Lecture 23: Cache Examples

• Today’s topics:
  ■ Cache access
  ■ Example problems in cache design
  ■ Caching policies
Accessing the Cache
Accessing the Cache

Direct-mapped cache: each address maps to a unique location in cache

8-byte words
Sets
Data array
Offset

Byte address

101000

8 words: 3 index bits
The Tag Array

Direct-mapped cache: each address maps to a unique address
Example Access Pattern

Direct-mapped cache: each address maps to a unique address

Tag array

Byte address

Tag

Compare

Data array

101000

8-byte words

Assume that addresses are 8 bits long
How many of the following address requests are hits/misses?
4, 7, 10, 13, 16, 68, 73, 78, 83, 88, 4, 7, 10…
Increasing Line Size

A large cache line size → smaller tag array, fewer misses because of spatial locality

32-byte cache line size or block size
Associativity

Set associativity → fewer conflicts; wasted power because multiple data and tags are read

Tag array

Compare

Tag

Byte address

10100000

Way-1

Way-2

Data array
Associativity

How many offset/index/tag bits if the cache has 64 sets, each set has 64 bytes, 4 ways

Tag array

Compare

Data array

Way-1   Way-2

Byte address

Tag

10100000
Example 1

- 32 KB 4-way set-associative data cache array with 32 byte line sizes

- How many sets?

- How many index bits, offset bits, tag bits?

- How large is the tag array?

Cache size = \#sets x \#ways x blocksize
Index bits = log_2(\#sets)
Offset bits = log_2(blocksize)
Addr width = tag + index + offset
Example 1

- 32 KB 4-way set-associative data cache array with 32 byte line sizes

\[
\text{cache size} = \#\text{sets} \times \#\text{ways} \times \text{block size}
\]

- How many sets? 256

- How many index bits, offset bits, tag bits?
  
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>5</td>
<td>19</td>
</tr>
<tr>
<td>\log_2(#\text{sets})</td>
<td>\log_2(\text{block size})</td>
<td>\text{addrsize-index-offset}</td>
</tr>
</tbody>
</table>

- How large is the tag array?
  
  \[
  \text{tag array size} = \#\text{sets} \times \#\text{ways} \times \text{tag size}
  \]

  \[
  = 19 \text{ Kb} = 2.375 \text{ KB}
  \]
Example 2

Show how the following addresses map to the cache and yield hits or misses. The cache is direct-mapped, has 16 sets, and a 64-byte block size. Addresses: 8, 96, 32, 480, 976, 1040, 1096

Offset = address % 64  (address modulo 64, extract last 6)  
Index = address/64 % 16     (shift right by 6, extract last 4)  
Tag = address/1024          (shift address right by 10)

<table>
<thead>
<tr>
<th>32-bit address</th>
<th>22 bits tag</th>
<th>4 bits index</th>
<th>6 bits offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:</td>
<td>0</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>96:</td>
<td>0</td>
<td>1</td>
<td>32</td>
</tr>
<tr>
<td>32:</td>
<td>0</td>
<td>0</td>
<td>32</td>
</tr>
<tr>
<td>480:</td>
<td>0</td>
<td>7</td>
<td>32</td>
</tr>
<tr>
<td>976:</td>
<td>0</td>
<td>15</td>
<td>16</td>
</tr>
<tr>
<td>1040:</td>
<td>1</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>1096:</td>
<td>1</td>
<td>1</td>
<td>8</td>
</tr>
</tbody>
</table>
Example 3

- A pipeline has CPI 1 if all loads/stores are L1 cache hits
- 40% of all instructions are loads/stores
- 85% of all loads/stores hit in 1-cycle L1
- 50% of all (10-cycle) L2 accesses are misses
- Memory access takes 100 cycles
- What is the CPI?
Example 3

• A pipeline has CPI 1 if all loads/stores are L1 cache hits
  40% of all instructions are loads/stores
  85% of all loads/stores hit in 1-cycle L1
  50% of all (10-cycle) L2 accesses are misses
Memory access takes 100 cycles
What is the CPI?

Start with 1000 instructions
1000 cycles (includes all 400 L1 accesses)
+ 400 (ld/st) x 15% x 10 cycles (the L2 accesses)
+ 400 x 15% x 50% x 100 cycles (the mem accesses)
= 4,600 cycles
CPI = 4.6
Assume that addresses are 8 bits long. How many of the following address requests are hits/misses?
4, 7, 10, 13, 16, 24, 36, 4, 48, 64, 4, 36, 64, 4
Example 4

Assume that addresses are 8 bits long.
How many of the following address requests are hits/misses?
4, 7, 10, 13, 16, 24, 36, 4, 48, 64, 4, 36, 64, 4

M H M H M M M H M M H M M M M
Cache Misses

• On a write miss, you may either choose to bring the block into the cache (write-allocate) or not (write-no-allocate)

• On a read miss, you always bring the block in (spatial and temporal locality) – but which block do you replace?
  - no choice for a direct-mapped cache
  - randomly pick one of the ways to replace
  - replace the way that was least-recently used (LRU)
  - FIFO replacement (round-robin)
Writes

• When you write into a block, do you also update the copy in L2?
  - write-through: every write to L1 → write to L2
  - write-back: mark the block as dirty, when the block gets replaced from L1, write it to L2

• Writeback coalesces multiple writes to an L1 block into one L2 write

• Writethrough simplifies coherency protocols in a multiprocessor system as the L2 always has a current copy of data
Types of Cache Misses

• Compulsory misses: happens the first time a memory word is accessed – the misses for an infinite cache

• Capacity misses: happens because the program touched many other words before re-touching the same word – the misses for a fully-associative cache

• Conflict misses: happens because two words map to the same location in the cache – the misses generated while moving from a fully-associative to a direct-mapped cache