Cache Summary

• Fast memory that exists between registers and main memory
• Takes advantage of temporal + spatial locality
  • Temporal locality: Programs often access the same location multiple times
  • Spatial locality: Programs often read/write to adjacent locations
• Caching can greatly reduce the number of accesses to main memory
  • Programs run much faster this way
• Caching is taken care of by the hardware
  • Programmers do not have explicit control over the caches
Cache Properties

• Cache size
• Cache block size
• Cache associativity
• Other derived quantities (such as the number of sets) can be computed from the above three properties
Cache Associativity

- Determines the number of different locations a given address can map to in the cache
- Ex. Cache associativity = 1 (direct-mapped)
  - This means that every address has only one possible line it can map to
- Ex. Cache associativity = Cache size / Block size (fully-associative)
  - This means that any address can map to any line of the cache
Cache Mapping

• To determine where addresses map into a cache, you need to break the address space up into TAG, SET, and OFFSET bits
  • Work from right to left
  • The log(B)-most bits are used to express block offset
  • The next log(S)-most bits are used to express the set number
  • The remaining bits represent the tag
Cache Mapping

• Think about it from a logical perspective
  • If you iterate through memory, it fills up the cache sequentially
  • Incrementing by the block size increments the set number
  • Incrementing by the block size * number of sets increments the tag

• This allows caches to capitalize on locality
## Example Cache

<table>
<thead>
<tr>
<th>Set</th>
<th>Valid</th>
<th>Tag</th>
<th>B0</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>Set</th>
<th>Valid</th>
<th>Tag</th>
<th>B0</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>1</td>
<td>2F</td>
<td>01</td>
<td>20</td>
<td>40</td>
<td>03</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>03</td>
<td>4F</td>
<td>D4</td>
<td>A1</td>
<td>3B</td>
<td>2</td>
<td>1</td>
<td>0E</td>
<td>99</td>
<td>09</td>
<td>87</td>
<td>56</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>06</td>
<td>11</td>
<td>23</td>
<td>6A</td>
<td>42</td>
<td>4</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>21</td>
<td>DE</td>
<td>AD</td>
<td>BE</td>
<td>EF</td>
<td>5</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>6</td>
<td>1</td>
<td>37</td>
<td>22</td>
<td>B6</td>
<td>DB</td>
<td>AA</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>11</td>
<td>00</td>
<td>12</td>
<td>51</td>
<td>55</td>
<td>7</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Example Problems

• The cache on the previous slide has the following properties:
  • 64 byte cache size
  • 4-byte block size
  • 2-way set associative

• Assume an 11-bit address space

• Data is 1-byte addressable
Example Problems

• HIT or MISS?
  • READ 0x435
  • READ 0x388
  • READ 0x0D3
Example Problems

• What is the miss rate for the following code?
  • Assume cache size 1 KB, direct-mapped, 16B block size

```c
for (int i = 0; i < 64; i++) {
    for (int j = 0; j < 64; j++) {
        array[i][j] = 0;
    }
}
```
Example Problems

• In the previous example, what code modifications can change the miss rate?

• What cache changes can changes the miss rate?
  • Changing the cache size?
  • Changing the associativity?
  • Changing the block size?
Cache Experiments

• Assume we have some way of querying the cache to see whether certain addresses hit or miss

• What sequences of accesses can help us find more about the cache?
  • Block size
  • Associativity
  • Cache size