The memory gap

1980: no cache in µproc;
1995 2-level cache on Alpha 21164 µproc

DRAM:
- value is stored as a charge on capacitor (must be refreshed; RAS/CAS)
- very small but slower than SRAM (factor of 5 to 10)

SRAM:
- value is stored on a pair of inverting gates (e.g., D-Latch)
- very fast but takes up more space than DRAM (4 to 6 transistors)

General Principles

• Locality
  - Temporal Locality: referenced again soon
  - Spatial Locality: nearby items referenced soon
• Locality + smaller mem is faster = memory hierarchy
  - Levels: each smaller, faster, more expensive/byte than level below
  - Inclusive: data found in top also found in the bottom
• Definitions
  - Upper is closer to processor
  - Block: minimum unit that present or not in upper level
  - Address = Block frame address + block offset address
  - Hit time: time to access upper level, including hit determination

Locality

• Temporal locality
  - Recently used item is likely to be re-used in near future
• Spatial locality
  - Addresses close together physically tend to be referenced close together in time

90/10 Locality Rule: Code executes 90% of its instructions in 10% of its code.

The memory hierarchy

Users want large, fast memories cheap! (conflict)

SRAM
- speed/capacity
- Lower power, faster

DRAM
- cost/capacity
- Refreshing, less expensive

Memory Technology Review

• DRAM:
  - value is stored as a charge on capacitor (must be refreshed; RAS/CAS)
  - very small but slower than SRAM (factor of 5 to 10)
• SRAM:
  - value is stored on a pair of inverting gates (e.g., D-Latch)
  - very fast but takes up more space than DRAM (4 to 6 transistors)
Who’s in control?

<table>
<thead>
<tr>
<th>Registers</th>
<th>Cache</th>
<th>Memory</th>
<th>Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>cost</td>
<td></td>
<td>size</td>
<td>compiler</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OS/hardware</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OS/user</td>
</tr>
</tbody>
</table>

Four Questions for Memory Hierarchy Designers

- Q1: Where can a block be placed in the upper level? (Block placement)
- Q2: How is a block found if it is in the upper level? (Block identification)
- Q3: Which block should be replaced on a miss? (Block replacement)
- Q4: What happens on a write? (Write strategy)

Q1: Where can a block be placed in the upper level?

- Block placement
  - Tag on each block
  - No need to check index or block offset
  - Increasing associativity shrinks index, expands tag

FA: No index
DM: Large index

IndexTagBlockOffset

Block address compared to block frame address (tag) for all frames in cache in parallel.

Given a block address, it can only be found in the set specified in index. All tags in index set must be compared to block address (in parallel) to find a hit.

Q2: How Is a Block Found If It Is in the Upper Level?

- Block identification
  - Tag on each block
  - No need to check index or block offset
  - Increasing associativity shrinks index, expands tag

Find block in set
Select set
Block address compared to block frame address (tag) for all frames in cache in parallel.

Q3: Which Block Should be Replaced on a Miss?

- Block replacement
  - Easy for Direct Mapped
  - Set Associative or Fully Associative:
    - Random (for large associativities, simple)
    - LRU (for smaller associativities, more complex)
    - FIFO (approximate LRU)

Alpha 21264: Data cache misses per 1000 instructions

<table>
<thead>
<tr>
<th>Use</th>
<th>Random</th>
<th>FIFO</th>
<th>Random</th>
<th>FIFO</th>
<th>Random</th>
<th>FIFO</th>
<th>Random</th>
<th>FIFO</th>
</tr>
</thead>
<tbody>
<tr>
<td>16KB</td>
<td>121.3</td>
<td>162.3</td>
<td>121.7</td>
<td>161.5</td>
<td>121.9</td>
<td>162.1</td>
<td>159.0</td>
<td>161.6</td>
</tr>
<tr>
<td>32KB</td>
<td>105.4</td>
<td>106.5</td>
<td>105.9</td>
<td>106.3</td>
<td>105.1</td>
<td>106.7</td>
<td>106.5</td>
<td>105.7</td>
</tr>
<tr>
<td>64KB</td>
<td>95.5</td>
<td>106.5</td>
<td>95.5</td>
<td>106.5</td>
<td>95.5</td>
<td>106.5</td>
<td>95.5</td>
<td>105.5</td>
</tr>
</tbody>
</table>
Q4: What Happens on a Write?

- Write through: The information is written to both the block in the cache and to the block in the lower-level memory.
- Write back: The information is written only to the block in the cache. The modified cache block is written to main memory only when it is replaced.
  - is block clean or dirty?
- Pros and Cons of each:
  - WT: read misses cannot result in writes (because of replacements)
  - WB: no writes of repeated writes
- WT always combined with write buffers so that don’t wait for lower level memory
- Write allocate vs. no write allocate