

CS 61C: Great Ideas in Computer Architecture

Lecture 14 –Cache Performance (Cache III)

Instructors:

Mike Franklin

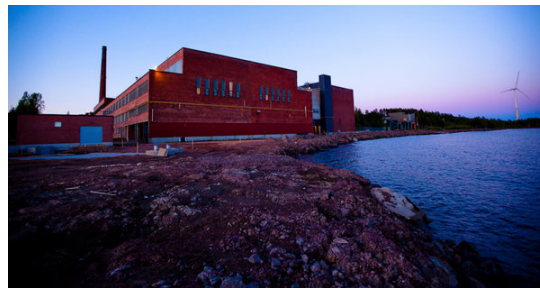
Dan Garcia

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

Fall 2011 -- Lecture #14

1

In the News... Major Expansion of World's Data Centers



- The next great expansion of the world's digital infrastructure is under way in developing markets like those of China, Brazil and Argentina.
- Growth expected to match levels from economy's "boom years" despite global slowdown

<http://www.nytimes.com/2011/09/28/technology/worlds-data-centers-expected-to-grow-survey-says.html>

9/28/11

Fall 2011 -- Lecture #14

2

Agenda

- Review – Direct Mapped Caches
- Measuring Performance
- Multi-Level Caches
- Associative Caches
- Cache Wrap Up

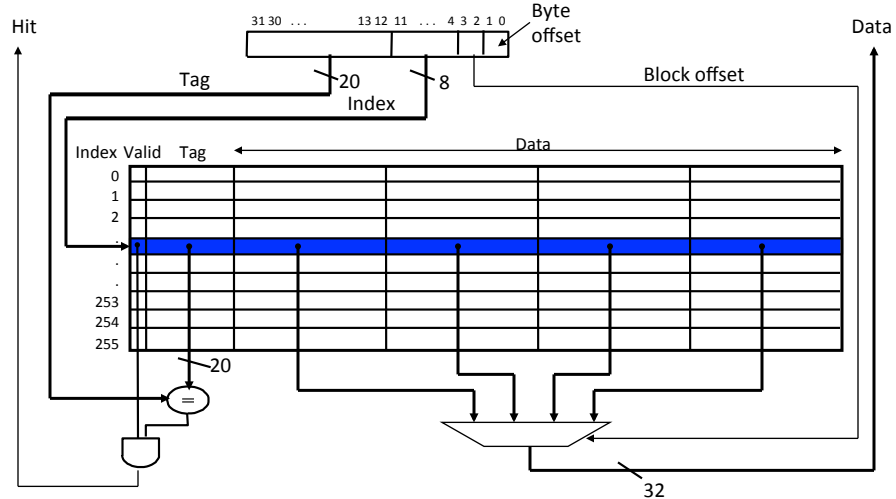
9/28/11

Fall 2011 -- Lecture #14

3

Review: Multiword Block Direct Mapped Cache

Four words/block, cache size = 1K words (256 blocks) (4KB Total data)



9/28/11

Fall 2011 -- Lecture #14

4

Measuring Performance

- Computers use a clock to determine when events take place within hardware
- *Clock cycles* – discrete time intervals
 - a.k.a. clocks, cycles, clock periods, clock ticks
- *Clock rate* or *clock frequency* – clock cycles per second (inverse of clock cycle time)
- 3 GigaHertz clock rate
 - => clock cycle time = $1/3 \times 10^9$ seconds
 - clock cycle time = 333 picoseconds (ps)

9/28/11

Fall 2011 -- Lecture #14

5

CPU Performance Factors

- But a program executes instructions
- Time = $\frac{\text{Seconds}}{\text{Program}}$

$$= \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock Cycle}}$$
- 1st term called *Instruction Count*
- 3rd term is *Clock Cycle Time* (1 / Clock rate)
- 2nd term abbreviated *CPI* for average *Clock cycles Per Instruction*
- Why CPI = 1? Why CPI > 1? Can CPI be < 1?

9/28/11

Fall 2011 -- Lecture #14

6

Average Memory Access Time (AMAT)

- Average Memory Access Time (AMAT) is the average to access memory considering both hits and misses

$$\text{AMAT} = \text{Time for a hit} + \text{Miss rate} \times \text{Miss penalty}$$
- What is the AMAT for a processor with a 200 psec clock, a miss penalty of 50 clock cycles, a miss rate of 0.02 misses per instruction and a cache access time of 1 clock cycle?

$$1 + 0.02 \times 50 = 2 \text{ clock cycles}$$

$$\text{Or } 2 \times 200 = 400 \text{ psecs}$$

- Potential impact of much larger cache on AMAT?
 - Lower Miss rate
 - Longer Access time (Hit time): smaller is faster
 Increase in hit time will likely add another stage to the pipeline
 At some point, increase in hit time for a larger cache may overcome the improvement in hit rate, yielding a decrease in performance

9/28/11

Fall 2011 -- Lecture #14

7

Measuring Cache Performance – Effect on CPI

- Assuming cache hit costs are included as part of the normal CPU execution cycle, then

$$\text{CPU time} = \text{IC} \times \text{CPI} \times \text{CC}$$

$$= \text{IC} \times (\underbrace{\text{CPI}_{\text{ideal}} + \text{Average Memory-stall cycles}}_{\text{CPI}_{\text{stall}}}) \times \text{CC}$$

- A simple model for Memory-stall cycles

$$\text{Memory-stall cycles} = \text{accesses/instruction} \times \text{miss rate} \times \text{miss penalty}$$

- This ignores extra costs of write misses – which were described previously

9/28/11

Fall 2011 -- Lecture #14

8

Impacts of Cache Performance

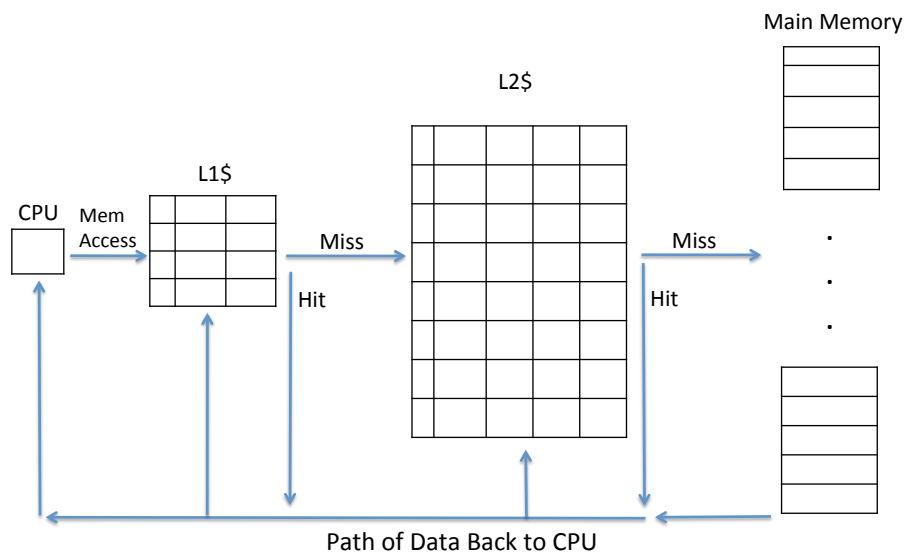
- Relative \$ penalty increases as processor performance improves (faster clock rate and/or lower CPI)
 - Memory speed unlikely to improve as fast as processor cycle time. When calculating CPI_{stall} , cache miss penalty is measured in processor clock cycles needed to handle a miss
 - Lower the CPI_{ideal} , more pronounced impact of stalls
- Processor with a CPI_{ideal} of 2, a 100 cycle miss penalty, 36% load/store instr's, and 2% I\$ and 4% D\$ miss rates
 - Memory-stall cycles = $2\% \times 100 + 36\% \times 4\% \times 100 = 3.44$
 - So $CPI_{stalls} = 2 + 3.44 = 5.44$
 - More than twice the CPI_{ideal} !
- What if the CPI_{ideal} is reduced to 1?
- What if the D\$ miss rate went up by 1%?

9/28/11

Fall 2011 -- Lecture #14

9

Multiple Cache Levels



9/28/11

Fall 2011 -- Lecture #14

10

Multiple Cache Levels

- With advancing technology, have more room on die for bigger L1 caches and for second level cache – normally a **unified** L2 cache (i.e., it holds both instructions and data,) and in some cases even a unified L3 cache

- New AMAT Calculation:

$$\mathbf{AMAT} = \mathbf{L1\ Hit\ Time} + \mathbf{L1\ Miss\ Rate} * \mathbf{L1\ Miss\ Penalty}$$

$$\mathbf{L1\ Miss\ Penalty} = \mathbf{L2\ Hit\ Time} + \mathbf{L2\ Miss\ Rate} * \mathbf{L2\ Miss\ Penalty}$$

and so forth (final miss penalty is Main Memory access time)

New AMAT Example

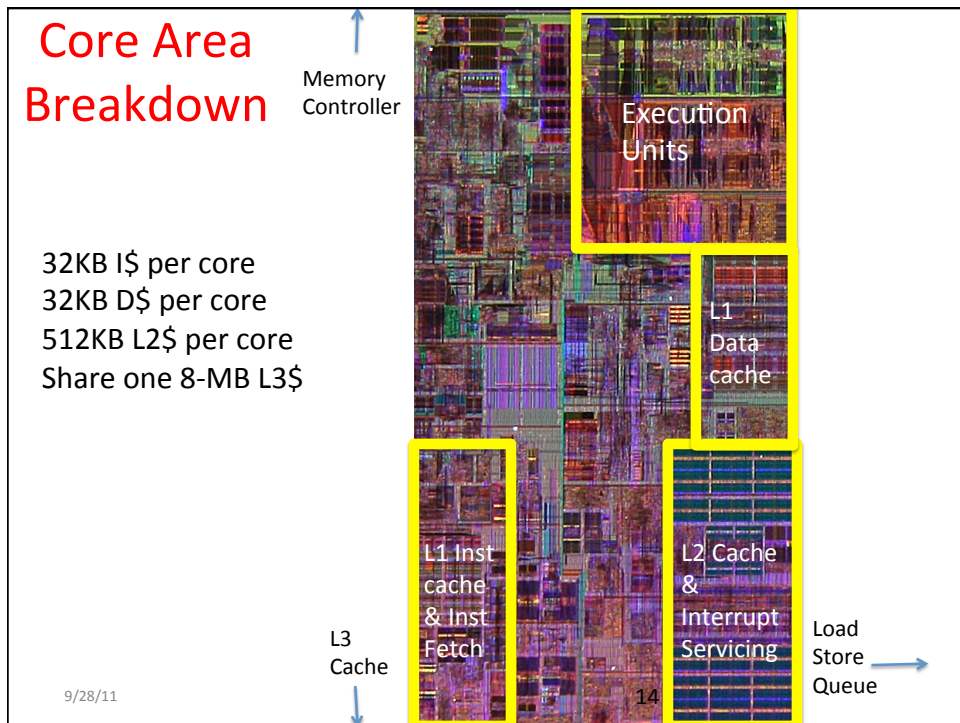
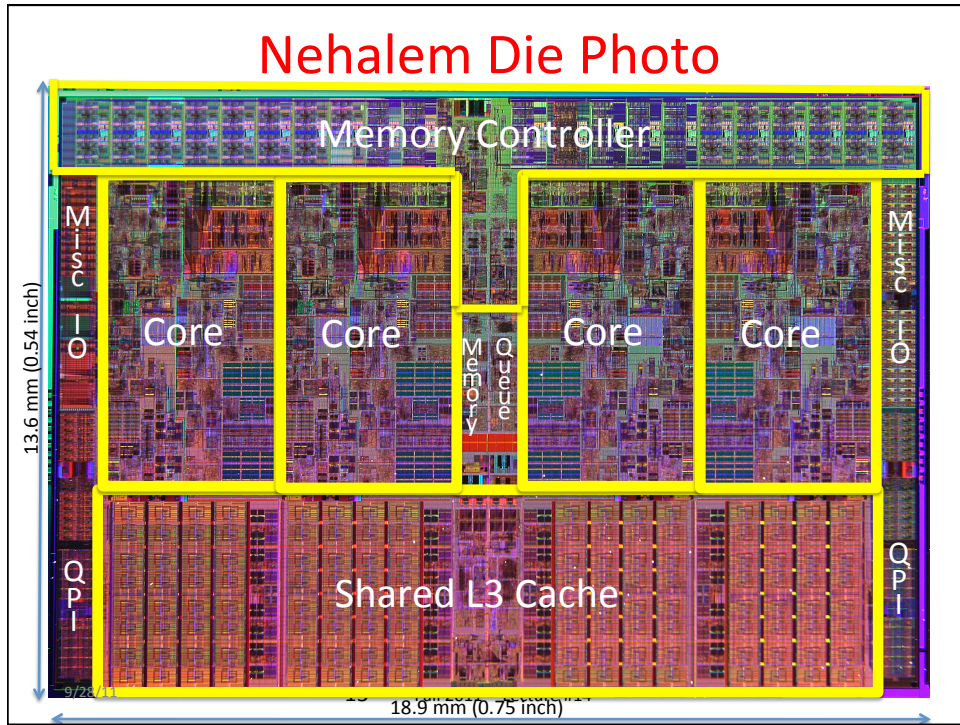
- 1 cycle L1 Hit Time, 2% L1 Miss Rate, 5 cycle L2 Hit Time, 5% L2 Miss Rate.
- 100 cycle Main Memory access time

- No L2 Cache:

$$\mathbf{AMAT} = 1 + .02 * 100 = 3$$

- With L2 Cache:

$$\mathbf{AMAT} = 1 + .02 * (5 + .05 * 100) = 1.2!$$



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)**

Fall 2011 -- Lecture #14

15

Reducing Cache Misses

- Allow more flexible block placement in cache
- *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)

Fall 2011 -- Lecture #14

16

Alternative Block Placement Schemes



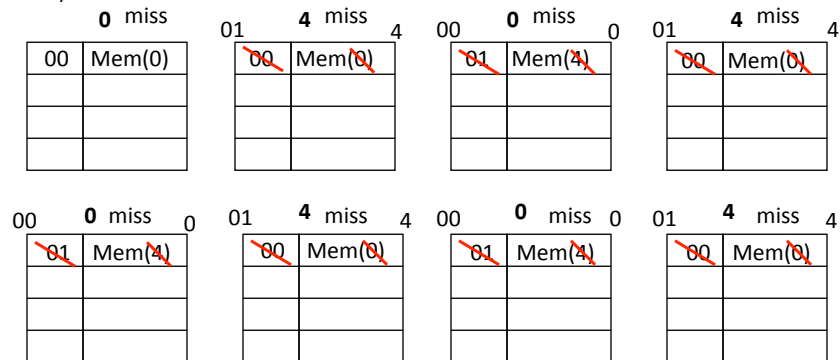
- DM placement: mem block 12 in 8 block cache: only one cache block where mem block 12 can be found— $(12 \text{ modulo } 8) = 4$
- SA placement: four sets x 2-ways (8 cache blocks), memory block 12 in set $(12 \text{ mod } 4) = 0$; either element of the set
- FA placement: mem block 12 can appear in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

Fall 2011 -- Lecture #14

17

Example: 4-Word Direct-Mapped \$ Worst-Case Reference String

- Consider the sequence of memory accesses
 Start with an empty cache - all blocks initially marked as not valid
 0 4 0 4 0 4 0 4

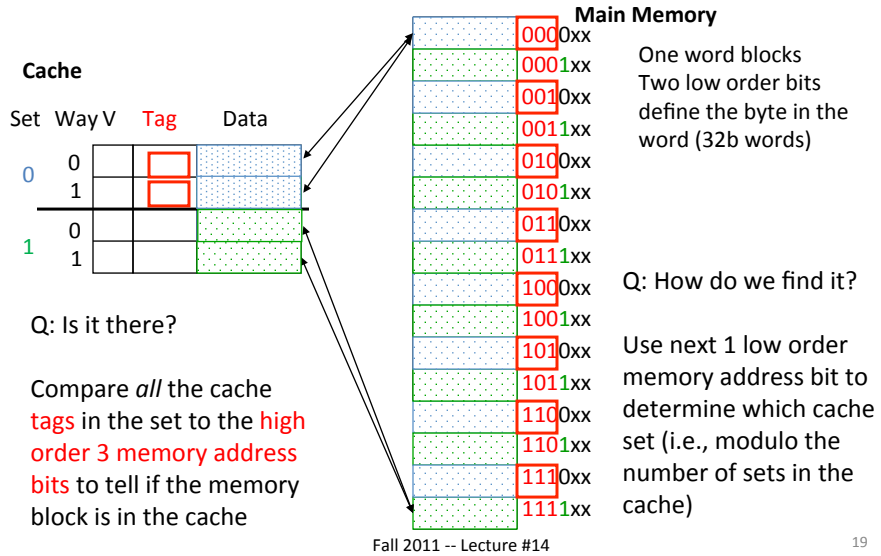


- 8 requests, 8 misses
- Ping pong effect due to conflict misses - two memory locations that map into the same cache block

Fall 2011 -- Lecture #14

18

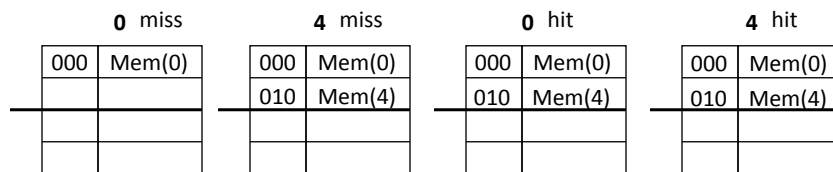
Example: 2-Way Set Associative \$ (4 words = 2 sets x 2 ways per set)



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

- Consider the sequence of memory accesses

Start with an empty cache - all blocks initially marked as not valid 0 4 0 4 0 4 0 4



- 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

One-way set associative (direct mapped)

Block	Tag	Data
0		
1		
2		
3		
4		
5		
6		
7		

Two-way set associative

Set	Tag	Data	Tag	Data
0				
1				
2				
3				

Four-way set associative

Set	Tag	Data	Tag	Data	Tag	Data	Tag	Data
0								
1								

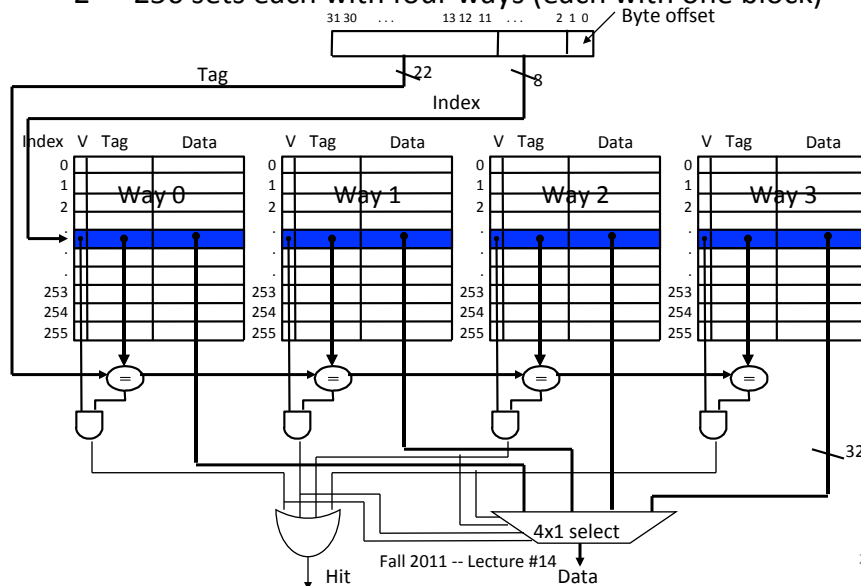
Eight-way set associative (fully associative)

Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data

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 \$.

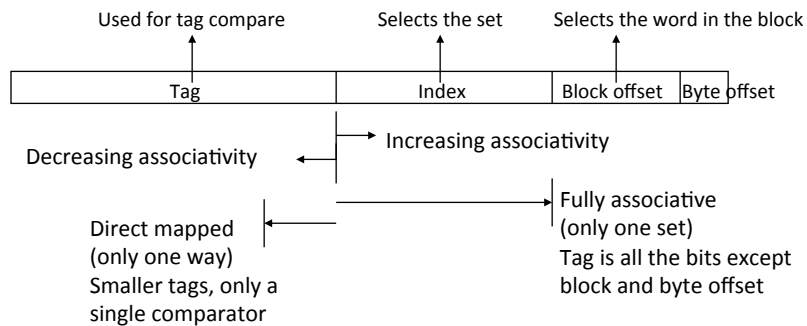
Four-Way Set-Associative Cache

- $2^8 = 256$ 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 of 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



Fall 2011 -- Lecture #14

23

Costs of Set-Associative Caches

- N-way set-associative cache costs
 - N comparators for tag comparisons
 - Must choose appropriate set (multiplexer) before data is available
- When miss occurs, which way's block selected for replacement?
 - Random Replacement: Hardware randomly selects a cache item and throw it out
 - Least Recently Used (LRU)**: one that has been unused the longest

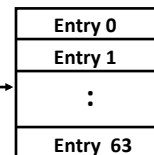
Fall 2011 -- Lecture #14

24

LRU Cache Block Replacement

- 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
 - On access set access bit for used block, clear other one
- Example of a Simple “Pseudo” LRU Implementation
 - Assume 64 Fully Associative entries in a set.
 - 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

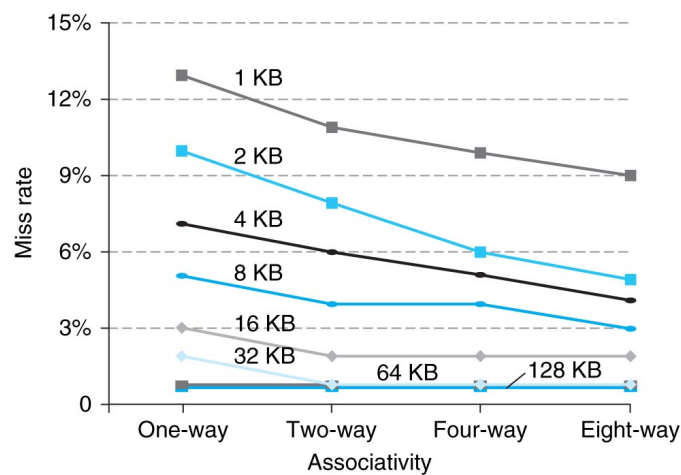
Replacement
Pointer



Fall 2011 -- Lecture #14

25

Benefits of Set-Associative Caches



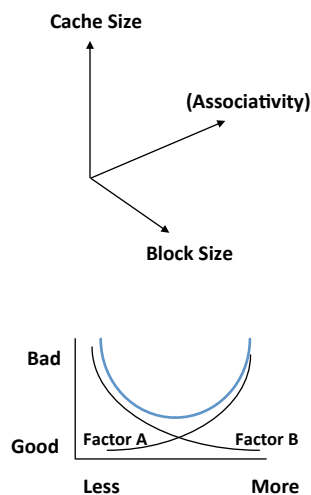
- Largest gains are in going from direct mapped to 2-way (20%+ reduction in miss rate)

Fall 2011 -- Lecture #14

26

The Cache Design Space

- Several interacting dimensions
 - Cache size
 - Block size
 - Write-through vs. write-back
 - Write allocation
 - Associativity
 - Replacement policy
- Optimal choice is a compromise
 - Depends on access characteristics
 - Workload
 - Use (I-cache, D-cache)
 - Depends on technology / cost
- Simplicity often wins



9/28/11

Fall 2011 -- Lecture #14

27

Characteristic	Intel Nehalem	AMD Opteron X4 (Barcelona)
L1 cache organization	Split instruction and data caches	Split instruction and data caches
L1 cache size	32 KB each for instructions/data per core	64 KB each for instructions/data per core
L1 cache associativity	4-way (I), 8-way (D) set associative	2-way set associative
L1 replacement	Approximated LRU replacement	LRU replacement
L1 block size	64 bytes	64 bytes
L1 write policy	Write-back, Write-allocate	Write-back, Write-allocate
L1 hit time (load-use)	Not Available	3 clock cycles
L2 cache organization	Unified (instruction and data) per core	Unified (instruction and data) per core
L2 cache size	256 KB (0.25 MB)	512 KB (0.5 MB)
L2 cache associativity	8-way set associative	16-way set associative
L2 replacement	Approximated LRU replacement	Approximated LRU replacement
L2 block size	64 bytes	64 bytes
L2 write policy	Write-back, Write-allocate	Write-back, Write-allocate
L2 hit time	Not Available	9 clock cycles
L3 cache organization	Unified (instruction and data)	Unified (instruction and data)
L3 cache size	8192 KB (8 MB), shared	2048 KB (2 MB), shared
L3 cache associativity	16-way set associative	32-way set associative
L3 replacement	Not Available	Evict block shared by fewest cores
L3 block size	64 bytes	64 bytes
L3 write policy	Write-back, Write-allocate	Write-back, Write-allocate
L3 hit time	Not Available	38 (?)clock cycles

9/28/11

Fall 2011 -- Lecture #14

28

Summary

- AMAT to measure cache performance
- Cache can have major impact on CPI
- Multi-level caches - Reduce Cache Miss Penalty
 - Optimize first level to be fast!
 - Optimize 2nd and 3rd levels to minimize the memory access penalty
- Set-associativity - Reduce Cache Miss Rate
 - Memory block maps into more than 1 cache block
 - N-way: n possible places in cache to hold a memory block
- Lots and lots of cache parameters!
 - Write-back vs. write through, write allocation, block size, cache size, associativity, etc.