Memory & Caches II

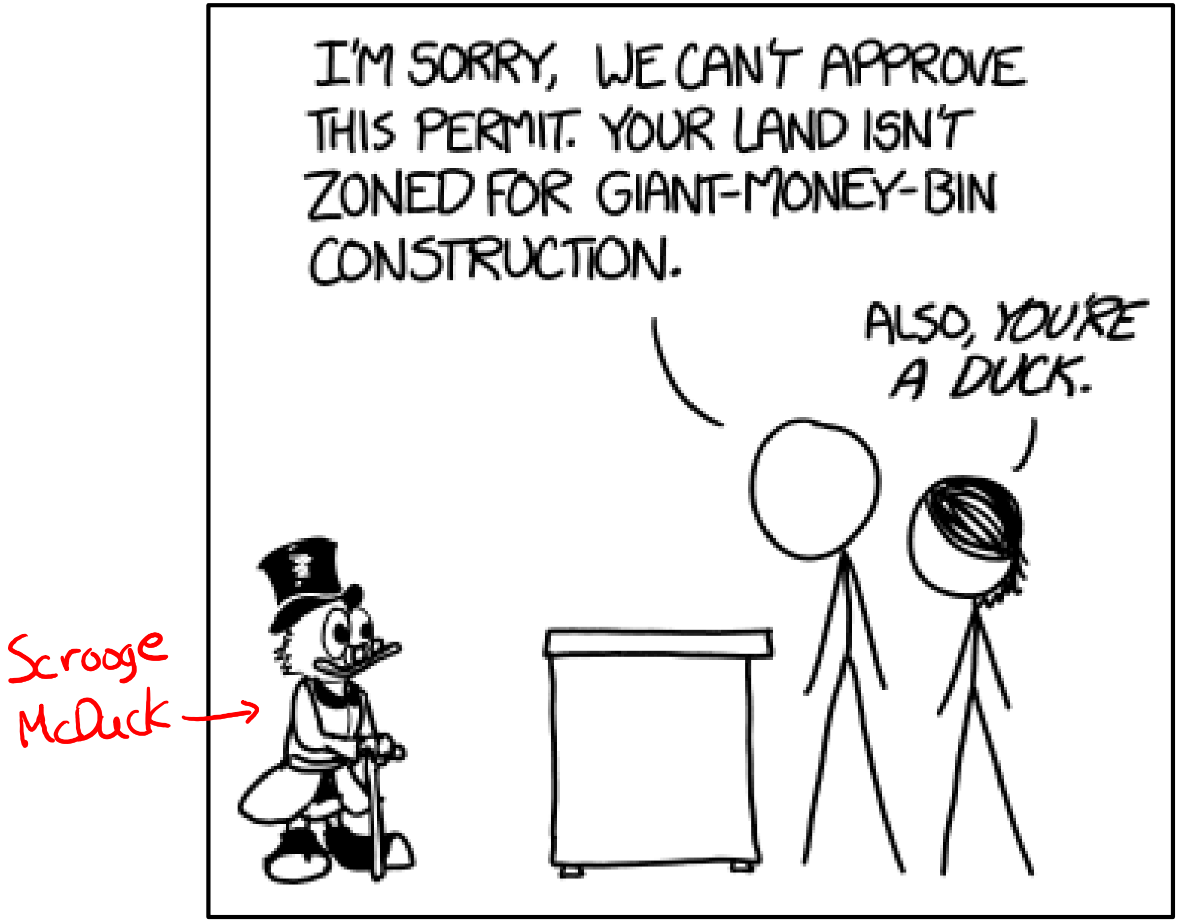
CSE 351 Autumn 2023

Instructor:

Justin Hsia

Teaching Assistants:

Afifah Kashif, Bhavik Soni, Cassandra Lam, Connie Lam, David Dai, Dawit Hailu, Ellis Haker, Eyoel Gebre, Joshua Tan, Malak Zaki, Naama Amiel, Nayha Auradkar, Nikolas McNamee, Pedro Amarante, Renee Ruan, Simran Bagaria, Will Robertson



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

Relevant Course Information

- Mid-quarter Survey due Wednesday (11/8)

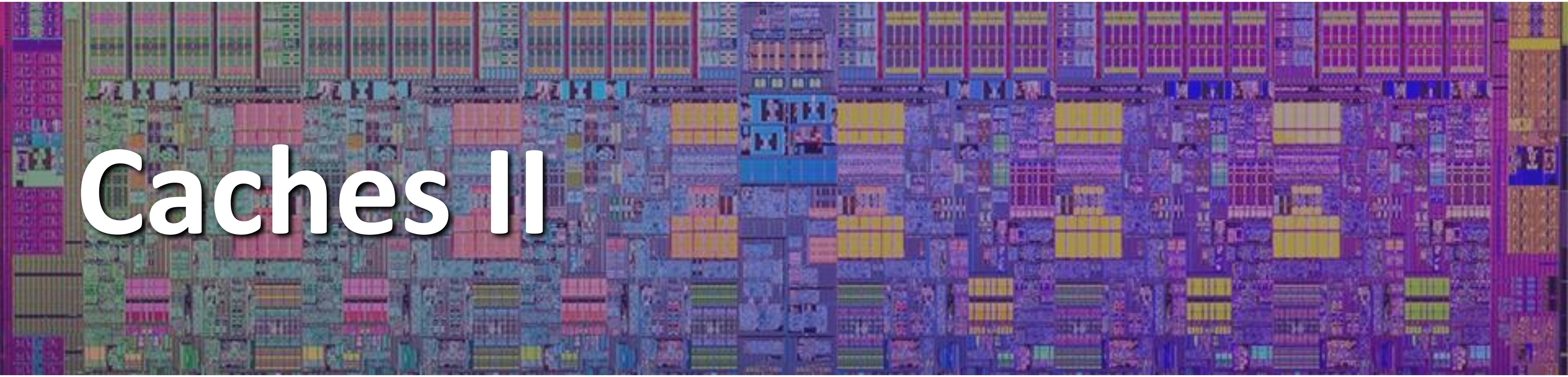
- hw16 due Wednesday (11/8)
- hw17 due *next* Wednesday (11/15)
  - Don't wait too long, this is a BIG hw (includes this lecture)

- Lab 3 due Friday (11/10)
  - Veteran's Day: no lecture, but some support hours (see Ed)

- Midterm grades will be released when we can
  - Regrade requests will be available afterward

New Lecture Format

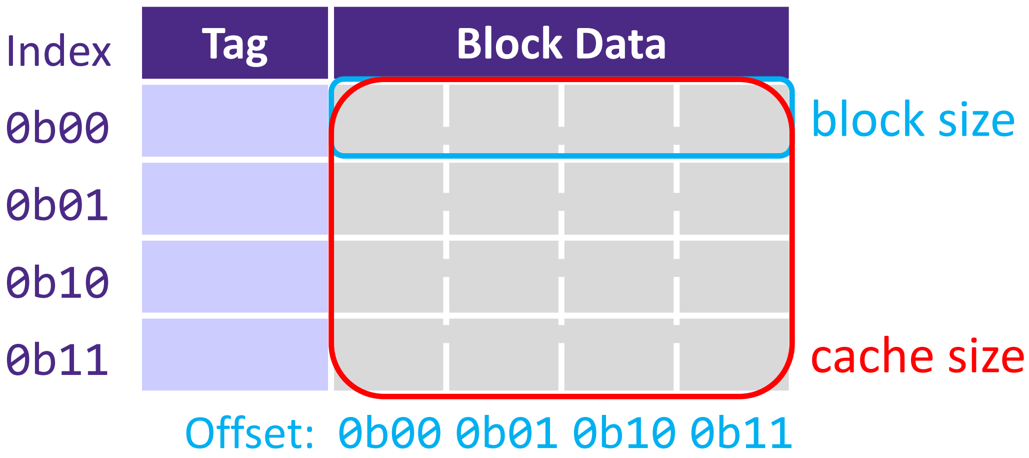
- Lesson Summary will be a little more detailed than before [5–7 min]
- Context section will "float" depending on content:
  - May be skipped for some lectures
  - May come after summary (as before) or may be held until end of lecture
  - Random call for share-out by section of the room for discussion questions
- Practice section will go problem-by-problem
  - Justin will introduce the problem
  - Then, limited group work time [3–8 min, depending on difficulty]
  - Random call for share-out by section of the room
    - Do not need to have finished the problem, just share out what work you did
- *The quicker we get responses, the quicker we can move on*



Caches II

Lesson Summary (1/2)

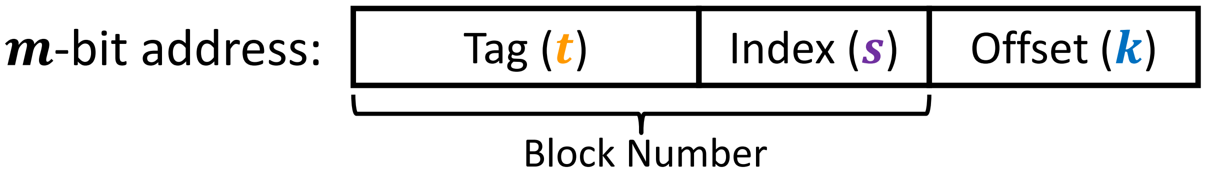
- Cache parameters define the cache geometry:
  - **Block size** is number of bytes per block
  - **Cache size** is number of bytes (or blocks) of data the cache can hold
- Finding a byte in the cache:
  - **Offset** refer to which byte in block
  - **Index** refers to which block in cache
- Example:
  - K = 4 B, C = 16 B = 4 blocks



Lesson Summary (2/2)

- **Direct-mapped cache:** each block in cache is assigned a unique index
  - Uses hash function of (block number) mod (# of cache indices)
    - Deterministic placement of each block, with many blocks mapping into the same index
    - Tag bits stored in cache and used to distinguish between blocks that map to same index

- Accessing the cache:
  (TIO address breakdown)



1) **Index** field tells you where to look in cache (width $s = \log_2 S$)
2) **Tag** field lets you check that data is the block you want (width $t = m - s - k$)
3) **Offset** field selects specified start byte within block (width $k = \log_2 K$)

Lesson Q&A

- Terminology:
  - Cache parameters: block size (K), cache size (C), number of indices (S)
  - Address fields: tag (t bits wide), index (s bits wide), block offset (k bits wide)
  - Cache organization: direct-mapped cache

- Learning Objectives:
  - Determine how memory addresses and data interact with the cache (*i.e.*, cache lookups, data movement).
  - Analyze how changes to cache parameters [and policies] affect performance metrics such as AMAT.

- What lingering questions do you have from the lesson?

Caches II – Practice

Practice Questions (1/2)

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

A. 0x3 — $\lfloor 3/8 \rfloor = 0$ — 0b 00 0|011 — block num 0

B. 0x1F — $\lfloor 31/8 \rfloor = 3$ — 0b 01 1|111 — block num 3

C. 0x30 — $\lfloor 48/8 \rfloor = 6$ — 0b 11 0|000 — block num 6

D. 0x38 — $\lfloor 56/8 \rfloor = 7$ — 0b 11 1|000 — block num 7

Practice Questions (2/2)

- 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:
    - (0xE: miss), (0xF: hit), (0x10: miss)

hit: block with data already in $

miss: data not in $, pulls block containing data from Mem

① pulls block containg 14 into $

② 14 & 15 are in the same block

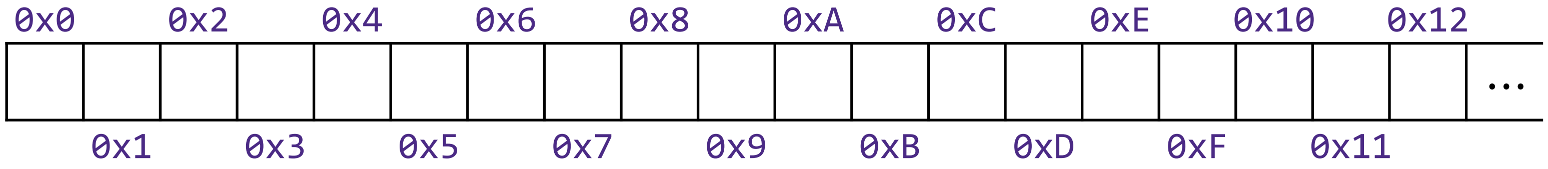
③ 16 is in a different block

A. 4 bytes

B. 8 bytes

C. 16 bytes

D. 32 bytes



Practice Questions (2/2)

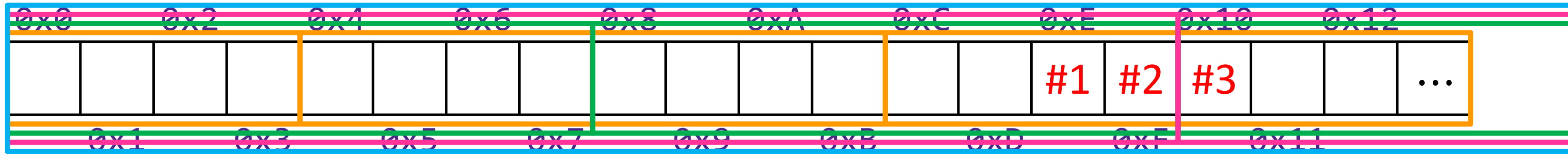
- Based on the following behavior, which of the following block sizes is NOT possible for our cache?
  - Cache starts *empty*, also known as a *cold cache*
  - Access (addr: hit/miss) stream:
    - (`0xE`: miss), (`0xF`: hit), (`0x10`: miss)
    - Need 0xE and 0xF in same block; 0x10 in different block

A. 4 bytes

B. 8 bytes

C. 16 bytes

D. 32 bytes

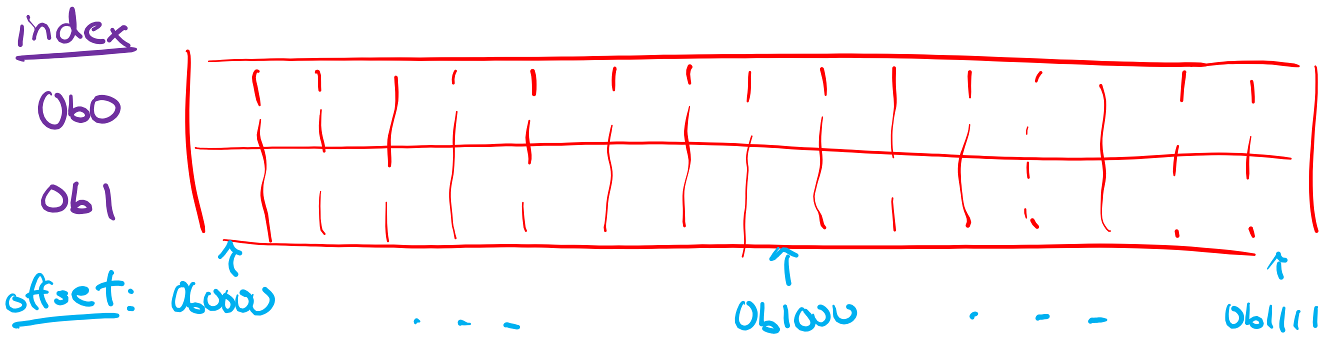


Homework Question Setup

- Consider the following function, which computes the dot product of two vectors:

```
float dotprod(float x[8], float y[8]) {
   float sum = 0.0;
   for (int i = 0; i < 8; i++)
      sum += x[i] * y[i];
   return sum;
}
```

  - `&x=0x0`, `&y=0x20`. `sizeof(float)=4`. Direct-mapped $ with indices, each 16 B.
  - **What does the cache look like?**



Homework Question Setup

- Consider the following function, which computes the dot product of two vectors:

```
float dotprod(float x[8], float y[8]) {
   float sum = 0.0;
   for (int i = 0; i < 8; i++)
      sum += x[i] * y[i];
   return sum;
}
```

  - &x=0x0, &y=0x20. sizeof(float)=4. Direct-mapped $ with indices, each 16 B.
  - **List out the first 8 memory accesses in terms of x and y, then translate those to addresses:**

(1) x[0] → 0b 0000 0000
(2) y[0] → 0b 0010 0000
(3) x[1] → 0b 0000 0100
(4) y[1] → 0b 0010 0100
(5) x[2] → 0b 0000 1000
(6) y[2] → 0b 0010 1000
(7) x[3] → 0b 0000 1100
(8) y[3] → 0b 0010 1100

Group Work Time

- During this time, you are encouraged to work on the following:
  1) If desired, continue your discussion
  2) Work on the homework problems
  3) Work on the current lab

- Resources:
  - You can revisit the lesson material
  - Work together in groups and help each other out
  - Course staff will circle around to provide support