Lecture 15: Memory Hierarchy—Motivation, Definitions, Four Questions about Memory Hierarchy

Professor Randy H. Katz
Computer Science 252
Spring 1996
Who Cares about Memory Hierarchy?

- Processor Only Thus Far in Course
  - CPU cost/performance, ISA, Pipelined Execution

- 1980: no cache in µproc;
- 1995 2-level cache, 60% trans. on Alpha 21164 µproc
General Principles

• Locality
  – *Temporal Locality*: referenced again soon
  – *Spatial Locality*: nearby items referenced soon

• Locality + smaller HW 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
Cache Measures

- **Hit rate**: fraction found in that level
  - So high that usually talk about **Miss rate**
  - Miss rate fallacy: as MIPS to CPU performance, miss rate to average memory access time in memory
- **Average memory-access time** = Hit time + Miss rate x Miss penalty (ns or clocks)
- **Miss penalty**: time to replace a block from lower level, including time to replace in CPU
  - **access time**: time to lower level = \( f(\text{lower level latency}) \)
  - **transfer time**: time to transfer block = \( f(\text{BW upper & lower, block size}) \)
Block Size vs. Cache Measures

- Increasing Block Size generally increases Miss Penalty and decreases Miss Rate
Implications For CPU

• Fast hit check since every memory access
  – Hit is the common case

• Unpredictable memory access time
  – 10s of clock cycles: wait
  – 1000s of clock cycles:
    » Interrupt & switch & do something else
    » New style: multithreaded execution

• How handle miss (10s => HW, 1000s => SW)?
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 12 placed in 8 block cache:
  - Fully associative, direct mapped, 2-way set associative
  - S.A. Mapping = Block Number Modulo Number Sets
Q2: How Is a Block Found If It Is in the Upper Level?

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

Block Address

Tag | Index | Block offset
FA: No index
DM: Large index
Q3: Which Block Should be Replaced on a Miss?

- Easy for Direct Mapped
- S.A. or F.A.:
  - Random (large associativities)
  - LRU (smaller associativities)

<table>
<thead>
<tr>
<th>Associativity:</th>
<th>Size</th>
<th>2-way LRU</th>
<th>2-way Random</th>
<th>4-way LRU</th>
<th>4-way Random</th>
<th>8-way LRU</th>
<th>8-way Random</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>16 KB</td>
<td>5.18%</td>
<td>5.69%</td>
<td>4.67%</td>
<td>5.29%</td>
<td>4.39%</td>
<td>4.96%</td>
</tr>
<tr>
<td></td>
<td>64 KB</td>
<td>1.88%</td>
<td>2.01%</td>
<td>1.54%</td>
<td>1.66%</td>
<td>1.39%</td>
<td>1.53%</td>
</tr>
<tr>
<td></td>
<td>256 KB</td>
<td>1.15%</td>
<td>1.17%</td>
<td>1.13%</td>
<td>1.13%</td>
<td>1.12%</td>
<td>1.12%</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**
Example: 21064 Data Cache

- Index = 8 bits: 256 blocks = $8192/(32\times1)$
Writes in Alpha 21064

- No write merging vs. write merging in write buffer

4 entry, 4 word

16 sequential writes in a row
### Structural Hazard: Instruction and Data?

#### Separate Instruction Cache and Data Cache

<table>
<thead>
<tr>
<th>Size</th>
<th>Instruction Cache</th>
<th>Data Cache</th>
<th>Unified Cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 KB</td>
<td>3.06%</td>
<td>24.61%</td>
<td>13.34%</td>
</tr>
<tr>
<td>2 KB</td>
<td>2.26%</td>
<td>20.57%</td>
<td>9.78%</td>
</tr>
<tr>
<td>4 KB</td>
<td>1.78%</td>
<td>15.94%</td>
<td>7.24%</td>
</tr>
<tr>
<td>8 KB</td>
<td>1.10%</td>
<td>10.19%</td>
<td>4.57%</td>
</tr>
<tr>
<td>16 KB</td>
<td>0.64%</td>
<td>6.47%</td>
<td>2.87%</td>
</tr>
<tr>
<td>32 KB</td>
<td>0.39%</td>
<td>4.82%</td>
<td>1.99%</td>
</tr>
<tr>
<td>64 KB</td>
<td>0.15%</td>
<td>3.77%</td>
<td>1.35%</td>
</tr>
<tr>
<td>128 KB</td>
<td>0.02%</td>
<td>2.88%</td>
<td>0.95%</td>
</tr>
</tbody>
</table>

Relative weighting of instruction vs. data access
2-way Set Associative, Address to Select Word

Two sets of Address tags and data RAM

Use address bits to select correct DRAM

2:1 Mux for the way
Cache Performance

CPU time = (CPU execution clock cycles + Memory stall clock cycles) x clock cycle time

Memory stall clock cycles = (Reads x Read miss rate x Read miss penalty + Writes x Write miss rate x Write miss penalty)

Memory stall clock cycles = Memory accesses x Miss rate x Miss penalty
Cache Performance

\[ \text{CPUtime} = IC \times (\text{CPI}_{\text{execution}} + \text{Mem accesses per instruction} \times \text{Miss rate} \times \text{Miss penalty}) \times \text{Clock cycle time} \]

Misses per instruction = Memory accesses per instruction \times \text{Miss rate}

\[ \text{CPUtime} = IC \times (\text{CPI}_{\text{execution}} + \text{Misses per instruction} \times \text{Miss penalty}) \times \text{Clock cycle time} \]
Improving Cache Performance

• Average memory-access time = Hit time + Miss rate x Miss penalty (ns or clocks)
• Improve performance by:
  1. Reduce the miss rate,
  2. Reduce the miss penalty, or
  3. Reduce the time to hit in the cache.
Summary

• CPU-Memory gap is major performance obstacle for performance, HW and SW
• Take advantage of program behavior: locality
• Time of program still only reliable performance measure
• 4Qs of memory hierarchy