CS 61C: Great Ideas in Computer Architecture (Machine Structures)

Set-Associative Caches

Instructors:
Randy H. Katz
David A. Patterson

http://inst.eecs.Berkeley.edu/~cs61c/fa10

Agenda

• Cache Memory Recap
• Administrivia
• Technology Break
• Set Associative Caches
Agenda

- Cache Memory Recap
- Administrivia
- Technology Break
- Set Associative Caches

Recap: Components of a Computer
Recap: Typical Memory Hierarchy

- Take advantage of the **principle of locality** to present the user with as much memory as is available in the **cheapest** technology at the speed offered by the **fastest** technology.

### Recap:
- **Cache Performance and Average Memory Access Time (AMAT)**

- CPU time = IC × CPI × CC
  = IC × (CPI\textsubscript{ideal} + Memory-stall cycles) × CC

  \[
  \text{Memory-stall cycles} = \text{Read-stall cycles} + \text{Write-stall cycles}
  \]

  \[
  \text{Read-stall cycles} = \frac{\text{reads/program} \times \text{read miss rate} \times \text{read miss penalty}}{	ext{CPI}\textsubscript{stall}}
  \]

  \[
  \text{Write-stall cycles} = \frac{\text{writes/program} \times \text{write miss rate} \times \text{write miss penalty}}{	ext{write buffer stalls}}
  \]

- AMAT is the average time to access memory considering both hits and misses

  \[
  \text{AMAT} = \text{Time for a hit} + \text{Miss rate} \times \text{Miss penalty}
  \]
Improving Cache Performance

- Reduce the time to hit in the cache
  - E.g., Smaller cache, direct mapped cache, smaller blocks, special tricks for handling for writes
- **Reduce the miss rate**
  - E.g., Bigger cache, larger blocks
  - *More flexible placement (increase associativity)*
- Reduce the miss penalty
  - E.g., Smaller blocks or critical word first in large blocks, special tricks for handling for writes, faster/higher bandwidth memories
  - Use multiple cache levels

Sources of Cache Misses: The 3Cs

- **Compulsory** (cold start or process migration, 1st reference):
  - First access to block impossible to avoid; small effect for long running programs
  - Solution: increase block size (increases miss penalty; very large blocks could increase miss rate)
- **Capacity**:
  - Cache cannot contain all blocks accessed by the program
  - Solution: increase cache size (may increase access time)
- **Conflict** (collision):
  - Multiple memory locations mapped to the same cache location
  - Solution 1: increase cache size
  - Solution 2: increase associativity (may increase access time)
Reducing Cache Misses

- Allow more flexible block placement
- Direct mapped $\$: memory block maps to exactly one cache block
- Fully associative $\$: allow a memory block to be mapped to any cache block
- Compromise: divide $\$ into sets, each of which consists of n “ways” (n-way set associative) to place memory block
  - Memory block maps to unique set determined by index field and is placed in any of the n-ways of that set
  - Calculation: (block address) modulo (# sets in the cache)

Alternative Block Placement Schemes

- DM placement: mem block 12 in 8 block cache: only one cache block where mem block 12 can be found—(12 modulo 8) = 4
- SA placement: four sets x 2-ways (8 cache blocks), memory block 12 in set (12 mod 4) = 0; either element of the set
- FA placement: mem block 12 can appear in any cache blocks
Agenda

• Cache Memory Recap
• Administrivia
• Technology Break
• Set Associative Caches

Late Breaking Results from Project #3, Part I

sgemm (100000x500)
3 levels blocking, loop unrolling

Thanks TA Andrew!
Late Breaking Results from Project #3, Part I

sgemm (100000x500)
Performance Histogram

Thanks TA Andrew!

Administrivia

- Posted Optional Extra Logicsim “At Home” Lab this week
- Project 3: TLP+DLP+Cache Opt (Due 11/13)
- Project 4: Single Cycle Processor in Logicsim (by tonight)
  - Due Part 2 due Saturday 11/27
  - Face-to-Face in lab 12/2
- EC: Fastest Project 3 (due 11/29)
- Final Review: Mon Dec 6, 3 hrs, afternoon (TBD)
- Final: Mon Dec 13 8AM-11AM (TBD)
  - Like midterm: T/F, M/C, short answers
  - Whole Course: readings, lectures, projects, labs, hw
  - Emphasize 2nd half of 61C + midterm mistakes
Agenda

• Cache Memory Recap
• Administrivia
• Technology Break
• Set-Associative Caches
Example: 4 Word Direct Mapped $Worst Case Reference String$

- Consider the main memory word reference string

Start with an empty cache - all blocks initially marked as not valid

\[
\begin{array}{cccc}
0 & 4 & 0 & 4 \\
4 & 0 & 4 & 0 \\
0 & 4 & 0 & 4 \\
4 & 0 & 4 & 0 \\
\end{array}
\]

- 8 requests, 8 misses
- Ping pong effect due to conflict misses - two memory locations that map into the same cache block
Example: 2-Way Set Associative $\$, (4 words = 2 sets x 2 ways per set)

Q: How do we find it?

Use next 1 low order memory address bit to determine which cache set (i.e., modulo the number of sets in the cache)

Q: Is it there?

Compare all the cache tags in the set to the high order 3 memory address bits to tell if the memory block is in the cache

Main Memory

One word blocks
Two low order bits define the byte in the word (32b words)

Example: 4 Word 2-Way SA $\$, Same Reference String

- Consider the main memory word reference string

Start with an empty cache - all blocks initially marked as not valid

0 4 0 4 0 4 0 4
Example: 4 Word 2-Way SA $ 
Same Reference String

- Consider the main memory word reference string
  Start with an empty cache - all blocks initially marked as not valid

<table>
<thead>
<tr>
<th></th>
<th>0 miss</th>
<th>4 miss</th>
<th>0 hit</th>
<th>4 hit</th>
</tr>
</thead>
<tbody>
<tr>
<td>000 Mem(0)</td>
<td>000 Mem(0)</td>
<td>000 Mem(0)</td>
<td>000 Mem(0)</td>
<td></td>
</tr>
<tr>
<td>010 Mem(4)</td>
<td>010 Mem(4)</td>
<td>010 Mem(4)</td>
<td>010 Mem(4)</td>
<td></td>
</tr>
</tbody>
</table>

- 8 requests, 2 misses
- Solves the ping pong effect in a direct mapped cache due to conflict misses since now two memory locations that map into the same cache set can co-exist!

Example: Eight Block Cache with Different Organizations

Total size of $ in blocks is equal to number of sets x associativity. For fixed $ size, increasing associativity decreases number of sets while increasing number of elements per set. With eight blocks, an 8-way set-associative $ is same as a fully associative $.
Four-Way Set Associative Cache

• $2^8 = 256$ sets each with four ways (each with one block)

Range of Set Associative Caches

• For a fixed size cache, each increase by a factor of two in associativity doubles the number of blocks per set (i.e., the number or ways) and halves the number of sets – decreases the size of the index by 1 bit and increases the size of the tag by 1 bit
Range of Set Associative Caches

• For a fixed size cache, each increase by a factor of two in associativity doubles the number of blocks per set (i.e., the number or ways) and halves the number of sets – decreases the size of the index by 1 bit and increases the size of the tag by 1 bit

Costs of Set Associative Caches

• When miss occurs, which way’s block selected for replacement?
  – Least Recently Used (LRU): one that has been unused the longest
    • Must track when each way’s block was used relative to other blocks in the set
    • For 2-way SA $, one bit per set $\rightarrow$ set to 1 when a block is referenced; reset the other way’s bit (i.e., “last used”)
  – N-way set associative cache costs
    • N comparators (delay and area)
    • MUX delay (set selection) before data is available
    • Data available after set selection (and Hit/Miss decision).
      DM $\$: block is available before the Hit/Miss decision
      • Not possible to just assume a hit and continue and recover later if it was a miss
Cache Block Replacement Policies

- **Random Replacement**
  - Hardware randomly selects a cache item and throws it out

- **Least Recently Used**
  - Hardware keeps track of access history
  - Replace the entry that has not been used for the longest time
  - For 2-way set-associative cache, need one bit for LRU replacement

- **Example of a Simple “Pseudo” LRU Implementation**
  - Assume 64 Fully Associative entries
  - Hardware replacement pointer points to one cache entry
  - Whenever access is made to the entry the pointer points to:
    - Move the pointer to the next entry
  - Otherwise: do not move the pointer

Benefits of Set Associative Caches

- Choice of DM $ or SA $ depends on the cost of a miss versus the cost of implementation

- Largest gains are in going from direct mapped to 2-way (20%+ reduction in miss rate)
• Three sources of misses (SPEC2000 integer and floating-point benchmarks)
  – Compulsory misses 0.006%; not visible
  – Capacity misses, function of cache size
  – Conflict portion depends on associativity and cache size

Two Machines’ Cache Parameters

<table>
<thead>
<tr>
<th></th>
<th>Intel Nehalem</th>
<th>AMD Barcelona</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1 cache organization &amp; size</td>
<td>Split I$ and D$; 32KB for each per core; 64B blocks</td>
<td>Split I$ and D$; 64KB for each per core; 64B blocks</td>
</tr>
<tr>
<td>L1 associativity</td>
<td>4-way (I), 8-way (D) set assoc.; ~LRU replacement</td>
<td>2-way set assoc.; LRU replacement</td>
</tr>
<tr>
<td>L1 write policy</td>
<td>write-back, write-allocate</td>
<td>write-back, write-allocate</td>
</tr>
<tr>
<td>L2 cache organization &amp; size</td>
<td>Unified; 256MB (0.25MB) per core; 64B blocks</td>
<td>Unified; 512KB (0.5MB) per core; 64B blocks</td>
</tr>
<tr>
<td>L2 associativity</td>
<td>8-way set assoc.; ~LRU</td>
<td>16-way set assoc.; ~LRU</td>
</tr>
<tr>
<td>L2 write policy</td>
<td>write-back</td>
<td>write-back</td>
</tr>
<tr>
<td>L3 cache organization &amp; size</td>
<td>Unified; 8192KB (8MB) shared by cores; 64B blocks</td>
<td>Unified; 2048KB (2MB) shared by cores; 64B blocks</td>
</tr>
<tr>
<td>L3 associativity</td>
<td>16-way set assoc.</td>
<td>32-way set assoc.; evict block shared by fewest cores</td>
</tr>
<tr>
<td>L3 write policy</td>
<td>write-back, write-allocate</td>
<td>write-back, write-allocate</td>
</tr>
</tbody>
</table>
Summary

• **Name of the Game: Reduce Cache Misses**
  – Two different memory blocks mapping to same cache block could knock each other out as program bounces from one memory location to the next

• **One way to do it: set-associativity**
  – Memory block maps into more than one cache block
  – N-way: n possible places in the cache to hold a given memory block
  – N-way Cache of \(2^{N+M}\) blocks: \(2^N\) ways \(\times\) \(2^M\) sets