Computer Systems Organization

CSE 521/560M
Lecture 15
Prof. Patrick Crowley

Plan for Today

- Questions
- Today’s discussion
- HW 1 returned
Importance of Memory Performance

Memory Hierarchy

- Registers: visible to ISA and renamed by hardware
- (Hierarchy of) Caches: plus their enhancements
  - Write buffers, victim caches, prefetch/stream buffers, etc.
- TLBs and their management
- Virtual memory system (O.S. level) and hardware assists
- Based on principle of locality
Illustrated Hierarchy

Questions that arise at each level

- What is the unit of information transferred from level to level? (word, block, page table entry, page)
- Where is that unit of info placed? (directed by ISA, restricted mapping, general mapping)
- How do we find if that unit of info is present? (depends on the mapping)
- What happens if there is no room for it? (structural hazard, replacement algorithm)
- What happens when we change the contents of the info unit? (i.e., what happens on a write)
Caches (on-chip, off-chip)

- Caches consist of a set of entries where each entry has
  - A block (or line) of data: some subset of memory content
  - A tag: allows us to recognize if the desired block is present
  - Status bits: valid, dirty, status for multiprocessors, etc.
- Capacity (or size) of a cache
  - Number of blocks * block size

Cache Organization

- Most restricted mapping
  - Direct-mapped cache. A given memory location (block) can only be mapped in a single place in the cache. Generally this place is given by:
    - (block address) mod (number of blocks in the cache)
    - Number of blocks usually a power of two
- Most general mapping
  - Fully-associative cache. A given memory location (block) can be mapped anywhere in the cache. No cache of decent size is implemented this way but this is the (general) mapping for pages (disk to main memory) and for small TLBs.
Cache Organization (cont’d)

- Less restricted mapping
  - *Set-associative* cache. Blocks in the cache are grouped into sets and a given memory location (block) maps into a set. Within the set the block can be placed anywhere. Sets with 2 (2-way set associative), 4, 8 and 16 blocks have been implemented.
  - Set usually chosen with
    - (block address) mod (number of sets in cache)
- Direct-mapped = 1-way set associative
- Fully associative with \( m \) entries is \( m \)-way set associative
- Capacity
  - Capacity = number of sets * set-associativity * block size

Organization Illustrated
Cache hit or miss?

- How to you detect if a memory address (a byte address) has a valid image in the cache
  - Address is decomposed into 3 fields
    - Block offset (depends on block size)
    - Index (depends on number of sets)
    - Tag (the remainder of the address)
  - The index determines the set, and the tag can be used to check for a match
  - Also need to check status bits for validity

Why Set-Associative Caches?

- Cons
  - The higher the associativity the larger the number of comparisons to be made in parallel for high-performance (can have an impact on cycle time for on-chip caches)
  - Higher associativity requires a wider tag array
- Pros
  - Better hit ratio
  - Great improvement from 1 to 2, less from 2 to 4, minimal after that (according to conventional wisdom)
Replacement Algorithm

- None for direct-mapped
- Random or LRU or pseudo-LRU for set-associative caches
  - Not a very important factor for performance

Write Policies

- Write-through
  - On a write hit, write both in cache and in memory
  - Pro: consistent view of memory (better for I/O)
  - Con: more memory traffic
- Write-back
  - On a write hit, write only in cache (requires a dirty bit)
  - Pro-con: reverse of write-through
- Write miss:
  - Write allocate
    - Usually used with write-back
  - No-write allocate (non-allocate, write-around)
    - Usually used with write-through
Classifying misses: The 3 C’s

• Compulsory (cold start)
  – The first time you touch a block. Reduced (for a given cache capacity and associativity) by having large blocks.

• Capacity
  – The working set is too big for the ideal cache of same capacity and block size (i.e., fully associative with optimal replacement algorithm). Only remedy: larger cache.

• Conflict (interference)
  – Mapping of two or more hot blocks to the same location. Increasing associativity decreases this type of miss.

• There is a fourth C: coherence misses (for multiprocessors)

Example Cache Hierarchies

<table>
<thead>
<tr>
<th>App</th>
<th>AMD Athlon</th>
<th>Intel Pentium 3</th>
<th>Intel Pentium 4</th>
<th>IBM PowerPC 405CR</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICache (L1)</td>
<td>64KB, 2-way</td>
<td>16KB, 2-way</td>
<td>12K RISC op</td>
<td>16KB, 2-way</td>
</tr>
<tr>
<td>DCache (L1)</td>
<td>64KB, 2-way</td>
<td>16KB, 2-way</td>
<td>Trace cache</td>
<td>2-way</td>
</tr>
<tr>
<td>L2 (on-chip)</td>
<td>256KB, 16-way</td>
<td>256-2048KB, 8-way</td>
<td>256KB, 8-way</td>
<td>None</td>
</tr>
</tbody>
</table>
Assignment

• Readings
  – For Monday
    • **Commentary**: The superblock: an effective technique for VLIW and superscalar compilation
  – For Wednesday
    • H&P: 5.3-5.5