

CS 61C: Great Ideas in Computer Architecture

Lecture 13 – Cache Basics (Cache II)

Instructors:
Mike Franklin
Dan Garcia

<http://inst.eecs.berkeley.edu/~cs61c/fa11>

Fall 2011 – Lecture #13 1

In the News...

The importance of using *the right* performance metrics



The film is based on the best-selling 2003 book in Michael Lewis chronicled the data-driven resurgence of the Oakland A's engineered by A's general manager Billy Beane and DePodesta, who used computer analysis to identify undervalued players.

“We had a completely new set of metrics that bore no resemblance to anything you’d seen. We didn’t solve baseball. But we reduced the inefficiency of our decision making.”

www.datacenterknowledge.com/archives/2011/09/23/the-lessons-of-moneyball-for-big-data-analysis/

9/28/11 Fall 2011 – Lecture #13 2

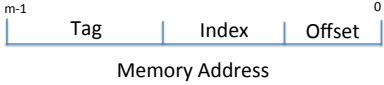
Agenda

- Review – Direct Mapped Caches
- Handling Writes (updates)
- Handling Hits and Misses
- Performance Considerations

9/28/11 Fall 2011 – Lecture #13 3

TIO: Mapping the Memory Address

- Lowest bits of address (*Offset*) determine *which byte within a block* it refers to.
- Full address format (m-bit address):



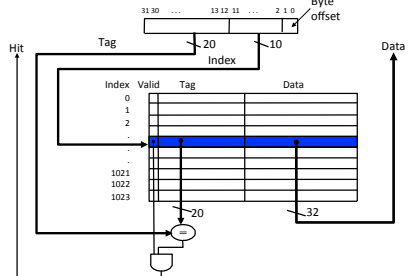
Memory Address

- n-bit Offset: a cache block is how many bytes?
- n-bit Index: cache has how many blocks?

9/28/11 Fall 2011 – Lecture #13 4

Direct Mapped Cache Example

One word (4 Byte) blocks, cache size = 1K words (or 4KB)



What kind of locality are we taking advantage of?

9/28/11 Fall 2011 – Lecture #13 5

Direct Mapped Cache (1Byte words)

- Consider the sequence of memory address accesses

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

0000 0001 0010 0011 0100 0011 0100 1111

0 1 2 3 4 3 4 15

0 miss

00	Mem(0)
00	Mem(1)

1 miss

00	Mem(0)
00	Mem(1)

2 miss

00	Mem(0)
00	Mem(1)
00	Mem(2)

3 miss

00	Mem(0)
00	Mem(1)
00	Mem(2)
00	Mem(3)

Time →

4 miss

00	Mem(0)
00	Mem(1)
00	Mem(2)
00	Mem(3)

3 hit!!!

01	Mem(4)
00	Mem(1)
00	Mem(2)
00	Mem(3)

4 hit!!!

01	Mem(4)
00	Mem(1)
00	Mem(2)
00	Mem(3)

15 miss

01	Mem(4)
00	Mem(1)
00	Mem(2)
00	Mem(3)

Time →

- 8 requests, 2 hits, 6 misses = 25% hit rate

9/28/11 Fall 2011 – Lecture #13 6

Taking Advantage of Spatial Locality

- Let cache block hold more than one byte (say, two)

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

0 1 2 3 4 3 4 15
0000 0001 0010 0011 0100 0011 0100 1111

0 miss 1 hit 2 miss

00	Mem(1)	Mem(0)
00	Mem(1)	Mem(0)
00	Mem(1)	Mem(0)
00	Mem(3)	Mem(2)

3 hit 4 miss 3 hit

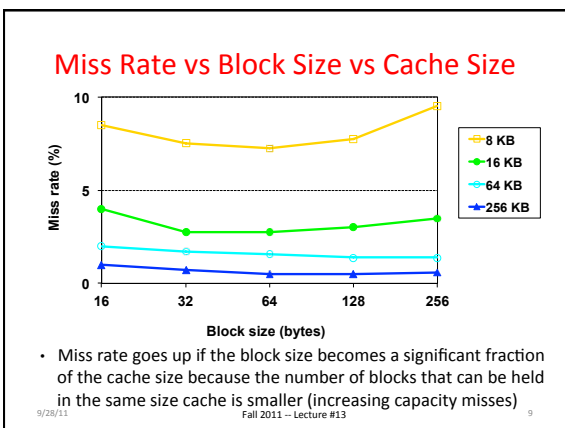
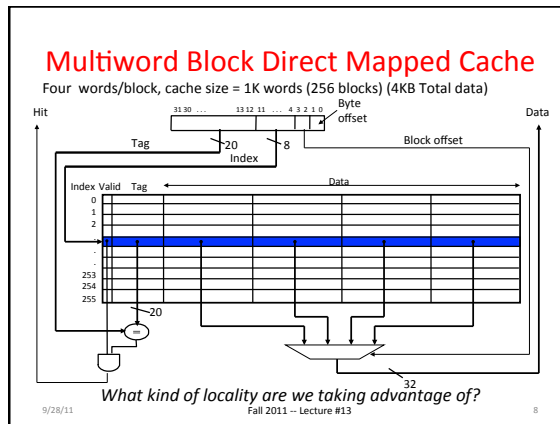
00	Mem(1)	Mem(0)
00	Mem(3)	Mem(2)
01	Mem(5)	Mem(4)
00	Mem(3)	Mem(2)

4 hit 15 miss

01	Mem(5)	Mem(4)
00	Mem(3)	Mem(2)
01	Mem(5)	Mem(4)
00	Mem(3)	Mem(2)

• 8 requests, 4 hits, 4 misses = 50% hit rate!

9/28/11 Fall 2011 -- Lecture #13 7



Cache-Memory Consistency? (1/2)

- Need to make sure cache and memory are consistent (i.e., know about all updates)

1) **Write-Through Policy:** write cache and write *through* the cache to memory

- Every write eventually gets to memory
- Too slow, so include Write Buffer to allow processor to continue once data in Buffer, Buffer updates memory in parallel to processor

9/28/11 Fall 2011 -- Lecture #13 10

Cache-Memory Consistency? (2/2)

2) **Write-Back Policy:** write only to cache and then write cache block *back* to memory when evict block from cache

- Writes collected in cache, only single write to memory per block
- Include bit to see if wrote to block or not, and then only write back if bit is set
 - Called "Dirty" bit (writing makes it "dirty")

9/28/11 Fall 2011 -- Lecture #13 11

Handling Cache Hits

- Read hits (I\$ and D\$)
 - Hits are good in helping us go fast
 - Misses are bad/slow us down
- Write hits (D\$ only)
 - Require cache and memory to be consistent
 - Write-through:** Always write the data into the cache block and the next level in the memory hierarchy
 - Writes run at the speed of next level in memory hierarchy -- so slow! -- or can use a write buffer and stall only if the write buffer is full
 - Allow cache and memory to be inconsistent
 - Write-back:** Write the data only into the cache block (cache block written back to next level in memory hierarchy when it is "evicted")
 - Need a **dirty bit** for each data cache block to tell if it needs to be written back to memory when evicted -- can use a **write buffer** to help "buffer" write-backs of dirty blocks

9/28/11 Fall 2011 -- Lecture #13 12

Handling Cache Misses (Single Word Blocks)

- Read misses (I\$ and D\$)
 - Stall execution, fetch the block from the next level in the memory hierarchy, install it in the cache and send requested word to processor, then resume execution
- Write misses (D\$ only)
 - Stall execution, fetch the block from next level in the memory hierarchy, install it in cache (may involve evicting a dirty block if using write-back), write the word from processor to cache, resume
 - or
 - **Write allocate:** just write word into the cache updating both tag and data; no need to check for cache hit, no need to stall
 - or
 - **No-write allocate:** skip the cache write (but must invalidate cache block since it will now hold stale data) and just write the word to write buffer (and eventually to the next memory level); no need to stall if write buffer isn't full (write through only)

Fall 2011 -- Lecture #13 13

Handling Cache Misses (Multiword Block Considerations)

- Read misses (I\$ and D\$)
 - Processed the same as for single word blocks – a miss returns the entire block from memory
 - Miss penalty grows as block size grows
 - **Early restart:** processor resumes execution as soon as the requested word of the block is returned
 - **Requested word first:** requested word is transferred from the memory to the cache (and processor) first
 - Nonblocking cache – allows the processor to continue to access cache while cache is handling an earlier miss
- Write misses (D\$)
 - If using write allocate must first fetch block from memory and then write word to block (or could end up with a “garbled” block in the cache.
 - E.g., for 4 word blocks, a new tag, one word of data from the new block, and three words of data from the old block)

9/28/11 Fall 2011 -- Lecture #13 14

“And In Conclusion..”

- Direct Mapped Cache – Each block in memory maps to one block in the cache.
 - Index to determine which block.
 - Offset to determine which byte within block
 - Tag to determine if it's the right block.
- AMAT to measure cache performance
- Cache can have major impact on CPI
- Multi-level cache can help

9/28/11 Fall 2011 -- Lecture #13 15

Peer Instruction

A. For a given cache size: a larger block size can cause a lower hit rate than a smaller one.

B. If you know your computer's cache size, you can often **make your code run faster**.

C. Memory hierarchies take advantage of **spatial locality** by keeping the most recent data items **closer** to the processor.

ABC
1: FFF
1: FFT
2: FTF
2: FTT
3: TFF
3: TTF
4: TTF
5: TTT

Fall 2011 -- Lecture #13

Peer Instruction Answer

A. Yes – if the block size gets too big, fetches become more expensive and the big blocks force out more useful data.

B. Certainly! That's call “tuning”

C. “Most Recent” items ⇒ **Temporal locality**

A. For a given cache size: a larger block size can cause a lower hit rate than a smaller one.

B. If you know your computer's cache size, you can often **make your code run faster**.

C. Memory hierarchies take advantage of **spatial locality** by keeping the most recent data items **closer** to the processor.

ABC
1: FFF
1: FFT
2: FTF
2: FTT
3: TFF
3: TTF
4: TTF
5: TTT

Fall 2011 -- Lecture #13