CSE 351

Caches
Section Feedback

• Before we begin, we’d like to get some feedback about section
• If you could answer the following questions on the provided notecard that would be great:
  • What is working?
  • What is not working?
  • What should we start doing?
• This is anonymous! Don’t put your name on the notecard
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 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

• Why do we order the bits TAG|INDEX|OFFSET?
  • This allows caches to capitalize on locality
• OFFSET bits are the lowest-order bits
  • Adjacent addresses are thus in the same block
  • Takes advantage of spatial locality
• INDEX bits are next
  • Adjacent blocks are in different sets
  • Takes advantage of temporal locality
• The location of the TAG bits is a result of the placement of the first two sets of bits
Example Cache: direct-mapped

<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>1</td>
<td>15</td>
<td>63</td>
<td>B4</td>
<td>C1</td>
<td>A4</td>
<td>8</td>
<td>0</td>
<td>0</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>9</td>
<td>1</td>
<td>00</td>
<td>01</td>
<td>12</td>
<td>23</td>
<td>34</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10</td>
<td>1</td>
<td>01</td>
<td>98</td>
<td>89</td>
<td>CB</td>
<td>BC</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0D</td>
<td>DE</td>
<td>AF</td>
<td>BA</td>
<td>DE</td>
<td>11</td>
<td>0</td>
<td>1E</td>
<td>4B</td>
<td>33</td>
<td>10</td>
<td>54</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>12</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>13</td>
<td>1</td>
<td>11</td>
<td>C0</td>
<td>04</td>
<td>39</td>
<td>AA</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>13</td>
<td>31</td>
<td>14</td>
<td>15</td>
<td>93</td>
<td>14</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>15</td>
<td>1</td>
<td>0F</td>
<td>FF</td>
<td>6F</td>
<td>30</td>
<td>00</td>
</tr>
</tbody>
</table>
Example Cache: 2-way set associative

<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 Cache: fully-associative

<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>1</td>
<td>1F4</td>
<td>00</td>
<td>01</td>
<td>02</td>
<td>03</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>1</td>
<td>AB</td>
<td>02</td>
<td>30</td>
<td>44</td>
<td>67</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>100</td>
<td>F4</td>
<td>4D</td>
<td>EE</td>
<td>11</td>
<td>0</td>
<td>1</td>
<td>34</td>
<td>FD</td>
<td>EC</td>
<td>BA</td>
<td>23</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>77</td>
<td>12</td>
<td>23</td>
<td>34</td>
<td>45</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>1</td>
<td>1C6</td>
<td>00</td>
<td>11</td>
<td>22</td>
<td>33</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>101</td>
<td>DA</td>
<td>14</td>
<td>EE</td>
<td>22</td>
<td>0</td>
<td>1</td>
<td>45</td>
<td>67</td>
<td>78</td>
<td>89</td>
<td>9A</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>1</td>
<td>01</td>
<td>70</td>
<td>00</td>
<td>44</td>
<td>A6</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>16</td>
<td>90</td>
<td>32</td>
<td>AC</td>
<td>24</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Example Problems

• The caches on the previous slides have the following properties:
  • 64 byte cache size
  • 4-byte block size
  • The associativity varies
• Assume an 11-bit address space
• Data is 1-byte addressable
Example Problems

• Direct-mapped cache:
  • READ 0x7AC
  • READ 0x24
  • READ 0x99F

• 2-way set associative cache:
  • READ 0x435
  • READ 0x388
  • READ 0x0D3

• Fully-associative cache:
  • READ 0x1DD
  • READ 0x719
  • READ 0x2AA
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