Class Announcements

1. Programming Assignment 2 Due on Wednesday October 21\textsuperscript{st} before 9AM

2. Collect your Midterm1 exams and Programming Assignment 1 feedback from me if you have not done so already.
Lecture Overview

1. Memory Hierarchy motivating example
2. Cache Organization
3. Direct Mapped Cache
Array copy

Changing the loop order while copying a 2 dimensional array alters the time taken by a factor of 37 on a CSL instructional machine
Memory Mountain from the CSAPP textbook
Memory Mountain

1. Read throughput decreases as the stride increases

2. Read throughput decreases as more data is copied.
Design the cache such that it can quickly do a lookup: given the address, decide if that location is in the cache (hit) or not (miss).

Divide all of memory into fixed size blocks. Transfer a block on a miss. Keep the block in the cache until something else knocks it out.

What block size should we pick?
Let's play with an unrealistically small cache example. It will hold only 4 blocks.

A block lands in a block frame set (textbook)

<table>
<thead>
<tr>
<th>frame#</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
</tr>
<tr>
<td>01</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>11</td>
</tr>
</tbody>
</table>

2 bits of address are used to determine the frame#. LSBs identify byte/word within the block.
Each main memory block maps to a specific block frame.

2 bits of the address define this mapping.
Many main memory blocks map to the same block frame.
Only 1 can be in the block frame.
We have to quickly decide if the right one is in the frame.
The only thing we have to use is the address.

So... store the remainder of the address of a block with the block. Called a tag.
Address as used by the cache for a lookup.

| tag | index # | byte w/ block |

Bits of SRAM cannot identify whether a block from memory has or has not been placed in a block frame.

So, keep 1 bit per frame to identify if data is valid or not.
Generic Cache Organization

Set 0:

Valid | Tag | 0 | 1 | \cdots | B-1
\vdots
Valid | Tag | 0 | 1 | \cdots | B-1

\vdots

Set 1:

\vdots

Set S-1:

Valid | Tag | 0 | 1 | \cdots | B-1
\vdots

\vdots

Cache size: \( C = B \times E \times S \) data bytes

\( B = 2^b \) bytes per cache block

\( E \) lines per set

\( S = 2^s \) sets

1 valid bit per line

\( t \) tag bits per line

\( S = 2^s \) sets

\( C = B \times E \times S \) data bytes
Completed diagram of the cache:

<table>
<thead>
<tr>
<th>tag</th>
<th>index #</th>
<th>byte w/i block</th>
</tr>
</thead>
</table>

address

valid

tags  data blocks

<table>
<thead>
<tr>
<th>00</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Looking up a memory address in Direct Mapped Cache
This cache is called direct mapped or 1-way set associative or set associative, with a set size of 1.

Each index # maps to exactly 1 block frame.
Cache Lookup

Three steps while determining whether a request is a hit or a miss:

• **Set selection**: Select the set where the address resides.

• **Line matching**: Select the cache line within the set.

• **Word extraction**: Extract the requested word from the right offset.
Lookup algorithm:
(cache receives address)

[use index to identify frame]

if frame is valid
  if frame's tag matches address' tag
    HIT
  else
    MISS
else
  MISS

Valid tag | data blocks
Looking up a memory address in Direct Mapped Cache

=1? (1) The valid bit must be set

Selected set (i):

(2) The tag bits in the cache line must match the tag bits in the address

(3) If (1) and (2), then cache hit, and block offset selects starting byte.

0110

m-1

Tag Set index Block offset

0 1 2 3 4 5 6 7

w0 w1 w2 w3

Looking up a memory address in Direct Mapped Cache