Memory Hierarchy

Jinkyu Jeong (jinkyu@skku.edu)
Computer Systems Laboratory
Sungkyunkwan University
http://csl.skku.edu
Outline

Textbook: P&H 5.5 (4th Ed.) / P&H 5.8 (5th Ed.)

• Common Principles
• Source of Misses
• Real World Stuff
The Memory Hierarchy

• Common principles apply at all levels of the memory hierarchy
  – Based on notions of caching

• At each level in the hierarchy
  – Block placement
  – Finding a block
  – Replacement on a miss
  – Write policy
Block Placement

• Determined by associativity
  – Direct mapped (1-way associative)
    • One choice for placement
  – n-way set associative
    • n choices within a set
  – Fully associative
    • Any location

• Higher associativity reduces miss rate
  – Increases complexity, cost, and access time
Finding a Block

<table>
<thead>
<tr>
<th>Associativity</th>
<th>Location method</th>
<th>Tag comparisons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct mapped</td>
<td>Index</td>
<td>1</td>
</tr>
<tr>
<td>n-way set associative</td>
<td>Set index, then search entries within the set</td>
<td>n</td>
</tr>
<tr>
<td>Fully associative</td>
<td>Search all entries</td>
<td>#entries</td>
</tr>
<tr>
<td></td>
<td>Full lookup table</td>
<td>0</td>
</tr>
</tbody>
</table>

- **Hardware caches**
  - Reduce comparisons to reduce cost

- **Virtual memory**
  - Full table lookup makes full associativity feasible
  - Benefit in reduced miss rate
Replacement

• Choice of entry to replace on a miss
  – Least recently used (LRU)
    • Complex and costly hardware for high associativity
  – Random
    • Close to LRU, easier to implement

• Virtual memory
  – LRU approximation with hardware support
Write Policy

• Write-through
  – Update both upper and lower levels
  – Simplifies replacement, but may require write buffer

• Write-back
  – Update upper level only
  – Update lower level when block is replaced
  – Need to keep more state

• Virtual memory
  – Only write-back is feasible, given disk write latency
Sources of Misses (1)

• Three C’s
  – Compulsory misses (aka cold start misses)
    • First access to a block
  – Capacity misses
    • Due to finite cache size
    • A replaced block is later accessed again
  – Conflict misses (aka collision misses)
    • In a non-fully associative cache
    • Due to competition for entries in a set
    • Would not occur in a fully associative cache of the same total size
Sources of Misses (2)

- **Cache Design Trade-offs**

<table>
<thead>
<tr>
<th>Design change</th>
<th>Effect on miss rate</th>
<th>Negative performance effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Increase cache size</td>
<td>Decrease capacity misses</td>
<td>May increase access time</td>
</tr>
<tr>
<td>Increase associativity</td>
<td>Decrease conflict misses</td>
<td>May increase access time</td>
</tr>
<tr>
<td>Increase block size</td>
<td>Decrease compulsory misses</td>
<td>Increases miss penalty.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For very large block size, may increase miss rate due to pollution.</td>
</tr>
</tbody>
</table>
Real World Stuff (1)

- Intel Nehalem 4-core processor
  - Per core: 32KB L1 I-cache, 32KB L1 D-cache, 512KB L2 cache
## Real World Stuff (2)

- **2-Level TLB Organization**

<table>
<thead>
<tr>
<th></th>
<th>Intel Nehalem</th>
<th>AMD Opteron X4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual addr</td>
<td>48 bits</td>
<td>48 bits</td>
</tr>
<tr>
<td>Physical addr</td>
<td>44 bits</td>
<td>48 bits</td>
</tr>
<tr>
<td>Page size</td>
<td>4KB, 2/4MB</td>
<td>4KB, 2/4MB</td>
</tr>
<tr>
<td>L1 TLB (per core)</td>
<td>L1 I-TLB: 128 entries for small pages, 7 per thread (2 × ) for large pages</td>
<td>L1 I-TLB: 48 entries</td>
</tr>
<tr>
<td></td>
<td>L1 D-TLB: 64 entries for small pages, 32 for large pages</td>
<td>L1 D-TLB: 48 entries</td>
</tr>
<tr>
<td></td>
<td>Both 4-way, LRU replacement</td>
<td>Both fully associative, LRU replacement</td>
</tr>
<tr>
<td>L2 TLB (per core)</td>
<td>Single L2 TLB: 512 entries 4-way, LRU replacement</td>
<td>L2 I-TLB: 512 entries</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L2 D-TLB: 512 entries</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Both 4-way, round-robin LRU</td>
</tr>
<tr>
<td>TLB misses</td>
<td>Handled in hardware</td>
<td>Handled in hardware</td>
</tr>
</tbody>
</table>
## Real World Stuff (3)

### 3-Level Cache Organization

<table>
<thead>
<tr>
<th></th>
<th>Intel Nehalem</th>
<th>AMD Opteron X4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>L1 caches</strong></td>
<td><strong>L1 I-cache</strong>: 32KB, 64-byte blocks, 4-way, approx LRU replacement, hit time n/a</td>
<td><strong>L1 I-cache</strong>: 32KB, 64-byte blocks, 2-way, LRU replacement, hit time 3 cycles</td>
</tr>
<tr>
<td><strong>per core</strong></td>
<td><strong>L1 D-cache</strong>: 32KB, 64-byte blocks, 8-way, approx LRU replacement, write-back/allocate, hit time n/a</td>
<td><strong>L1 D-cache</strong>: 32KB, 64-byte blocks, 2-way, LRU replacement, write-back/allocate</td>
</tr>
<tr>
<td><strong>L2 unified</strong></td>
<td>256KB, 64-byte blocks, 8-way, approx LRU replacement, write-back/allocate, hit time n/a</td>
<td>512KB, 64-byte blocks, 16-way, approx LRU replacement, write-back/allocate, hit time 9 cycles</td>
</tr>
<tr>
<td><strong>cache</strong></td>
<td><strong>L3 unified</strong></td>
<td></td>
</tr>
<tr>
<td><strong>per core</strong></td>
<td><strong>L3 cache</strong>: 8MB, 64-byte blocks, 16-way, replacement n/a, write-back/allocate, hit time n/a</td>
<td><strong>L3 cache</strong>: 2MB, 64-byte blocks, 32-way, replace block shared by fewest cores, write-back/allocate, hit time 32 cycles</td>
</tr>
</tbody>
</table>

n/a: data not available
Real World Stuff (4)

• **Miss Penalty Reduction**
  – Return requested word first
    • Then back-fill rest of block
  – Non-blocking miss processing
    • Hit under miss: allow hits to proceed
    • Miss under miss: allow multiple outstanding misses
  – Hardware prefetch: instructions and data
  – Opteron X4: bank interleaved L1 D-cache
    • Two concurrent accesses per cycle

• **Inclusion vs. exclusion policy**
  – Intel & most other processors: inclusion policy
  – AMD processors: exclusion policy
Concluding Remarks

• Fast memories are small, large memories are slow
  – We really want fast, large memories 😞
  – Caching gives this illusion 😊

• Principle of locality
  – Programs use a small part of their memory space frequently

• Memory hierarchy
  – L1 cache ↔ L2 cache ↔ … ↔ DRAM memory ↔ disk

• Memory system design is critical for multiprocessors