Plan for Today

• Announcements
  – PM2 due today
  – Design presentations due Nov 4

• Questions

• Today’s discussion: Memory Hierarchy Basics
Design Presentations, Nov 4

• Your presentation should clearly describe your processor and the design choices you made in PM1 & 2

• Presentation logistics
  – Limit your presentation to 10 minutes, and expect 5 mins of questions.
  – Be sure that all group members contribute to the presentation.

• Send a copy of your PPT presentation to Shakir before class
  – You are free to use your own laptop for the presentation.
  – If you prefer to use Prof. Crowley’s laptop, you must submit your slides via email by 3pm or bring a copy on a USB memory stick.

• Also, before class, upload 10-minute screencast version (see next)
Project Design Screencast

• For at least two reasons, you will create 10 min screencasts of your design presentations

1. Use SnagIt (or another tool). My demo:
   • http://www.arl.wustl.edu/~pcrowley/cse/561/scast_demo.avi

2. Upload the video to youtube.com

3. Post the link on the course discussion page
Project Logistics

- Groups of three

- Three major deadlines
  - Design due Oct 19
  - Project Demo Nov 23
  - Report due Dec 7

- This week: PM2 due, design presentations

### Project Timeline

<table>
<thead>
<tr>
<th>Task</th>
<th>Due</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Milestone 1</td>
<td>Oct 19</td>
<td>Processor Design</td>
</tr>
<tr>
<td>Project Milestone 2</td>
<td>Nov 2</td>
<td>Detailed Design &amp; VHDL</td>
</tr>
<tr>
<td>Design Presentations</td>
<td>Nov 4</td>
<td>Presentation of detailed design</td>
</tr>
<tr>
<td>Project Demo</td>
<td>Nov 23</td>
<td>Demo working VHDL model</td>
</tr>
<tr>
<td>Final Report</td>
<td>Dec 7</td>
<td>Final report due</td>
</tr>
</tbody>
</table>
## Project Groups

<table>
<thead>
<tr>
<th>Group</th>
<th>Member 1</th>
<th>Member 2</th>
<th>Member 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>David Lu</td>
<td>Michael Schultz</td>
<td>Austin Abrams</td>
</tr>
<tr>
<td>2</td>
<td>Yong Fu</td>
<td>Greg Galloway</td>
<td>Christopher Thomas</td>
</tr>
<tr>
<td>3</td>
<td>Timothy York</td>
<td>Haowei Yuan</td>
<td>Stephen Schuh</td>
</tr>
<tr>
<td>4</td>
<td>Yu-Ying Liang</td>
<td>Abu Sayeed Saifullah</td>
<td>Raphael Njuguna</td>
</tr>
<tr>
<td>5</td>
<td>Cory Flanagin</td>
<td>Jessica Schupp</td>
<td></td>
</tr>
</tbody>
</table>
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
  – Main memory
  – Disks
  – Remote memory
• Based on principle of locality
Illustrated Hierarchy

- CPU
  - Registers
  - Register reference
    - Size: 500 bytes
    - Speed: 0.25 ns

- Cache
  - Cache reference
    - Size: 64 KB
    - Speed: 1 ns

- Memory
  - Memory reference
    - Size: 512 MB
    - Speed: 100 ns

- I/O bus
   - I/O devices
     - Disk memory reference
       - Size: 100 GB
       - Speed: 5 ms

© 2003 Elsevier Science (USA). All rights reserved.
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

Fully associative: block 12 can go anywhere

Direct mapped: block 12 can only go into block 4 (12 mod 8)

Set associative: block 12 can go anywhere in set 0 (12 mod 4)

© 2003 Elsevier Science (USA). All rights reserved.
Cache hit or miss?

• How to 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
Detecting a Cache Hit

- 32 bit addresses
- 2 blocks per set
- 2K sets
Why Set-Associative Caches?

• Pros
  – Better hit ratio
  – Great improvement from 1 to 2, less from 2 to 4, minimal after that (according to conventional wisdom)

• 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
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 & coherence)
  – Con: more memory traffic

• Write-back
  – On a write hit, write only in cache (requires a dirty bit)
    • Only update memory when evicted block is dirty
  – 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></th>
<th>AMD Athlon</th>
<th>Intel P3</th>
<th>Intel P4</th>
<th>IBM PPC 405CR</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>App</strong></td>
<td>Desktop</td>
<td>Desktop, server</td>
<td>Desktop</td>
<td>Embedded</td>
</tr>
<tr>
<td><strong>ICache(L1)</strong></td>
<td>64KB, 2-way</td>
<td>16KB, 2-way</td>
<td>12K uop trace cache</td>
<td>16KB, 2-way</td>
</tr>
<tr>
<td><strong>DCache(L1)</strong></td>
<td>64KB, 2-way</td>
<td>16KB, 2-way</td>
<td>8KB, 4-way</td>
<td>8KB, 2-way</td>
</tr>
<tr>
<td><strong>L2(on-chip)</strong></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 Wednesday
    • None
  
  – For Monday
    • H&P: Read C.3-C.4, 5.2
  
  – For Wednesday (Nov 11)
    • H&P: sections 5.3-5.6