Computer Architecture: Cache Optimizations

Berk Sunar and Thomas Eisenbarth

ECE 505

WPI

Memory Hierarchy Design

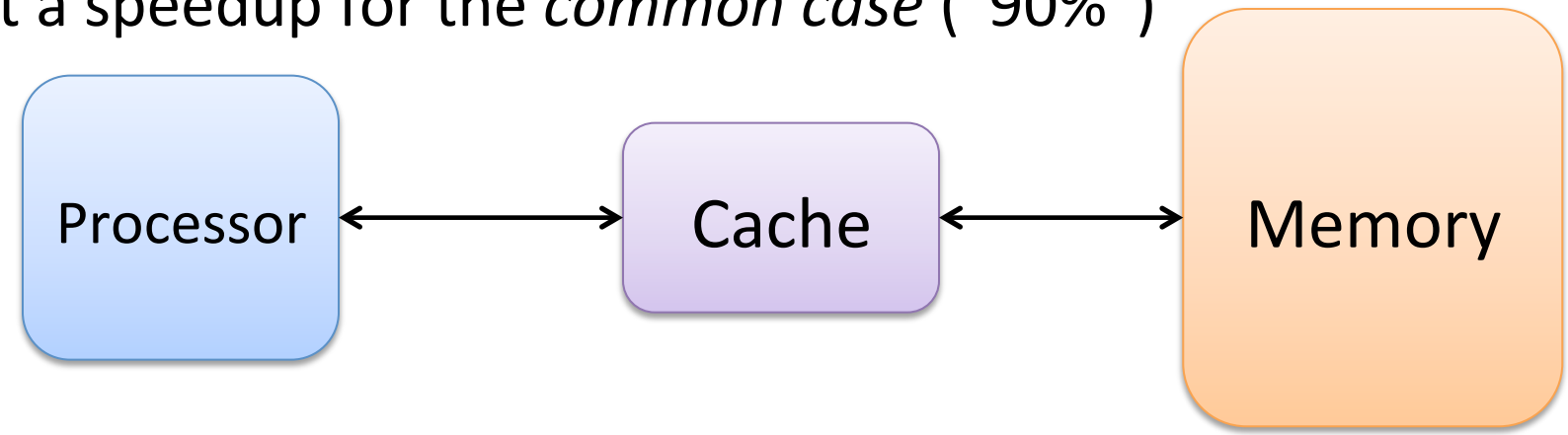
- Basic Principles of Data Cache Chap. 2
- Six basic optimizations for caches Appendix B
- Performance of Caches
- Ten advanced optimizations for caches
- Virtual Memory and Virtual Machines

Last time: Why Caches?

10/90 rule (of thumb)**:**

- Instruction Memory: 10% of static instns make up 90% of executed instns (Inner loops)
- Data Memory:10% of variables account for 90% of accesses (Frequently used globals, inner loop stack variables)

Cache Idea: Use small but fast memory holding the 10% to get a speedup for the *common case* ("90%")



The three C's (4 Cs)

- Compulsory (cold): first ever access to a block
  - would miss even in an inifinite-sized cache
- Capacity: miss because cache is not big enough
  - would miss even in fully associative cache
- Conflict: miss because of low associativity
  - remaining misses
- *(Coherence*: misses due to external invalidations)
  - only if other components can access memory (e.g., multiprocessors, systems with I/O)

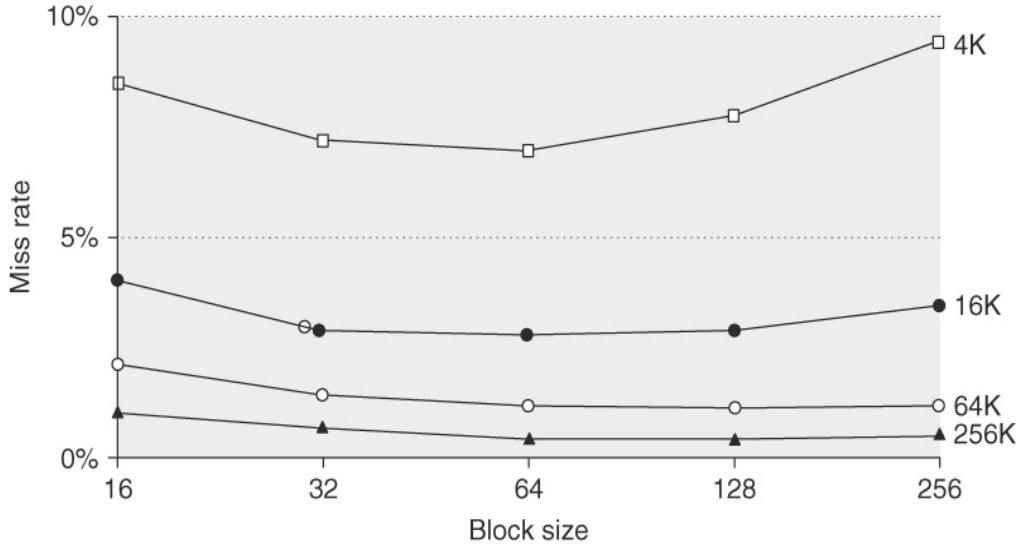
How to improve Cache Performance?

1. Larger Block Size
2. Bigger Caches
3. Higher Associativity
4. Multilevel caches
5. Prioritizing read misses over write misses
6. Avoiding address translation during indexing to reduce hit time

Larger Block Size

Use spatial locality:

- reduce compulsory misses
- but increases miss penalty and
- may increase capacity and conflict misses in smaller caches.



Larger Block Size: Example

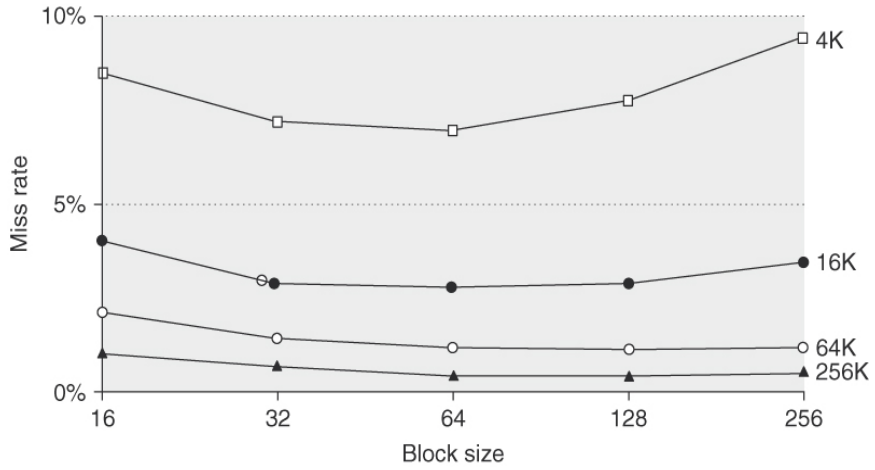
- Assume following Memory interface
  - Takes 80 cycles overhead, then
  - Delivers 16 bytes every 2 cycles
  - E.g. 16 bytes every 82 cycles, 32 bytes every 84 cycles etc.

Q:Which block size has smallest average cache access time?
Assume hit time is 1 cycle

Hint: use: Average Memory Access Time
= Hit Time + (Miss Rate * Miss Penalty)

Miss rates:

Block Size	4k cache	16k cache	64k cache
16	8.57%	3.94%	2.04%
32	7.24%	2.87%	1.35%
64	7.00%	2.64%	1.06%
128	7.78%	2.77%	1.02%



Larger Block Size: Solution

- Assume following Memory interface
  - Takes 80 cycles overhead, then
  - Delivers 16 bytes every 2 cycles
  - E.g. 16 bytes every 82 cycles, 32 bytes every 84 cycles etc.

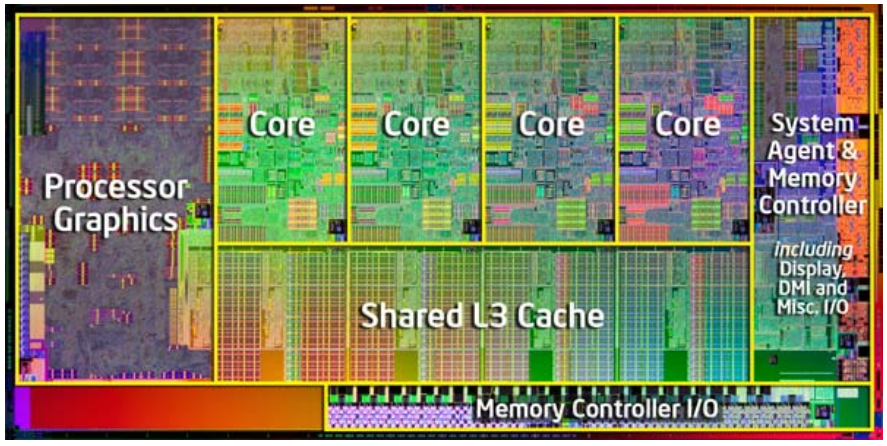
Q:Which block size has smallest average cache access time?

Hint: use: Average Memory Access Time
= Hit Time + (Miss Rate * Miss Penalty)

Block Size	Miss Penalty	4k cache	16k cache	64k cache
16	82	8.027	4.231	2.673
32	84	7.082	3.411	2.134
64	88	7.160	3.323	1.922
128	96	8.469	3.659	1.979

Larger Caches to Reduce Miss Rate

- (obvious one)
- Might result in higher hit time
- Higher cost and higher power consumption (caches are large in Hardware)



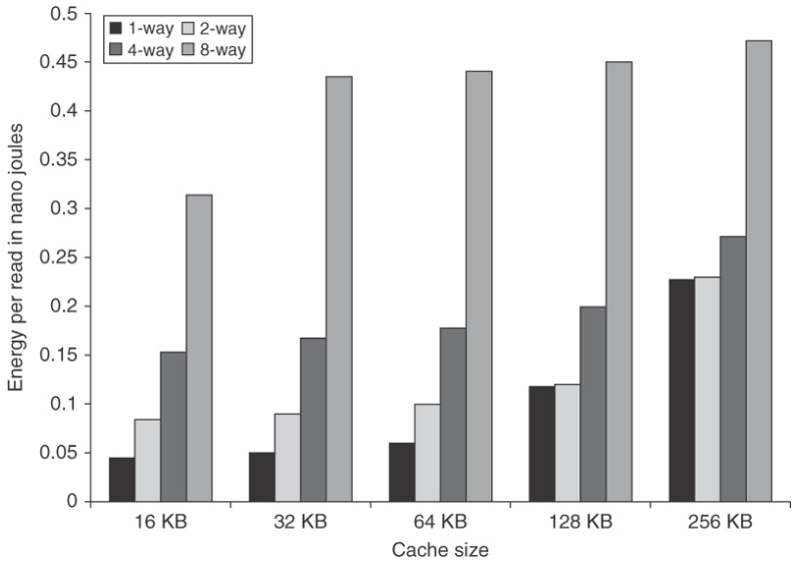
Intel i5-3320

Recall Example: Miss rates:

Block Size	4k cache	16k cache	64k cache
16	8.57%	3.94%	2.04%
32	7.24%	2.87%	1.35%
64	7.00%	2.64%	1.06%
128	7.78%	2.77%	1.02%

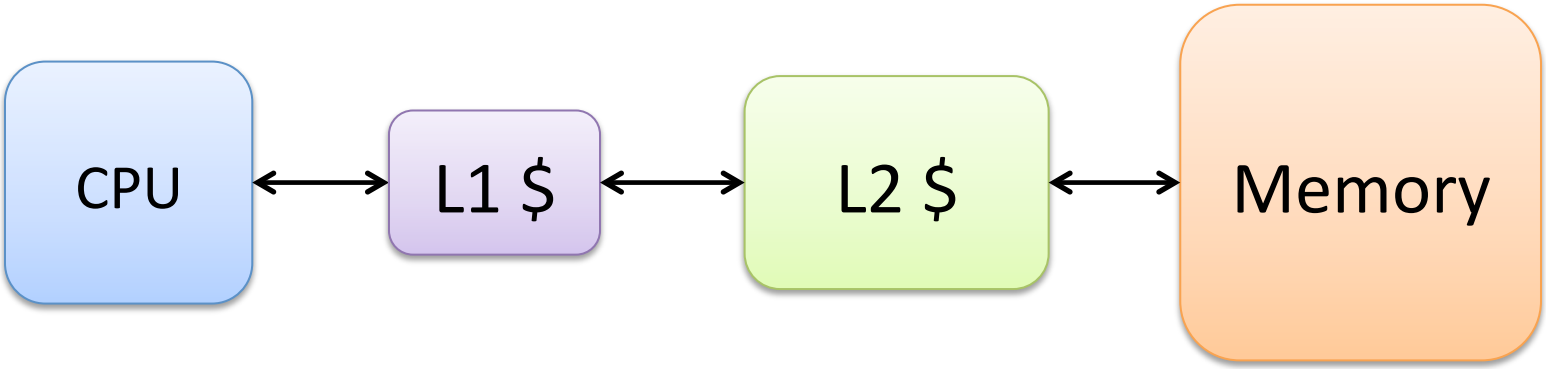
Higher Associativity to Reduce Cache Misses

+ Higher Associativity reduces conflict misses
  - **Recall 2:1 rule of thumb:** Direct Mapped cache of size N has same miss rate as 2-way set associative cache of size N/2
- May increase hit time
- Increases power consumption (later)



Energy increases with cache size and with increased associativity

Multilevel Caches to reduce Miss Penalty



- Forced by tradeoff between cache size (reducing miss rate) vs. hit time (increases with larger cache)
  - L1 cache close to CPU speed
  - L2 cache large enough to reduce miss rates (and still much faster than memory access)
  - Lx cache can have larger blocks, bigger capacity and higher associativity, while saving power (why?)

Multilevel Caches to reduce Miss Penalty

- Local miss rate: misses in cache / accesses to cache
  - Not a good metric for L2 cache: L1 cache gets most of hits
- Global miss rate: misses in cache / CPU memory accesses
- *New* Average Memory Access Time

  $= \text{Hit Time}_{L1} + \text{Miss Rate}_{L1} \times (\text{Hit Time}_{L2} + \text{Miss Rate}_{L2} \times \text{Miss Penalty}_{L2})$

  - Since $\text{Miss Penalty}_{L1} = \text{Hit Time}_{L2} + \text{Miss Rate}_{L2} \times \text{Miss Penalty}_{L2}$

Q: Of 1000 memory references, 50 misses in L1 and 20 misses in L2, Miss Penalty for L2 is 100, Hit time for L2 is 11 cycles, for L1 is 1

- compute all miss rates for L1 and L1 cache
- What is the average Memory access time?

Example 1:

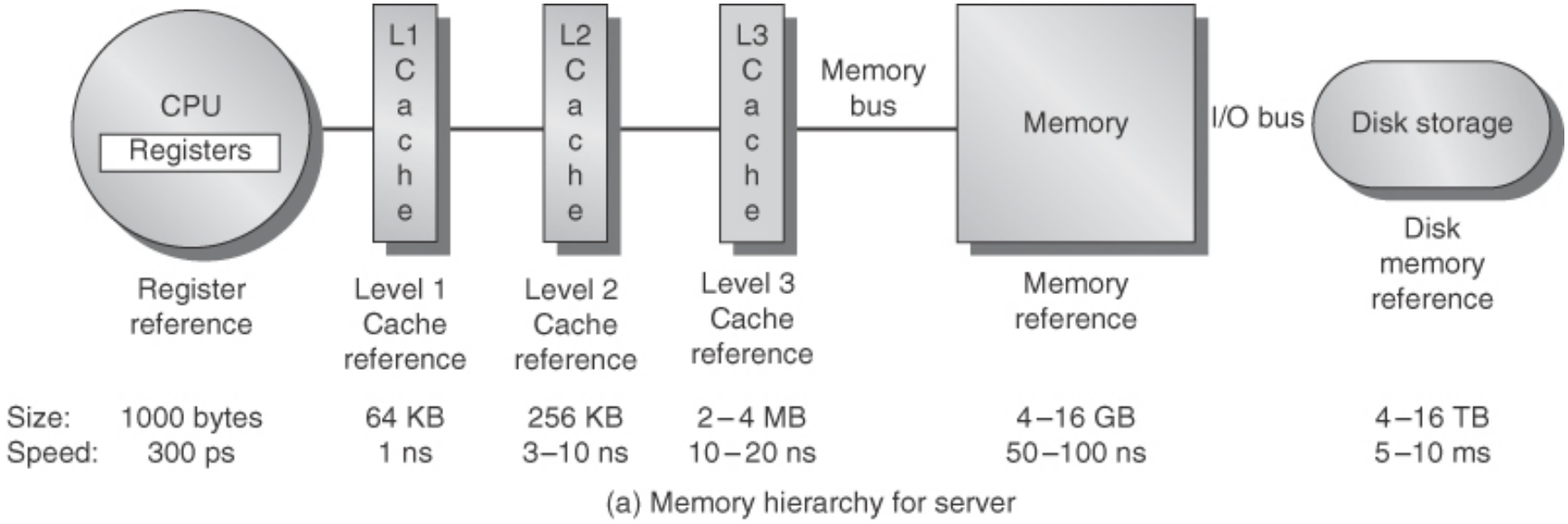
Of 1000 memory references, 50 misses in L1 and 20 misses in L2, Miss Penalty for L2 is 100, Hit "penalty" for L2 is 10 cycles, for L1 the hit time is 1:

- L1 Local Miss Rate = L1 Global Miss Rate = 50/1000 = 5%
- L2 Local Miss Rate = 20/50 = 40%
- L2 Global Miss Rate = 20/1000 = 2%

- Average Memory access time

  = 1 + 5% x (10 + 40% x 100) = 1 + .05 x 50 = 3.5 cycles

Recall:Typical Memory Hierarchy



(a) Memory hierarchy for server

- Server type CPUs commonly have 3 levels of cache
- Newer CPUs now feature L4 faster memory (about 128MB)
- Future CPUs will have memory in same package

Prioritizing Read Misses Over Write Misses

- Make common case fast
- Typical implementation: write buffer (common)
  - Serves read access before writes have been completed
  - May introduce hazards, as updated value might be in write buffer (RAW hazard through memory)
  - Possible solution: check write buf on read miss
- Little effect on power consumption

Avoiding Address Translation During Indexing to Reduce Hit Time

- Caches must handle address translation between virtual (OS) and physical addresses
- Common Method: use page offset (part of address that is not randomized) to index cache
  - Removes TLB (Translation Lookaside Buffer) access from critical path of cache access
  - But limits size and structure of L1 cache
- Used by Intel iX, not used in some ARM processors

Six Basic Optimizations Wrap-Up

- Most optimizations come with tradeoffs
- Depending on application scenario, the choices will vary
  - Server vs. Desktop PC: Server cache usually bigger, as many users on one machine: higher memory bandwidth needed
  - PC vs. Embedded: Energy is big concern in Handhelds, and performance is usually less critical

Basic Cache Optimizations Overview

Technique	Hit time	Miss penalty	Miss Rate	Hardware complexity
Larger block size				
Larger cache size				
Higher associativity				
Multilevel caches				
Read Priority over writes				
Avoiding Address translation during cache indexing				

\+ Improvement

\- Gets worse

Number indicates complexity (0 to 3)

Basic Cache Optimizations Overview

Technique	Hit time	Miss penalty	Miss Rate	Hardware complexity
Larger block size		–	+	0
Larger cache size	–		+	1
Higher associativity	–		+	1
Multilevel caches		+		2
Read Priority over writes		+		1
Avoiding Address translation during cache indexing	+			1

\+ Improvement

\- Gets worse

Number indicates complexity (0 to 3)

Memory Hierarchy Design

- Basic Principles of Data Cache Chap. 2
- Six basic optimizations for caches Appendix B

- Ten advanced optimizations for caches

- Virtual Memory and Virtual Machines

Optimizations targeted:

From: Average Memory Access Time

= Hit Time + (Miss Rate * Miss Penalty)

- Reducing *hit time:*
- Reducing *miss penalty:*
- Reducing *miss rate:*

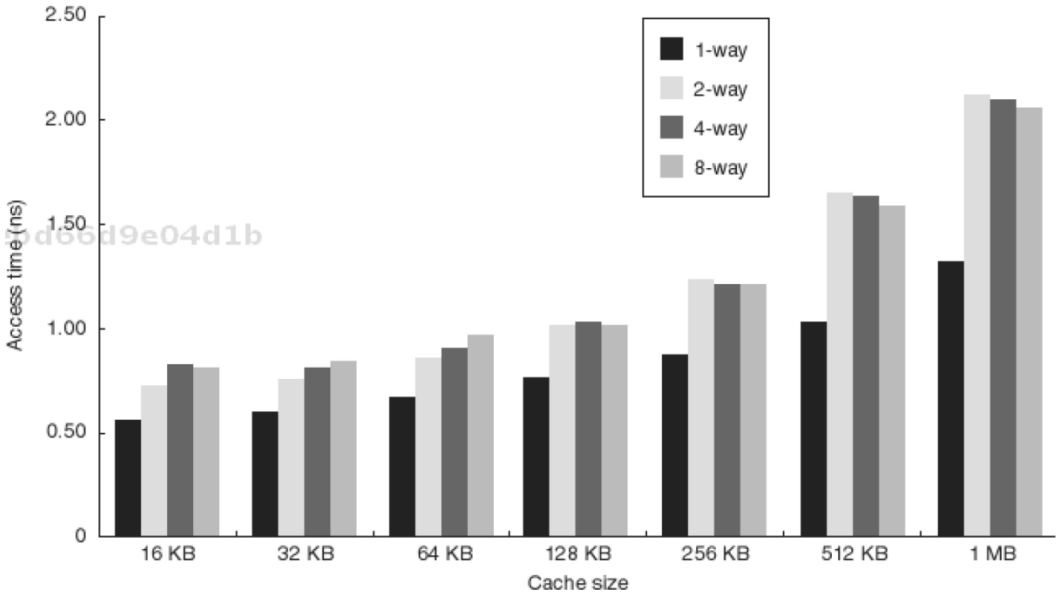
Additional optimization goals:

- Increasing Cache Bandwidth
- Decreasing Power Consumption

Way prediction to reduce hit time

- **Idea:** predict *way* within a cache set that is most likely to be accessed next→ reduce fetch latency for that way
- (block within set is *way*)
- **Benefit:** Combines high speed of direct mapped cache with the reduced conflict misses of set-associative caches

Access times vs associativity and cache size



Way prediction to reduce hit time

- Prediction accuracy 90% for 2-way SA$ (80% for 4w)
- Popular with (low) two-way set associative caches (used ARM Cortex A8 w/4-way S-A $)
- Reduces conflict misses while maintaining hit speed of direct-mapped cache
- Extreme version: *way selection* also accesses predicted block (especially for I-cache)
  - Saves power (>60%), but increases access time on miss.
  - Makes pipelining of cache accesses difficult

Pipelined Cache Access
to increase bandwidth

- Yields fast clock speed and high bandwidth, but slow hits.
  - Hit time: until Pentium 3: 1 cycle, P4: 2cc, I7: 4cc
  - Increased pipeline stages enable higher associativity (more ways), but higher read latency and higher miss penalty