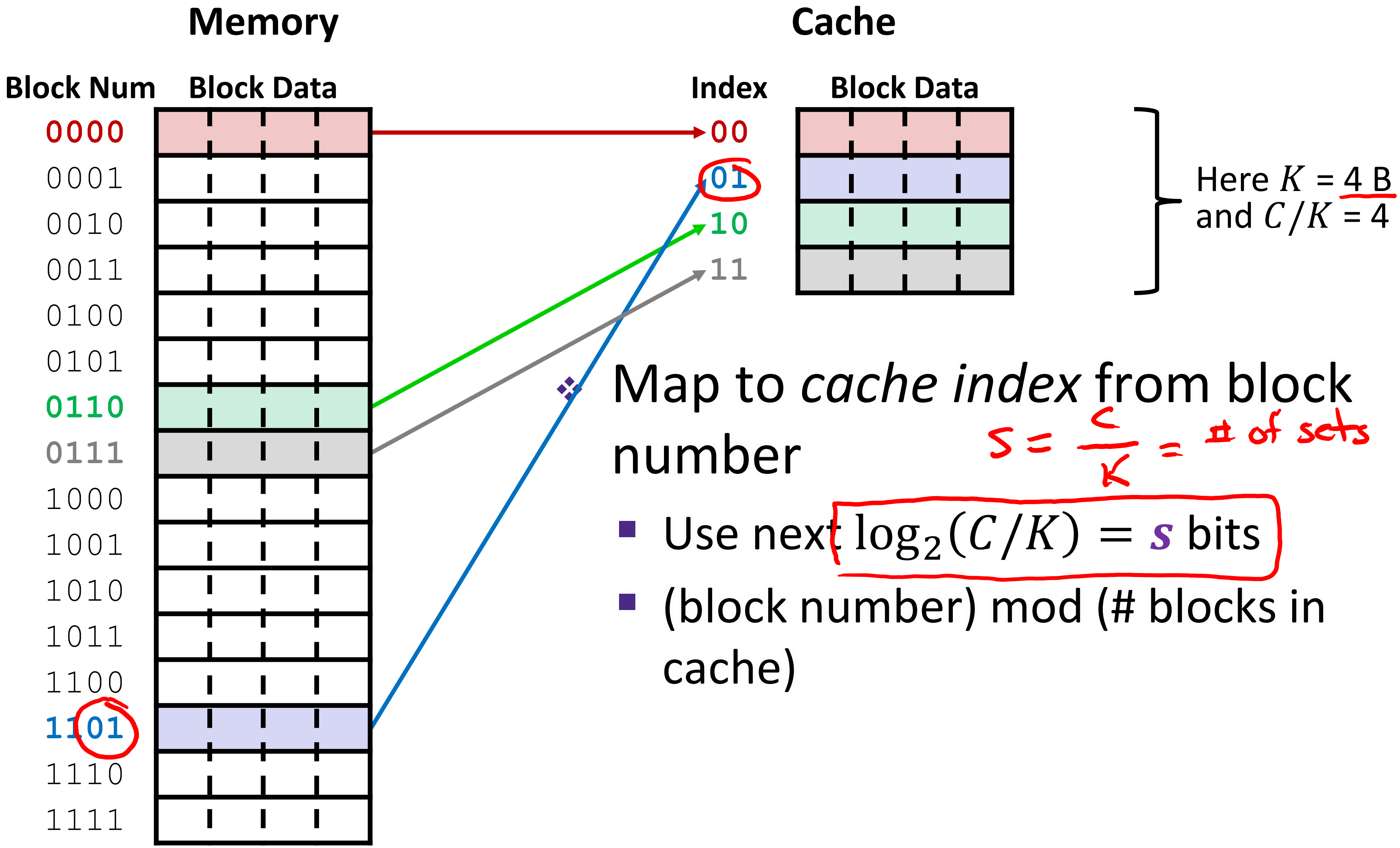


Insert:

5
27
34
102
119

Apply hash function to map data to “buckets”

Place Data in Cache by Hashing Address

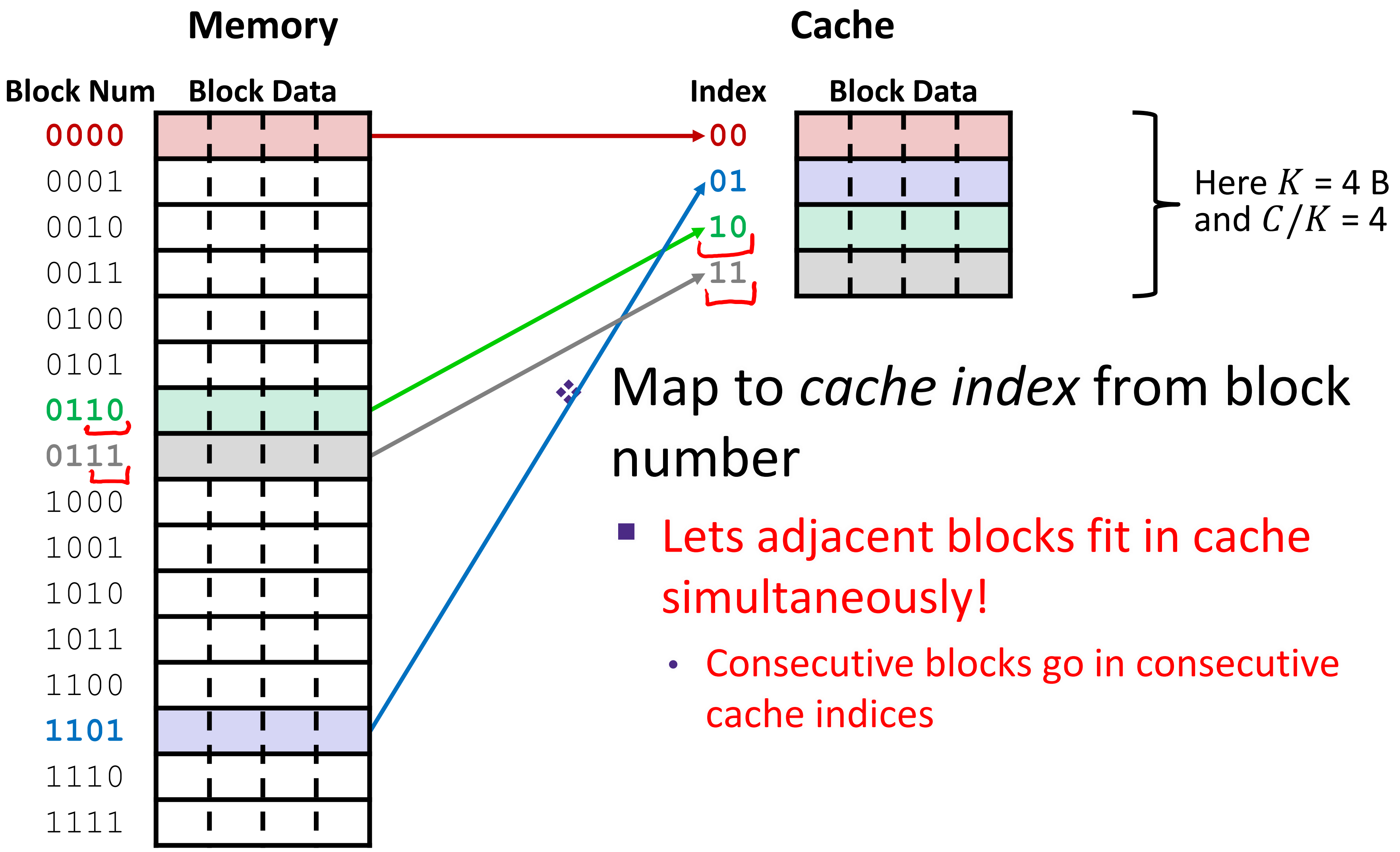


Here K = 4 B
and C/K = 4

Map to *cache index* from block number $s = \frac{C}{K} = \#$ of sets

- Use next $\boxed{\log_2(C/K) = \boldsymbol{s} \text{ bits}}$
- (block number) mod (# blocks in cache)

Place Data in Cache by Hashing Address



Here K = 4 B and C/K = 4

- Map to *cache index* from block number
  - Lets adjacent blocks fit in cache simultaneously!
    - Consecutive blocks go in consecutive cache indices

Polling Question

$k = \log_2 K$

- 6-bit addresses, block size K = 4 B, and our cache holds S = 4 blocks. $m = 6$, $k = 2$, $s = \log_2 S = 2$
- A request for address **0x2A** results in a cache miss. Which index does this block get loaded into and which 3 other addresses are loaded along with it?
  - Vote on Ed Lessons

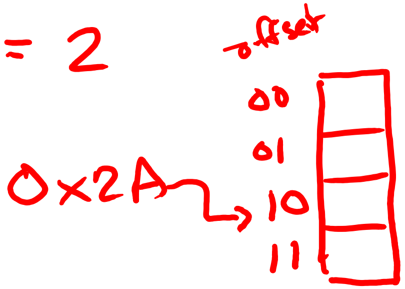
tag	index	offset
2	2	2

6

$0x2A$

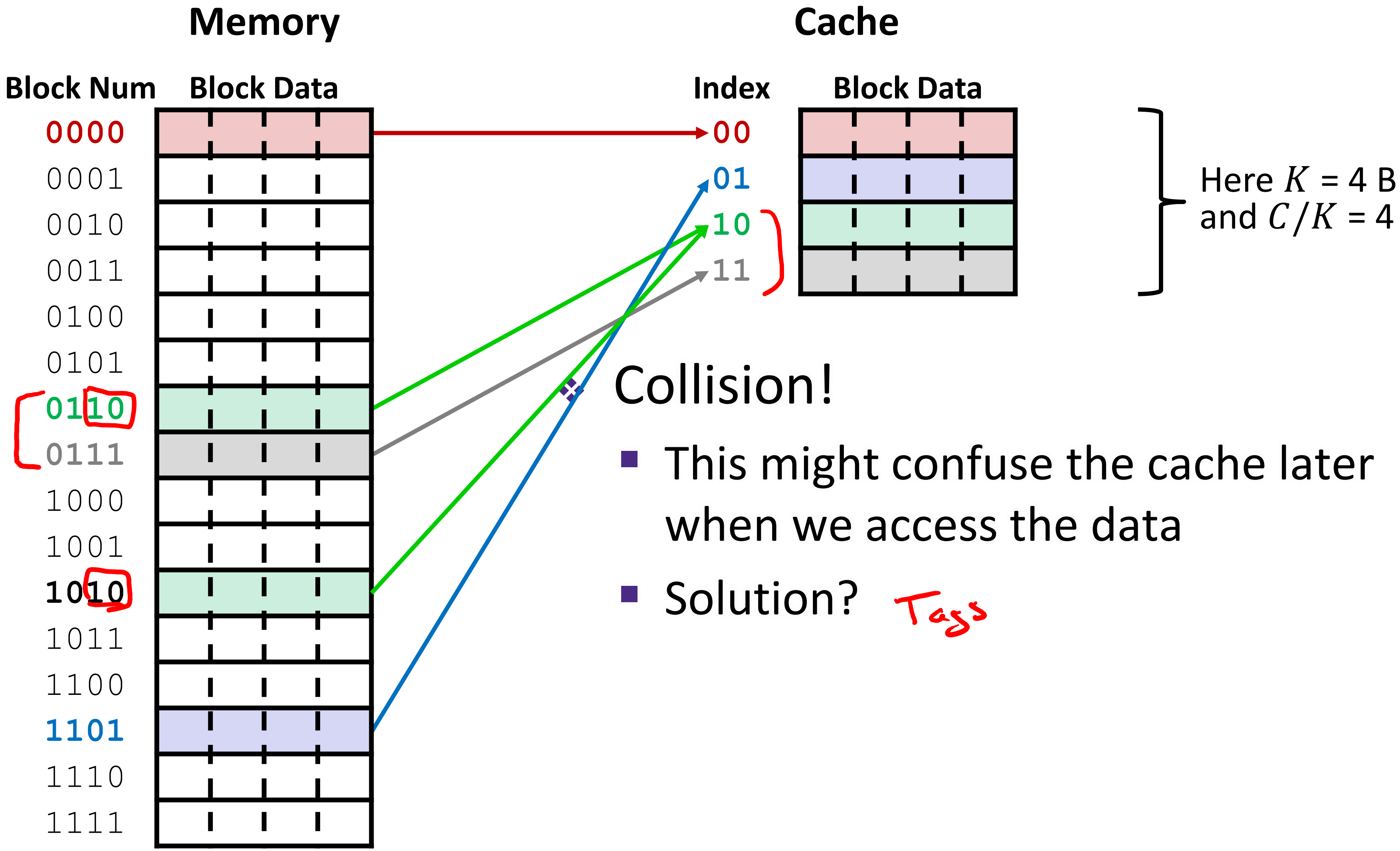
$= 0b10\underbrace{10}_{\substack{\text{index}\\ 2}}\underbrace{10}_{\substack{\text{offset}\\ 2}}$

index = 2



0x28
0x29
0x2A
0x2B

Place Data in Cache by Hashing Address

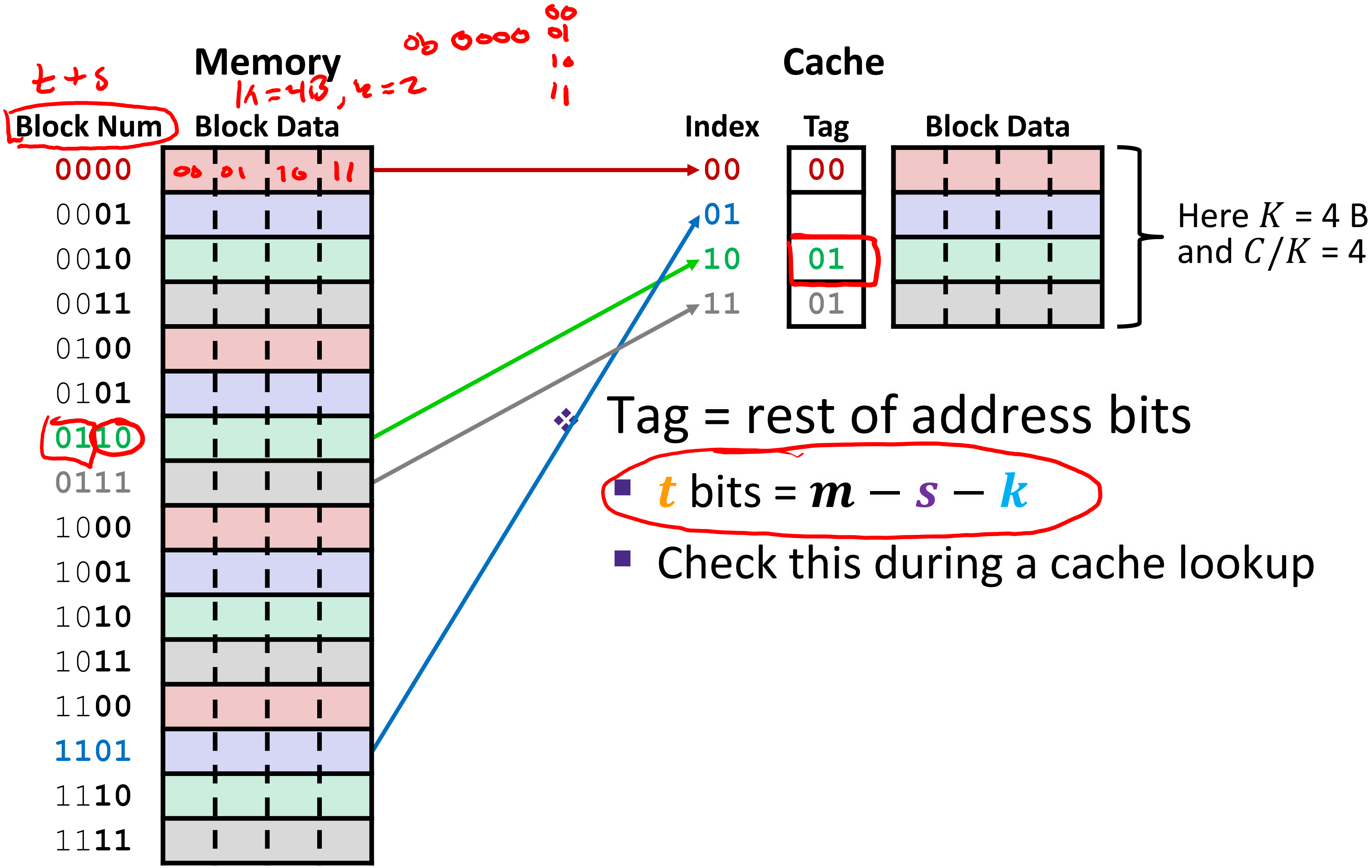


Here K = 4 B and C/K = 4

Collision!

- This might confuse the cache later when we access the data
- Solution? Tags

Tags Differentiate Blocks in Same Index



Memory

Block Num Block Data

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

Cache

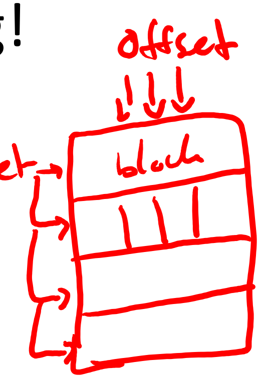
Index	Tag	Block Data
00	00	
01		
10	01	
11	01	

Here K = 4 B and C/K = 4

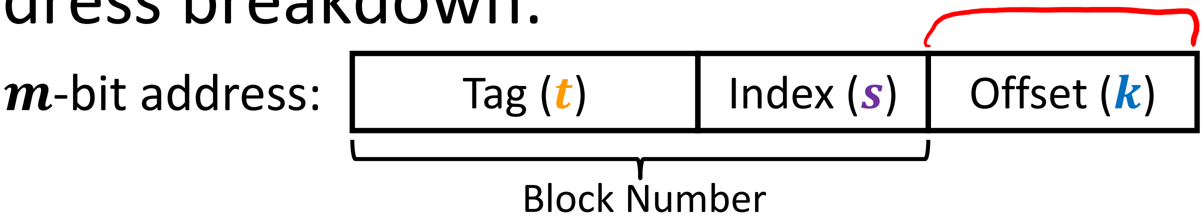
- Tag = rest of address bits
  - t bits = $m - s - k$
  - Check this during a cache lookup

Checking for a Requested Address

- CPU sends address request for chunk of data
  - Address and requested data are not the same thing!
    - Analogy: your friend ≠ their phone number



- TIO address breakdown:



  - **Index** field tells you where to look in cache
  - **Tag** field lets you check that data is the block you want
  - **Offset** field selects specified start byte within block

  - **Note:** t and s sizes will change based on hash function

Cache Puzzle

- Based on the following behavior, which of the following block sizes is NOT possible for our cache?
  - Cache starts *empty*, also known as a *cold cache*
  - Access (addr: hit/miss) stream:
    - (14: miss), (15: hit), (16: miss)

A. 4 bytes

B. 8 bytes

C. 16 bytes

[✓ D. 32 bytes]

E. We're lost…

Explanation:
Miss to 14 loads a cache block. For block sizes of 4, 8, and 16 bytes, this miss loads both addresses 14 and 15, but not 16. Thus, the access to 15 is a hit while the access to 16 is a miss.

A block size of 32 bytes would load addresses 0-31, and the access to 16 would be a hit. Therefore, the listed sequence is not possible with 32B blocks.