Caches III
CSE 351 Winter 2020

Instructor:
Ruth Anderson

Teaching Assistants:
Jonathan Chen
Justin Johnson
Porter Jones
Josie Lee
Jeffery Tian
Callum Walker
Eddy (Tianyi) Zhou

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

- hw15 due Friday (2/21)
- Lab 3 due Monday (2/24)
- hw16 due Monday (2/24)
- hw17 due Wednesday (2/26)
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
Review: Direct-Mapped Cache

Hash function: (block number) mod (# of blocks in cache)

- Each memory address maps to exactly one index in the cache
- Fast (and simpler) to find a block

Here \( K = 4 \) B and \( C/K = 4 \)
Direct-Mapped Cache Problem

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

Solution?

![Memory and Cache Diagram]

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

- 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 more than one way

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

Solves conflict problem!
Cache Organization (3)

- **Associativity** \((E)\): # of ways for each set
  - Such a cache is called an “\(E\)-way set associative cache”
  - We now index into cache sets, of which there are \(S = \frac{C}{K} \times E\)
  - Use lowest \(\log_2(\frac{C}{K} / E) = s\) bits of block address
    - Direct-mapped: \(E = 1\), so \(s = \log_2(\frac{C}{K})\) as we saw previously
    - Fully associative: \(E = \frac{C}{K}\), so \(s = 0\) bits

Note: The textbook uses “\(b\)” for offset bits.
Example Placement

Where would data from address 0x1833 be placed?

- Binary: 0b 0001 1000 0011 0011

\[
t = m - s - k \quad s = \log_2(C/K/E) \quad k = \log_2(K)
\]

\[
s = \log_2(C/K/E) \quad m - \text{bit address:} \quad \begin{array}{ccc}
\text{Tag} (t) & \text{Index} (s) & \text{Offset} (k)
\end{array}
\]

- Direct-mapped
- 2-way set associative
- 4-way set associative

\[
\begin{array}{|c|c|c|}
\hline
\text{Set} & \text{Tag} & \text{Data} \\
\hline
0 & 0 & \checkmark \\
1 & 1 & \\
2 & 1 & \\
3 & 1 & \checkmark \\
4 & 1 & \\
5 & 1 & \\
6 & 1 & \\
7 & 1 & \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|c|}
\hline
\text{Set} & \text{Tag} & \text{Data} \\
\hline
0 & 0 & \\
0 & 1 & \\
0 & 2 & \checkmark \\
0 & 3 & \checkmark \\
1 & 0 & \\
1 & 1 & \checkmark \\
1 & 2 & \checkmark \\
1 & 3 & \checkmark \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|c|}
\hline
\text{Set} & \text{Tag} & \text{Data} \\
\hline
0 & 0 & \\
0 & 1 & \checkmark \\
0 & 2 & \checkmark \\
0 & 3 & \checkmark \\
1 & 0 & \checkmark \\
1 & 1 & \checkmark \\
1 & 2 & \checkmark \\
1 & 3 & \checkmark \\
\hline
\end{array}
\]
Block Replacement

- Any empty block in the correct set may be used to store block.
- 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”).

![Diagram of cache sets and tags](image)
Polling Question

- We have a cache of size 2 KiB with block size of 128 B. If our cache has 2 sets, what is its associativity?
  - Vote at [http://pollev.com/rea](http://pollev.com/rea)
  - A. 2
  - B. 4
  - C. 8
  - D. 16
  - E. We’re lost...

- If addresses are 16 bits wide, how wide is the Tag field? $k = \log_2(K) = 7$ bits, $s = \log_2(s) = 1$ bit, $t = m - s - k = \boxed{8}$ bits
General Cache Organization \((S, E, K)\)

- **Cache size:**
  \[ C = K \times E \times S \text{ data bytes} \]
  (doesn’t include \(V\) or \(Tag\))

- **Valid bit**

E = blocks (or lines) per set

- **Set**
- **Line** (block plus management bits)

\(S\) sets
\(= 2^s\)

\(K\) = bytes per block
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Variable</th>
<th>Formulas</th>
</tr>
</thead>
</table>
| Block size              | $K (B \text{ in book})$ | $M = 2^m \leftrightarrow m = \log_2 M$
|                         |          | $S = 2^s \leftrightarrow s = \log_2 S$
|                         |          | $K = 2^k \leftrightarrow k = \log_2 K$                                  |
| Cache size              | $C$      | $C = K \times E \times S$
| Associativity           | $E$      | $s = \log_2(C/K/E)$                                                     |
| Number of Sets          | $S$      | $m = t + s + k$                                                          |
| Address space           | $M$      |                                                                          |
| Address width           | $m$      |                                                                          |
| Tag field width         | $t$      |                                                                          |
| Index field width       | $s$      |                                                                          |
| Offset field width      | $k (b \text{ in book})$ |                                                                 |
Example Cache Parameters Problem

4 KiB address space, 125 cycles to go to memory.

Fill in the following table:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache Size</td>
<td>256 B</td>
</tr>
<tr>
<td>Block Size</td>
<td>32 B</td>
</tr>
<tr>
<td>Associativity</td>
<td>2-way</td>
</tr>
<tr>
<td>Hit Time</td>
<td>3 cycles</td>
</tr>
<tr>
<td>Miss Rate</td>
<td>20%</td>
</tr>
<tr>
<td>Tag Bits</td>
<td>5</td>
</tr>
<tr>
<td>Index Bits</td>
<td>2</td>
</tr>
<tr>
<td>Offset Bits</td>
<td>5</td>
</tr>
<tr>
<td>AMAT</td>
<td>3 + 0.2(125) = 28 clock cycles</td>
</tr>
</tbody>
</table>

\[ 2^n B \iff m = 12 \text{ bits} \]

\[ t = m - s - k \]

\[ s = \log_2 (C/K/E) \]

\[ k = \log_2 (K) \]

\[ \text{AMAT} = HT + MR \times MP \]
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

$E =$ blocks/lines per set

$S =$ # sets = $2^s$

Address of byte in memory:

- $t$ bits: tag
- $s$ bits: set index
- $k$ bits: block offset

data begins at this offset

valid bit

$K =$ bytes per block
Example: Direct-Mapped Cache ($E = 1$)

Direct-mapped: One line per set

Block Size $K = 8\ \text{B}$

<table>
<thead>
<tr>
<th>Set 0</th>
<th>Set 1</th>
<th>Set 2</th>
<th>...</th>
<th>Set S-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>Tag</td>
<td>0 1 2 3 4 5 6 7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>V</td>
<td>Tag</td>
<td>0 1 2 3 4 5 6 7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>V</td>
<td>Tag</td>
<td>0 1 2 3 4 5 6 7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>V</td>
<td>Tag</td>
<td>0 1 2 3 4 5 6 7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

8B in block

Address of int:

<table>
<thead>
<tr>
<th>Bits</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0...1</td>
<td>100</td>
</tr>
</tbody>
</table>

find set

$S=2^s$ sets
Example: Direct-Mapped Cache ($E = 1$)

Direct-mapped: One line per set
Block Size $K = 8 \text{ B}$
Example: Direct-Mapped Cache ($E = 1$)

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

No match? Then old line gets evicted and replaced
Example: Set-Associative Cache ($E = 2$)

2-way: Two lines per set
Block Size $K = 8 \text{ B}$
Example: Set-Associative Cache ($E = 2$)

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

Address of **short int**: 
8 bits: 0...01 100

valid? + match: yes = hit

compare both

block offset
Example: Set-Associative Cache ($E = 2$)

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

Valid? + Match: yes = hit

Address of short int:

\[ \begin{array}{c}
\text{bits} \\
0...01 \\
100 \\
\end{array} \]

compare both

Block offset

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), ...
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
Example Code Analysis Problem

- Assuming the cache starts cold (all blocks invalid) and \( \text{sum}, i, \) and \( j \) are stored in registers, calculate the miss rate:
  - \( m = 12 \) bits, \( C = 256 \) B, \( K = 32 \) B, \( E = 2 \)

```c
#define SIZE 8
long ar[SIZE][SIZE], sum = 0; // &ar=0x800
for (int i = 0; i < SIZE; i++)
  for (int j = 0; j < SIZE; j++)
    sum += ar[i][j];
```

**Challenge:** What is the miss rate if we switch the ordering of the for-loops?