CS152 – Computer Architecture and Engineering

Lecture 14 – Cache II

2004-10-19

Dave Patterson
(www.cs.berkeley.edu/~patterson)

John Lazzaro
(www.cs.berkeley.edu/~lazzaro)

www-inst.eecs.berkeley.edu/~cs152/
Today’s Lecture - Caches Reloaded

- Classic cache designs
- Cache misses and performance
- Writing to caches
Last Time: Fully Associative Cache

Cache Tag (27 bits)  Byte Select

Block # ("Tags")

Cache Data
Holds 4 blocks

Ex: 0x01

Byte 31  ...  Byte 1  Byte 0
Byte 31  ...  Byte 1  Byte 0

Valid Bit

Return bytes of "hit" cache line

Hit

Ideal, but expensive ...

Cal
Building a cache with one comparator

Blocks of a certain color may only appear in one line of the cache.

32-bit Memory Address

<table>
<thead>
<tr>
<th>Block #</th>
<th>32-byte blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

2\(^{27} - 1\)

Which block? Color Byte #

- 25 bits
- 2 bits
- 5 bits

Cache index
Example: A Direct Mapped Cache

31  7  6  5  4  0

<table>
<thead>
<tr>
<th>Cache Tag (25 bits)</th>
<th>Index</th>
<th>Byte Select</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Cache Tags

=

Hit

PowerPC 970: 64K direct-mapped Level-1 I-cache

Valid Bit

Cache Data

Ex: 0x01

Ex: 0x00

Return bytes of "hit" cache line

CS 152 L14 Cache II ()

UC Regents Fall 2004 © UCB
The limits of direct-mapped caches ...

Hybrid Design: Set Associative Cache

“N-way” set associative -- N is number of blocks for each color

<table>
<thead>
<tr>
<th>Cache Tag (26 bits)</th>
<th>Index</th>
<th>Byte Select (4 bits)</th>
</tr>
</thead>
</table>

Ex: 0x01

Cache Tag (26 bits)
Index
Byte Select (4 bits)

Cache Data  Valid  Cache Tags
Cache Block
Cache Block
16 bytes

=  
=  

Hit Left
Hit Right

Return bytes of “hit” set member

PowerPC 970: 32K 2-way set associative L1 D-cache

16 bytes

Cache Data  Valid  Cache Blocks
Cache Block
Cache Block
16 bytes

Cal
The benefits of set-associativity ...

Q. What costs (over direct mapped) for this benefit?

What if both regions have same block color?

Homework 3 due 10/20 (Wednesday), 283 Soda, in CS 152 box at 5 PM

Lab 3 final demo on 10/22 (Friday)

Lab 3 report due: Monday 10/25 11:59 PM
Cache Misses & Performance
Recall: Performance Equation

<table>
<thead>
<tr>
<th>Seconds</th>
<th>Instructions</th>
<th>Cycles</th>
<th>Seconds</th>
<th>Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program</td>
<td>Program</td>
<td>Instruction</td>
<td>Cycle</td>
<td>Cycle</td>
</tr>
</tbody>
</table>

Earlier, computed from ...

**Machine CPI**

Assumes a constant memory access time.

True CPI depends on the Average Memory Access Time (AMAT) for Inst & Data

$$AMAT = \text{Hit Time} + (\text{Miss Rate} \times \text{Miss Penalty})$$

Goal: Reduce AMAT

Beware! Improving one term may hurt other terms, and increase AMAT!

True CPI = Ideal CPI + Memory Stall Cycles. See Section 7.3, COD/3e for details.
One type of cache miss: Conflict Miss

N blocks of same color in use at once, but cache can only hold M < N of them

Solution: Increase M (Associativity)

Other Solutions

Increase number of cache lines (# blocks in cache)

Q. Why does this help?

Add a small “victim cache” that holds blocks recently removed from the cache.

Q. Why does this help?

If hit time increases, AMAT may go up!

AMAT = Hit Time + (Miss Rate x Miss Penalty)
Other causes of cache misses ...

Capacity Misses
Cache cannot contain all blocks accessed by the program
Solution: Increase size of the cache

Compulsory Misses
First access of a block by a program
Mostly unavoidable
Solution: Prefetch blocks (via hardware, software)

Miss rates (relative)

Cache Size (KB)

Also "Coherency Misses": other processes update memory
Thinking about cache miss types ...

What kind of misses happen in a fully associative cache of infinite size?

A. Compulsory misses. Must bring each block into cache.

In addition, what kind of misses happen in a finite-sized fully associative cache?

A. Capacity misses. Program may use more blocks than can fit in cache.

In addition, what kind of misses happen in a set-associative or direct-map cache?

A. Conflict misses.

(all questions assume the replacement policy used is considered “optimal”)
Cache Block Replacement Strategy

After a cache read miss, if there are no empty cache blocks, which block should be removed from the cache?

The Least Recently Used (LRU) block? Appealing, but hard to implement.

A randomly chosen block? Easy to implement, how well does it work?

Miss Rate for 2-way Set Associative Cache

<table>
<thead>
<tr>
<th>Size</th>
<th>Random</th>
<th>LRU</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 KB</td>
<td>5.7%</td>
<td>5.2%</td>
</tr>
<tr>
<td>64 KB</td>
<td>2.0%</td>
<td>1.9%</td>
</tr>
<tr>
<td>256 KB</td>
<td>1.17%</td>
<td>1.15%</td>
</tr>
</tbody>
</table>

Also, try other LRU approx.
Why Hit Time Matters: CPU Interface!

Hit time is directly tied to clock rate of CPU.

If left unchecked, it increases when cache size and associativity increases.

Note that XScale pipelines both instruction and data caches, adding stages to the CPU pipeline.
Writes and Caches
## Write Policy: Write-Through vs Write-Back

<table>
<thead>
<tr>
<th>Policy</th>
<th>Write-Through</th>
<th>Write-Back</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Data written to cache block also written to lower-level memory</td>
<td>Write data only to the cache Update lower level when a block falls out of the cache</td>
</tr>
<tr>
<td>Debug</td>
<td>Easy</td>
<td>Hard</td>
</tr>
<tr>
<td>Do read misses produce writes?</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Do repeated writes make it to lower level?</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

Additional option -- let writes to an un-cached address allocate a new cache line (“write-allocate”).
Write Buffers for Write-Through Caches

Holds data awaiting write-through to lower level memory

Q. Why a write buffer?
A. So CPU doesn’t stall

Q. Why a buffer, why not just one register?
A. Bursts of writes are common.

Q. Are Read After Write (RAW) hazards an issue for write buffer?
A. Yes! Drain buffer before next read, or check write buffers.
Cache Design: Datapath + Control

Most design errors come from incorrect specification of state machine behavior!
Common bugs: Stalls, Block replacement, Write buffer

State Machine

To CPU

Control

To Lower Level Memory

Blocks

Addr

Din

Dout

To Lower Level Memory

Tags

Addr

Din

Dout
Lab 4: Why some groups have problems

Example bug: When does a write go to the cache?

A1. When address is already in the cache.

Issue: Must check tag before writing, or else may overwrite the wrong address!

Options: Stall and do tag check, or pipeline check.

A2. Always: we do allocate on write.
Conclusions

- The cache design spectrum: from direct mapped to fully associative.
- AMAT (Ave. Memory Access Time) = Hit Time + (Miss Rate x Miss Penalty)
- Cache misses: conflict, capacity, compulsory, and coherency.
- Lab 4 and project bugs usually from cache specification errors.