2005-3-8

John Lazzaro

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

TAs: Ted Hong and David Marquardt

www-inst.eecs.berkeley.edu/~cs152/
Last Time: Locality encourages caching

- **Bad**
- **Temporal Locality**
- **Spatial Locality**

Today ... Caches Reloaded

- Cache misses and performance: how do we size the cache?
- Practical cache design: a state machine and a controller.
- Write buffers and caches
- The cache-DRAM interface
Recall: Color-coding main memory

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

32-bit Memory Address

<table>
<thead>
<tr>
<th>31</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Which block?</td>
<td>Color</td>
<td>Byte #</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25 bits</td>
<td>2 bits</td>
<td>5 bits</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Cache index

Block #

\[
\begin{array}{c}
0 \\
1 \\
2 \\
3 \\
4 \\
5 \\
6 \\
7 \\
\vdots \\
2^{27} - 1 \\
\end{array}
\]

32-byte blocks
Recall: A Direct Mapped Cache

PowerPC 970: 64K direct-mapped Level-1 I-cache
Recall: Set Associative Cache

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

Cache Tag (26 bits)  Index (2 bits)  Byte Select (4 bits)

Cache Data  Valid  Cache Tags  Cache Tags  Valid  Cache Data

Cache Block  16 bytes  Cache Block  16 bytes  Cache Block

Ex: 0x01

Hi Right

Hit Right

Return bytes of "hit" set member

=  

=  

Cache block halved to keep * cached bits constant.

PowerPC 970: 32K 2-way set associative L1 D-cache
Cache Misses & Performance
Recall: Performance Equation

<table>
<thead>
<tr>
<th>Seconds Program</th>
<th>Instructions Program</th>
<th>Cycles Instruction</th>
<th>Seconds Cycle</th>
</tr>
</thead>
</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 = Hit Time + (Miss Rate x Miss Penalty)

Goal: Reduce AMAT

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

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

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)

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”)
5. On **Friday 3/11** in lab section, you will demonstrate the processor running on the Calinx board. Unlike the 3/4 checkoff, this demo will test all facets of the processor (including the issues discussed in Problem 4). During this demo, the TA will provide you with secret test code. If you are able to pass these tests on your first try, you will receive bonus points. You can also receive bonus points if you fix your processor to pass the tests within your section time. If your processor is not fixed by the end of section, your TA will provide source for the secret test code, for use in your weekend debugging sessions.
Admin: Midterm coming soon ...

**Week from Today**

<table>
<thead>
<tr>
<th>Day</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>Midterm Review Session in Class</td>
</tr>
<tr>
<td>W</td>
<td>Midterm I: 6PM to 9PM, 320 Soda (not 310!) (note: no class 12:30-2)</td>
</tr>
</tbody>
</table>

**Midterm is one week from Thursday, in evening, no class that day.**

**Will cover format, material, and ground rules for test in this session.**
Practical Cache Design
Cache Design: Datapath + Control

Datapath for performance, control for correctness. Most design errors come from incorrect specification of state machine behavior!

Red text will highlight state machine requirements ...
Recall: State Machine Design ...

Cache controller state machines like this, but more states, and perhaps several connected machines ...
Issue #1: Control for CPU interface ....

For reads, your state machine must:

1. sense REQ
2. latch Addr
3. create Wait
4. put Data Out on the bus.

An example interface ... there are other possibilities.
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>

Part of your state machine decides which block to replace.
Issue #3: High performance block fetch

With proper memory layout, one row access delivers entire cache block to the sense amp.

Two state machine challenges: (1) Bring in the word requested by CPU with lowest latency
(2) Bring in rest of cache block ASAP

12-bit row address input

1 of 4096 decoder

4096 rows

33,554,432 usable bits
(tester found good bits in bigger array)

8196 bits delivered by sense amps

Select requested bits, send off the chip

2048 columns

Each column 4 bits deep
Issue #3 (continued): DRAM Burst Reads

One request ...

DRAM can be set up to request an N byte region starting at an arbitrary N+k within region

Many returns ...

State machine challenges: (1) setting up correct block read mode (2) delivering correct word direct to CPU (3) putting all words in cache in right place.
Writes and Caches
## Issue #4: When to write to lower level ...

<table>
<thead>
<tr>
<th>Policy</th>
<th>Write-Through</th>
<th>Write-Back</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data written to cache block also written to lower-level memory</td>
<td>Write data only to the cache</td>
<td>Update lower level when a block falls out of the cache</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>

**State machine design**

(1) **Write-back** puts most write logic in cache-miss machine.  
(2) **Write-through** isolates writing in its own state machine.
Issue #5: Write-back DRAM Burst Writes

State machine challenges: (1) putting cache block into correct location (2) what if a read or write wants to use DRAM before the burst is complete? Must stall ...
If we choose write-through ...

<table>
<thead>
<tr>
<th>Policy</th>
<th>Write-Through</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data written to cache block also written to lower-level memory</td>
<td>No</td>
</tr>
</tbody>
</table>

Do read misses produce writes?  | No  |
Do repeated writes make it to lower level? | Yes |

State machine design issue: handling writes without stalling the machine until the written word is safely in the lower level (DRAM)
Issue #6: Avoid write-through write stalls

Solution: add a “write buffer” to cache datapath

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.

On reads, state machine checks cache and write buffer -- what if word was removed from cache before lower-level write? On writes, state machine stalls for full write buffer, handles write buffer duplicates.
Issue #7: Optimizing the hit time ...

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.

State machine design issue: pipelining cache control!
Cache Design: Debugging

Just like pipelining lab, you will write assembly language programs to test the "edge cases" that will find bugs in your state machine ... TAs have their own test suite for checkoffs.
Recall: Fall 04 Lab 4 TA reflections

Common bugs: Stalls, Block replacement, Write buffer

Example: Comprehensive test rigs seen as a “checkoff item” for Lab report, done last. Actual debugging proceeds in haphazard, painful way.

A Better Way: One group spent 10 hours up front writing a cache test module. Brandon “The best cache testing I’ve ever seen”. They finished on time. An example of working smart.
Common bug: When to write to cache?

A1. If no write-allocate ... 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.
Lectures: Coming up next ...

Last new-material lecture before midterm.