Lecture 20: Memory Hierarchy—Main Memory and Enhancing its Performance

Professor Alvin R. Lebeck
Computer Science 220
Fall 1999

Grinch-Like Stuff

• HW #4 Due November 12
• Projects...
• Finish reading Chapter 5
Review: Reducing Miss Penalty Summary

• Five techniques
  – Read priority over write on miss
  – Subblock placement
  – Early Restart and Critical Word First on miss
  – Non-blocking Caches (Hit Under Miss)
  – Second Level Cache

• Can be applied recursively to Multilevel Caches
  – Danger is that time to DRAM will grow with multiple levels in between

Review: Improving Cache Performance

1. Reduce the miss rate,
2. Reduce the miss penalty, or
3. Reduce the time to hit in the cache
   Fast hit times by small and simple caches
   Fast hits via avoiding virtual address translation
   Fast hits via pipelined writes
   Fast writes on misses via small subblocks
   Fast hits by using multiple ports
   Fast hits by using value prediction
Review: Cache Optimization Summary

<table>
<thead>
<tr>
<th>Technique</th>
<th>MR</th>
<th>MP</th>
<th>HT</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Larger Block Size</td>
<td>+</td>
<td>–</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Higher Associativity</td>
<td>+</td>
<td>–</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Victim Caches</td>
<td>+</td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Pseudo-Associative Caches</td>
<td>+</td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>HW Prefetching of Instr/Data</td>
<td>+</td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Compiler Controlled Prefetching</td>
<td>+</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>Compiler Reduce Misses</td>
<td>+</td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Priority to Read Misses</td>
<td>+</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Subblock Placement</td>
<td>+</td>
<td>+</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Early Restart &amp; Critical Word 1st</td>
<td>+</td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Non-Blocking Caches</td>
<td>+</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>Second Level Caches</td>
<td>+</td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Small &amp; Simple Caches</td>
<td>–</td>
<td>+</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Avoiding Address Translation</td>
<td>+</td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Pipelining Writes</td>
<td>+</td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

Main Memory Background

• Performance of Main Memory:
  – Latency: Cache Miss Penalty
    » Access Time: time between request and word arrives
    » Cycle Time: time between requests
  – Bandwidth: I/O & Large Block Miss Penalty (L2)

• Main Memory is DRAM: Dynamic Random Access Memory
  – Dynamic since needs to be refreshed periodically (8 ms)
  – Addresses divided into 2 halves (Memory as a 2D matrix):
    » RAS or Row Access Strobe
    » CAS or Column Access Strobe

• Cache uses SRAM: Static Random Access Memory
  – No refresh (6 transistors/bit vs. 1 transistor/bit)
  – Address not divided

• Size: DRAM/SRAM - 4-8
  Cost/Cycle time: SRAM/DRAM - 8-16
Main Memory Performance

- **Simple**: CPU, Cache, Bus, Memory same width (1 word)

- **Wide**: CPU/Mux 1 word; Mux/Cache, Bus, Memory N words (Alpha: 64 bits & 256 bits)

- **Interleaved**: CPU, Cache, Bus 1 word: Memory N Modules (4 Modules); example is word interleaved

```
Main Memory Performance

- **Timing model**
  - 1 to send address,
  - 6 access time, 1 to send data
  - Cache Block is 4 words

- **Simple M.P.** = 4 x (1+6+1) = 32
- **Wide M.P.** = 1 + 6 + 1 = 8
- **Interleaved M.P.** = 1 + 6 + 4x1 = 11
```

<table>
<thead>
<tr>
<th>Address</th>
<th>Bank 0</th>
<th>Address</th>
<th>Bank 1</th>
<th>Address</th>
<th>Bank 2</th>
<th>Address</th>
<th>Bank 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>1</td>
<td></td>
<td>2</td>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>5</td>
<td></td>
<td>6</td>
<td></td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>9</td>
<td></td>
<td>10</td>
<td></td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>13</td>
<td></td>
<td>14</td>
<td></td>
<td>15</td>
<td></td>
</tr>
</tbody>
</table>

© Alvin R. Lebeck 1999

CPS 220

Page 4
Independent Memory Banks

• Memory banks for independent accesses vs. faster sequential accesses
  – Multiprocessor
  – I/O
  – Miss under Miss, Non-blocking Cache
• Superbank: all memory active on one block transfer
• Bank: portion within a superbank that is word interleaved

<table>
<thead>
<tr>
<th>Superbank Number</th>
<th>Bank Number</th>
<th>Bank Offset</th>
</tr>
</thead>
</table>

How many banks?
• number banks \(\geq\) number clocks to access word in bank
  – For sequential accesses, otherwise will return to original bank before it has next word ready
• DRAM trend towards deep narrow chips.
  => fewer chips
  => harder to have banks of chips
  => put banks inside chips (internal banks)
Avoiding Bank Conflicts

- Lots of banks
  ```c
  int x[256][512];
  for (j = 0; j < 512; j = j+1)
    for (i = 0; i < 256; i = i+1)
      x[i][j] = 2 * x[i][j];
  ```
- Even with 128 banks, since 512 is multiple of 128, conflict
  - structural hazard
- SW: loop interchange or declaring array not power of 2
- HW: Prime number of banks
  - bank number = address mod number of banks
  - address within bank = address / number of banks
  - modulo & divide per memory access?
  - address within bank = address mod number words in bank (3, 7, 31)
  - bank number? easy if \(2^n\) words per bank

Fast Bank Number

- Chinese Remainder Theorem
  As long as two sets of integers \(a_i\) and \(b_i\) follow these rules
  \[
  b_i = x \mod a_i, \quad 0 \leq b_i < a_i, \quad 0 \leq x < a_0 \times a_1 \times a_2 \times \ldots
  \]
  and that \(a_i\) and \(a_j\) are co-prime if \(i \neq j\), then the integer \(x\) has only one solution (unambiguous mapping):
  - bank number = \(b_0\), number of banks = \(a_0\) (= 3 in example)
  - address within bank = \(b_1\), number of words in bank = \(a_1\) (= 8 in example)
  - \(N\) word address 0 to \(N-1\), prime no. banks, # of words is power of 2
Prime Mapping Example

<table>
<thead>
<tr>
<th>Seq. Interleaved Bank Number:</th>
<th>Modulo Interleaved Bank Number:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2</td>
<td>0 1 2</td>
</tr>
<tr>
<td>Address within Bank:</td>
<td></td>
</tr>
<tr>
<td>0 0 1 2</td>
<td>0 16 8</td>
</tr>
<tr>
<td>1 3 4 5</td>
<td>9 1 17</td>
</tr>
<tr>
<td>2 6 7 8</td>
<td>18 10 2</td>
</tr>
<tr>
<td>3 9 10 11</td>
<td>3 19 11</td>
</tr>
<tr>
<td>4 12 13 14</td>
<td>12 4 20</td>
</tr>
<tr>
<td>5 15 16 17</td>
<td>21 13 5</td>
</tr>
<tr>
<td>6 18 19 20</td>
<td>6 22 14</td>
</tr>
<tr>
<td>7 21 22 23</td>
<td>15 7 23</td>
</tr>
</tbody>
</table>

Fast Memory Systems: DRAM specific

- Multiple RAS accesses: several names (page mode)
  - 64 Mbit DRAM: cycle time = 100 ns, page mode = 20 ns

- New DRAMs to address gap; what will they cost, will they survive?
  - Synchronous DRAM: Provide a clock signal to DRAM, transfer synchronous to system clock
  - RAMBUS: reinvent DRAM interface (Intel will use it)
    - Each Chip a module vs. slice of memory
    - Short bus between CPU and chips
    - Does own refresh
    - Variable amount of data returned
    - 1 byte / 2 ns (500 MB/s per chip)
  - Cached DRAM (CDRAM): Keep entire row in SRAM
Main Memory Summary

• Big DRAM + Small SRAM = Cost Effective
  – Cray C-90 uses all SRAM (how many sold?)

• Wider Memory

• Interleaved Memory: for sequential or independent accesses

• Avoiding bank conflicts: SW & HW

• DRAM specific optimizations: page mode & Specialty DRAM, CDRAM
  – Niche memory or main memory?
    » e.g., Video RAM for frame buffers, DRAM + fast serial output

• IRAM: Do you know what it is?

Next Time

• Memory hierarchy and virtual memory