Lecture 22: Caching in

CMPE 120: Computer Organization & Architecture

Olivia Weng

Logistics

- Reminder: check-in with professor is 1% of your final grade
  - Sign up link
- Grade remediation plan
  - 30+ students have already signed up
  - Slots are getting booked quickly → sign up soon!
- HW4 due this today (+1 late day). HW4 Quiz next Tuesday.

Grade remediation plan

- HW1 Quiz Remediation: meet with professor for 15 minutes
  - Bring a newly hand-written solution. Explain what you got wrong on your quiz and your new solution.
  - Replaces HW1 quiz grade & drops lowest HW Quiz grade
  - Schedule a time on my calendar in the next ~3 weeks (10/30 - 11/20)
    - Link: https://calendar.app.google/d3D4hm3JbmpXcrVt5
    - 100 appointment slots available for 67 students
    - If none of these times work, email me to schedule a time during this window.

- Final exam grade can replace the midterm grade if higher
  - → Midterm grade replacement = max(final exam grade, midterm exam grade)

Ideal memory system

- We want a large and fast memory
  - But, we cannot have large and fast memory

- We have
  - small and fast memory
  - large and slow memory
  - speeds/sizes in between



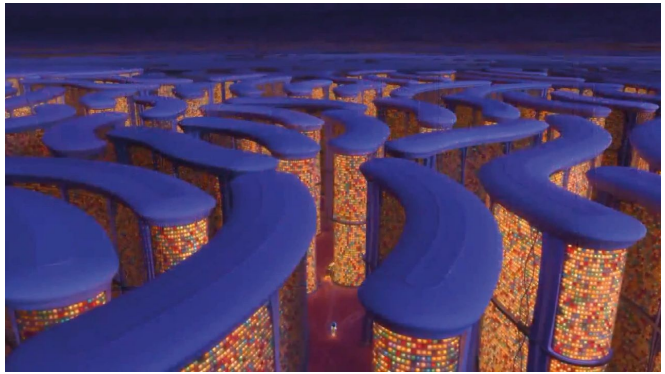
- With memory locality, we can make this work

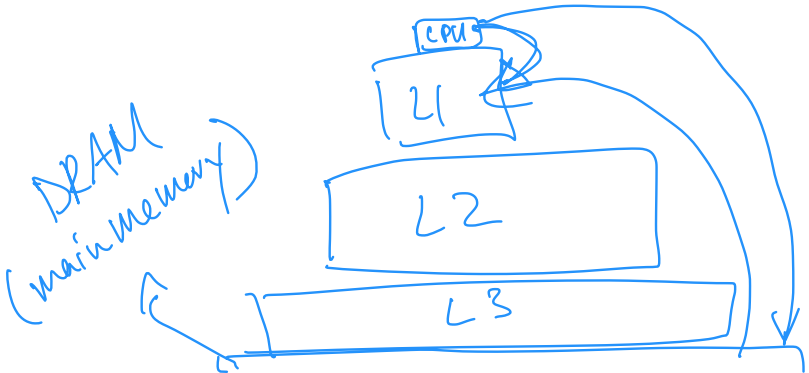
Cache examples

- What are some examples?



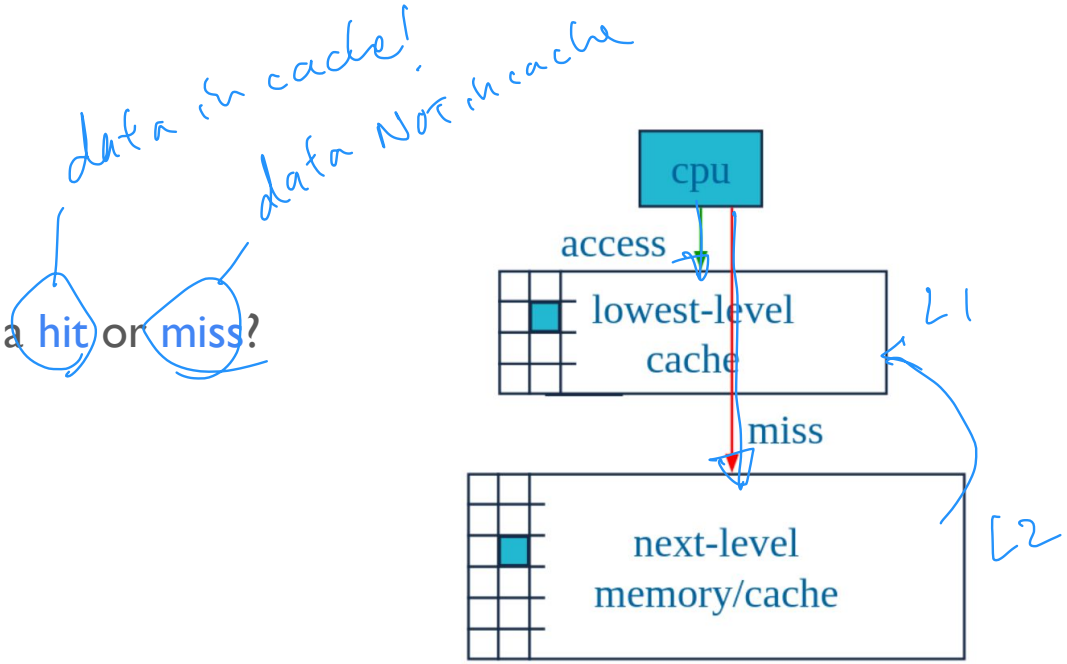






Caching issues

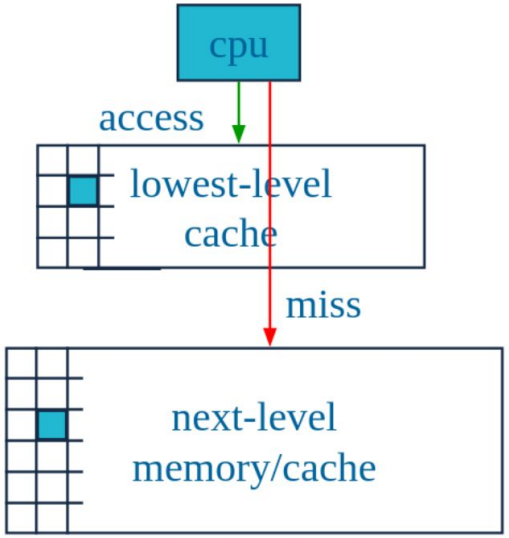
- On memory access
  - How do I know if this is a hit or miss?



Caching issues

- On memory access
  - How do I know if this is a hit or miss?

- On cache miss
  - Where to put the new data?
  - What data to kick out? – replacement policy
    - Typically select the Least Recently Used (LRU)
  - How to keep track of what data this is?



Cache design

- How to map data chunks from main memory to blocks in the cache?

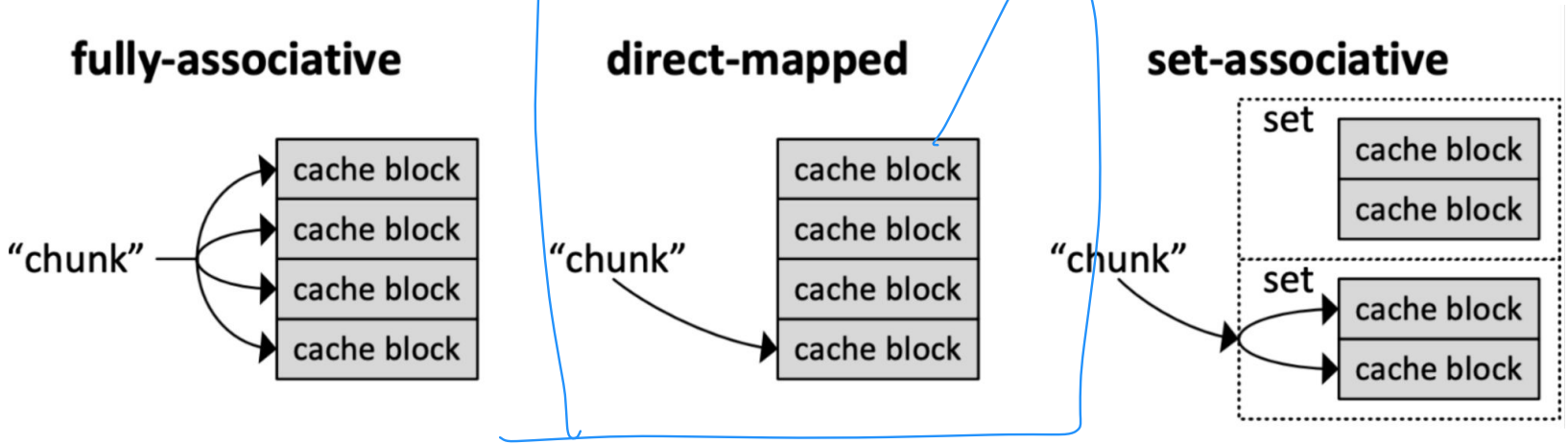
Cache design

byte-addressing

- How to map data chunks from main memory to blocks in the cache?

block stores a "chunk" of data

- 3 designs

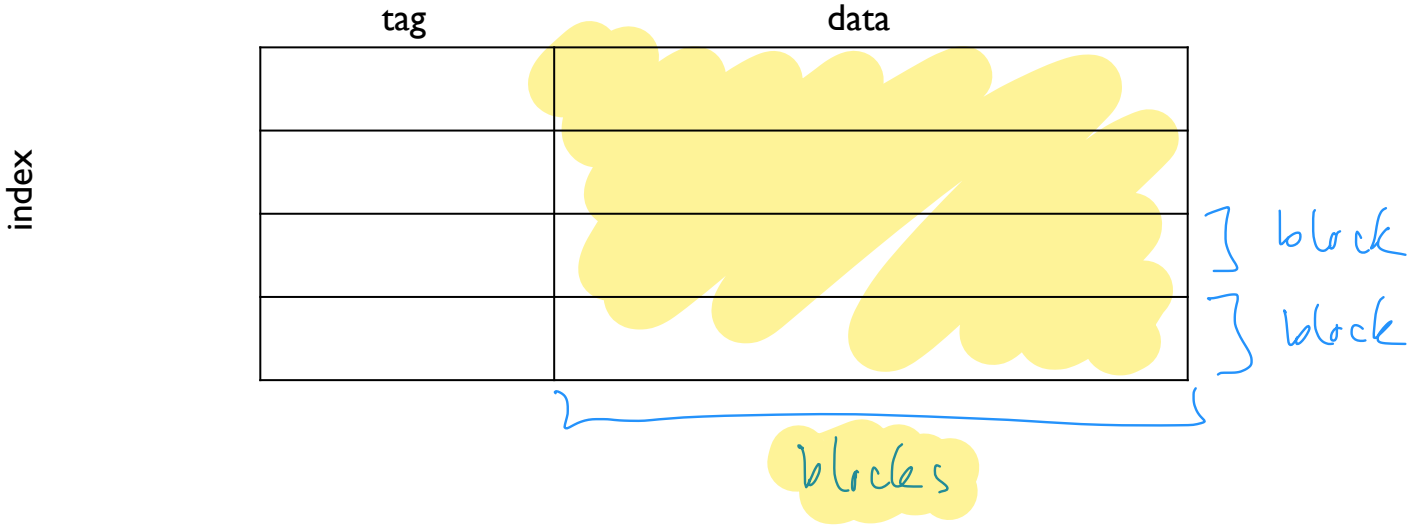


Direct mapped Cache

- A cache that can put a block of data in exactly 1 place

Direct mapped Cache

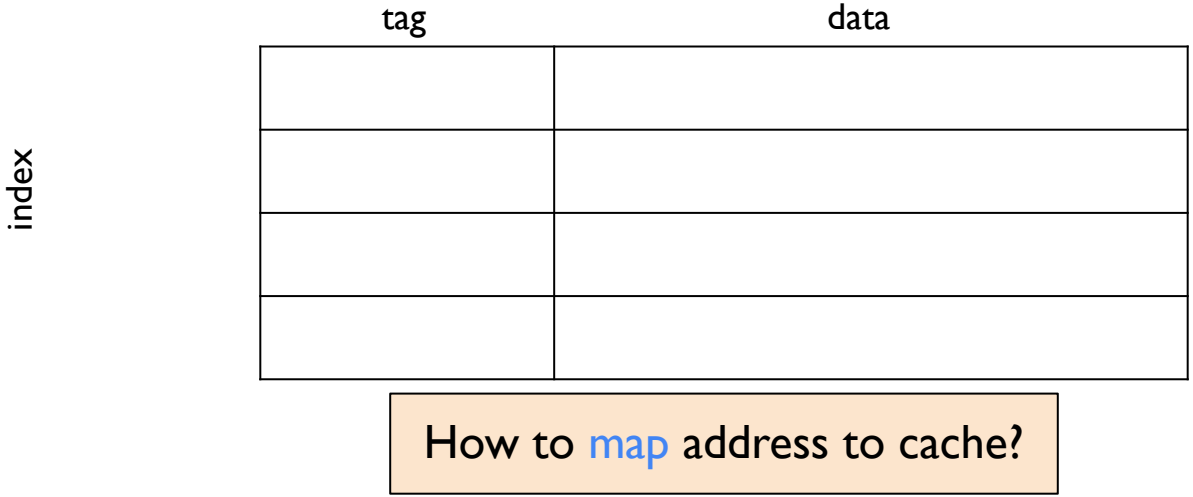
- A cache that can put a block of data in exactly 1 place
- Example: 4-block direct mapped cache
  - block size = 4 bytes (1 word)
  - each word in memory maps to exactly 1 cache location



4 blocks x block size
4 x 4 bytes
= 16 bytes
cache capacity

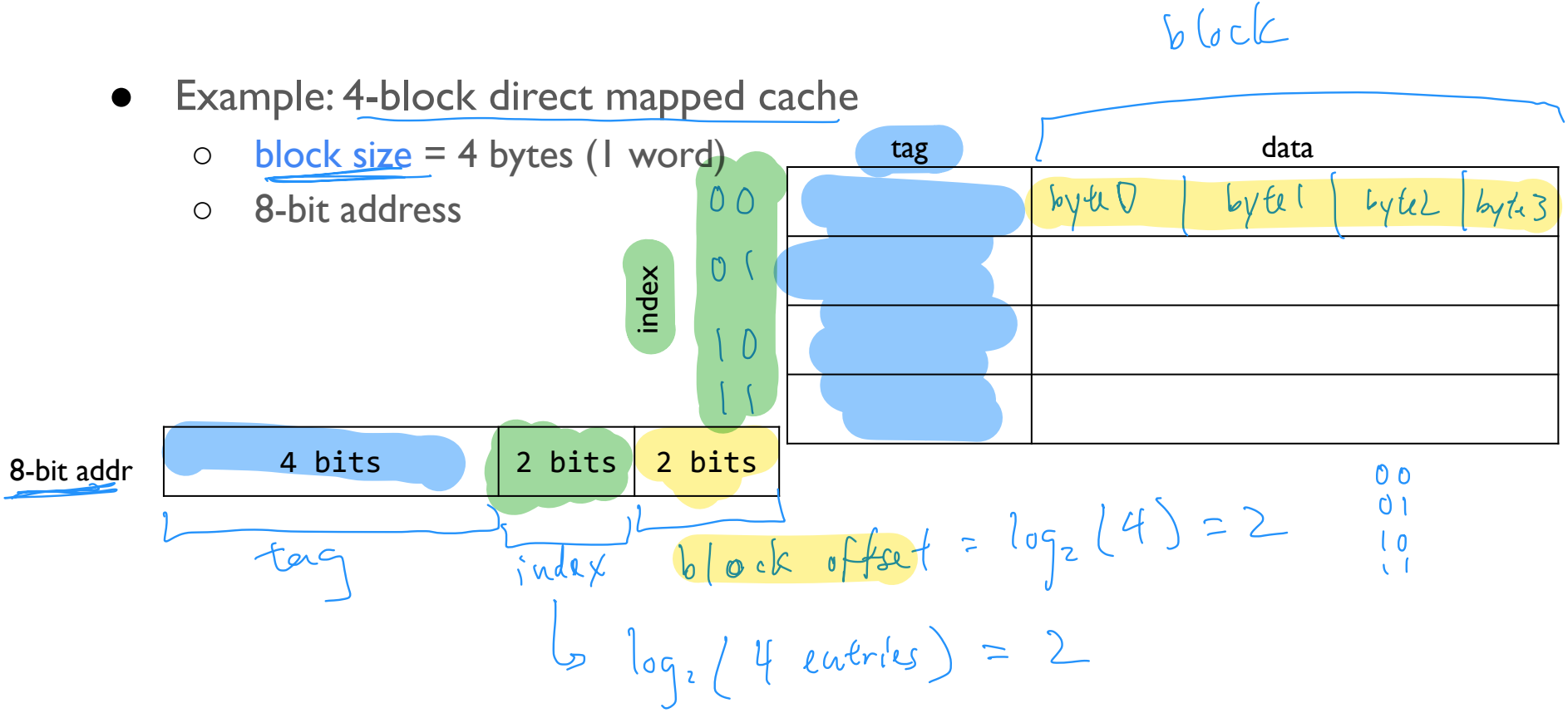
Direct mapped Cache

- Example: 4-block direct mapped cache
  - block size = 4 bytes (1 word)
  - each word in memory maps to exactly 1 cache location
  - 8-bit address



Direct mapped Cache

- Example: 4-block direct mapped cache
  - block size = 4 bytes (1 word)
  - 8-bit address



Extracting addresses from code

- A and B are int (4-byte) arrays with 2 elements.
  - Assume `i`, `j`, and `sum` are all in registers
  - A's base address: 0x10
  - B's base address: 0x38

```
int sum = 0;
for (int i = 0; i < 2; i++) {
    for (int j = 0; j < 2; j++) {
        sum += A[j] - B[j];
    }
}
```

A[0], B[0], A[1], B[1]

A[0], B[0], A[1], B[1]

What is the stream of addresses?

A. A[0], B[0], A[1], B[0], A[0], B[1], A[1], B[1]

B. A[0], B[0], A[0], B[0], A[1], B[1], A[1], B[1]

C. A[0], B[0], A[1], B[1], A[0], B[0], A[1], B[1]

D. A[0], B[0], A[1], B[1], A[0], B[1], A[0], B[1]

Direct Mapped Cache

Hit rate?
↳ # hits / # accesses

A. 25% B: 100%
B. 50%
C. 75%
D. 0%

- A cache that can put a block of data in exactly 1 place

Item	Addr b10	Addr b2	
A[0]	16	0001 00 00	M
B[0]	56	0011 10 00	M
A[1]	20	0001 01 00	M
B[1]	60	0011 11 00	M
A[0]	16	0001 00 00	H
B[0]	56	0011 10 00	H
A[1]	20	0001 01 00	H
B[1]	60	0011 11 00	H

tag | index | ↳ block offset

index

	tag	data
00	0001	A[0]
01	0001	A[1]
10	0011	B[0]
11	0011	B[1]

1. check index bits
2. compare tag
   ↳ if equal → hit!
   ↳ if NOT equal → miss!

Direct Mapped Cache

What is the hit rate for this code on this cache?

- A cache that can put a block of data in exactly 1 place

Item	Addr b10	Addr b2
A[0]	16	0001 00 00
B[0]	56	0011 10 00
A[1]	20	0001 01 00
B[1]	60	0011 11 00
A[0]	16	0001 00 00
B[0]	56	0011 10 00
A[1]	20	0001 01 00
B[1]	60	0011 11 00

index

tag	data

Miss types

- Useful to think about why cache suffers misses to design
  - caches that avoid misses
  - code that avoid misses

- Why do we miss on A[0] on first access?

Miss types

- Useful to think about why cache suffers misses to design
  - caches that avoid misses
  - code that avoid misses

- Why do we miss on A[0] on first access?
  - Never seen address A[0] before
  - Known as compulsory miss

to had to happen

Smaller direct mapped cache

2-block direct mapped cache
block size = 4 bytes

→ (1) check index bits
(2) compare tag

block = 4 bytes

Item	Addr b10	Addr b2	H/M?
A[0]	16	00010 0 00	M
B[0]	56	00111 0 00	M
A[1]	20	00010 1 00	M
B[1]	60	00111 1 00	M
A[0]	16	00010 0 00	M
B[0]	56	00111 0 00	M
A[1]	20	00010 1 00	M
B[1]	60	00111 1 00	M

tag | index | block offset

index	tag	data
0	00010	A[0]
1	00111	B[1]

What is the hit rate on this cache?

A. 75%
B. 50%
C. 25%
D. 12.5%
E. 0%

What happens as the cache gets bigger or smaller?

- Bigger caches generally have higher hit rates
  - Why? Because they can store more data

What happens as the cache gets bigger or smaller?

- Bigger caches generally have higher hit rates
  - Why? Because they can store more data

- Your code works best when your data fits in cache!

Miss types

- Why do we miss on A[0] on first access?
  - Compulsory miss

- Why do we miss A[0] on the second access?

Miss types

- Why do we miss on A[0] on first access?
  - Compulsory miss

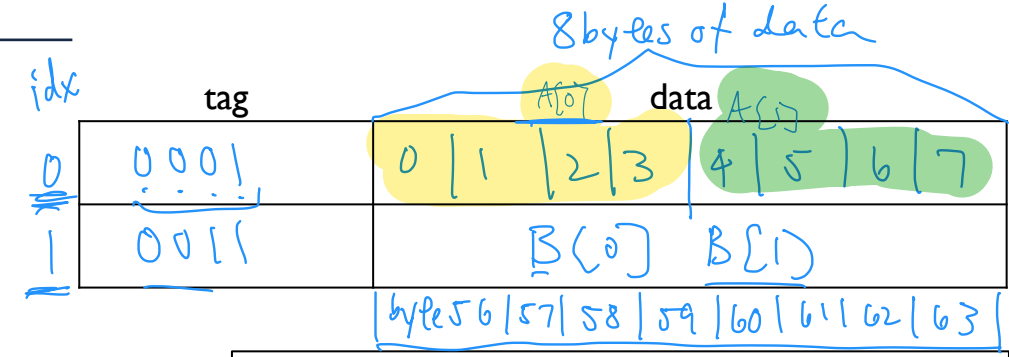
- Why do we miss A[0] on the second access?
  - Capacity miss
    - If only have room for 2 items but trying to store 4, then you're out of space

Larger block sizes

Item	Addr b10	Addr b2	H/M?
A[0]	16	0001 0 000	M
B[0]	56	0011 1 000	M
A[1]	20	0001 0 100	H
B[1]	60	0011 1 100	H
A[0]	16	0001 0 000	H
B[0]	56	0011 1 000	H
A[1]	20	0001 0 100	H
B[1]	60	0011 1 100	H

6 hits / 8 accesses = 75%.

Each block now holds 2 words (8 bytes)



What is the hit rate on this cache?

A. 75%
B. 50%
C. 25%
D. 12.5%
E. 0%

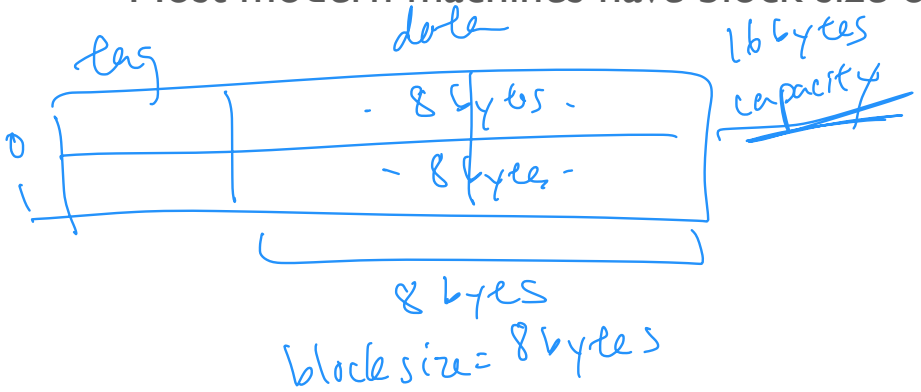
Larger block sizes

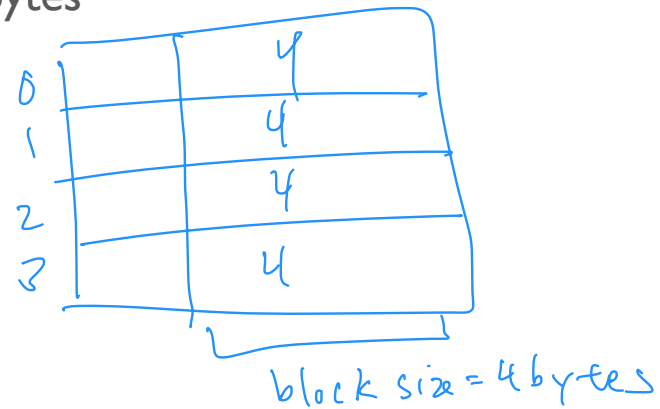
- Larger blocks help reduce misses by lowering spatial locality misses

Larger block sizes

- Larger blocks help reduce misses by lowering spatial locality misses

- The larger the block size, the fewer blocks can be stored in same sized cache
  - Tradeoff!
  - Most modern machines have block size of 64 bytes





Key caching formulas

- With direct mapped caches
  - `num_block_offset_bits = log2(block_size_in_bytes)`
  - `num_cache_entries = cache_size / block_size`
  - `num_index_bits = log2(num_cache_entries)`
  - `num_tag_bits = address_size - num_index_bits - num_block_offset_bits`