Universidad
Carlos III de Madrid
www.uc3m.es

ARCOS

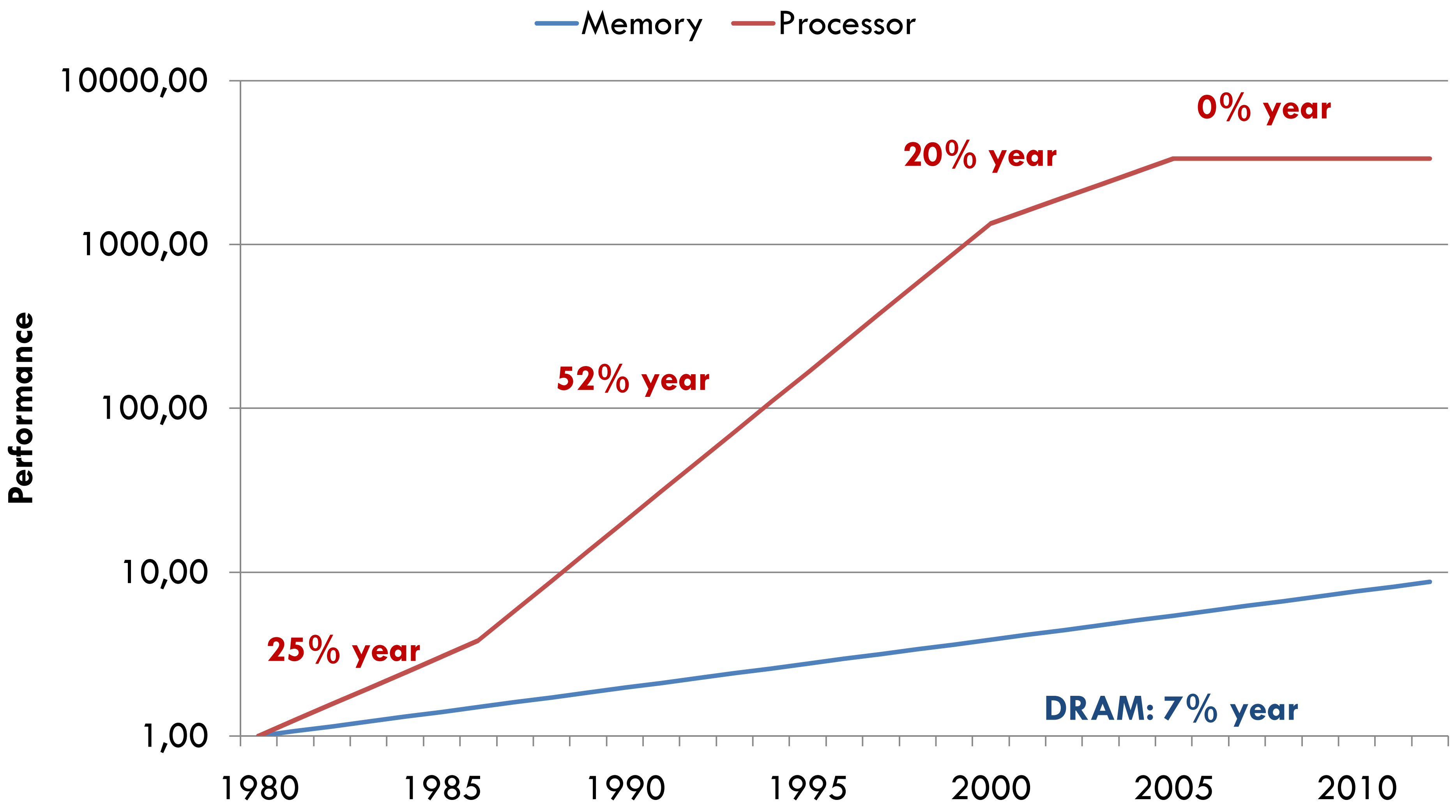
COMPUTER ARCHITECTURE

Cache Memory

Contents

- □ **Introduction.**
- □ Cache performance.
- □ Trade-offs in cache design.
- □ Basic cache optimizations.

Performance evolution (1/latency)



Multi-core effect

- **Intel Core i7**
  - 2 data access (64 bits) per cycle.
  - 4 cores, 3.2 GHz → 25.6 x 10^9 access/sec
  - Instruction demand: 12.8 x 10^9 of 128 bits.
  - Peak bandwidth: 409.6 GB/sec

- **SDRAM Memory**
  - DDR2 (2003): 3.2 GB/sec – 8.5 GB/sec
  - DDR3 (2007): 6.4 GB/sec – 17.06 GB/sec
  - DDR4 (2014?): 17.05 GB/sec – 25.6 GB/sec

- **Solutions:**
  - Multi-gate memory, pipelined caches, multi-level caches, per-core caches, instruction/data separation.

Locality principle

- **Locality principle:**
  - Program property exploited in hardware design.
  - Programs access to a relatively small portion of address space.
- **Types of locality:**
  - **Temporal locality**: Recently accessed elements tend to be accessed again.
    - Examples: loops, variable reuse, …
  - **Spatial locality**: Elements next to a recently accessed element ten to be accessed in near future.
    - Examples: sequential instruction execution, arrays, …

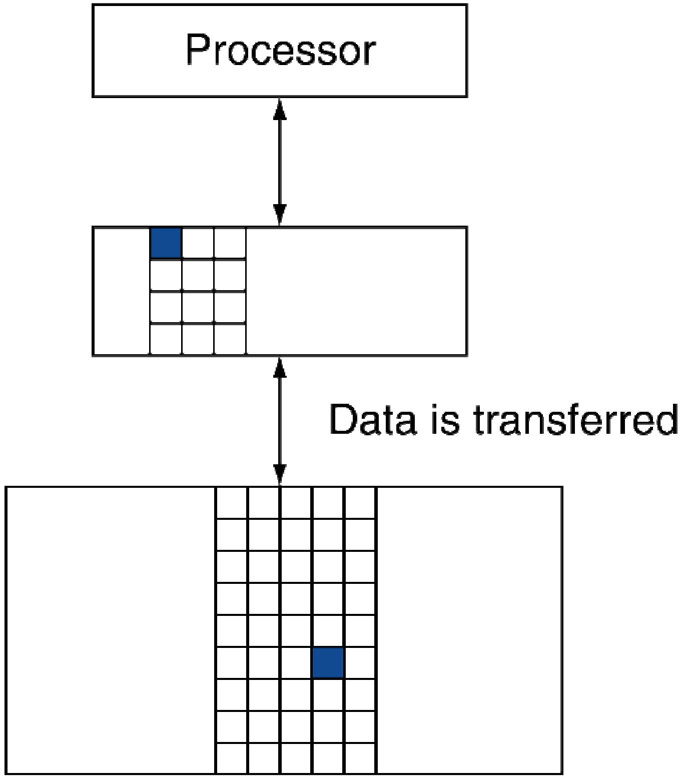
Situation (2008)

- SRAM – Static RAM
  - Access time: 0.5 ns – 2.5 ns
  - Cost per GB: 2000$ - 5000$

- DRAM – Dynamic RAM
  - Access time: 50ns – 70 ns
  - Cost per GB: 20$ - 75$

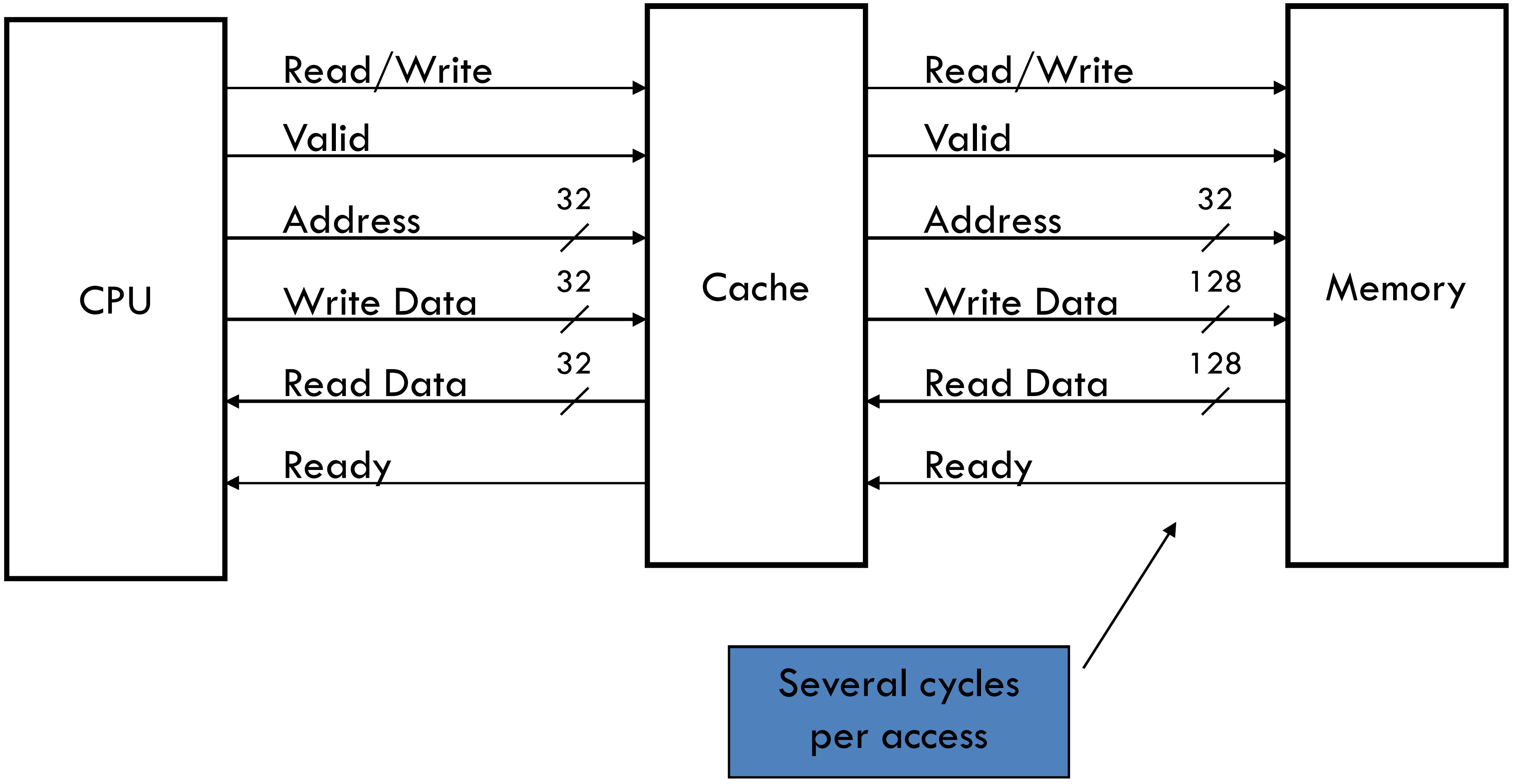
- Magnetic disk
  - Access time: 5,000,000 ns – 20,000,000 ns
  - Cost per GB: 0.20 $ - 2$

Memory hierarchy

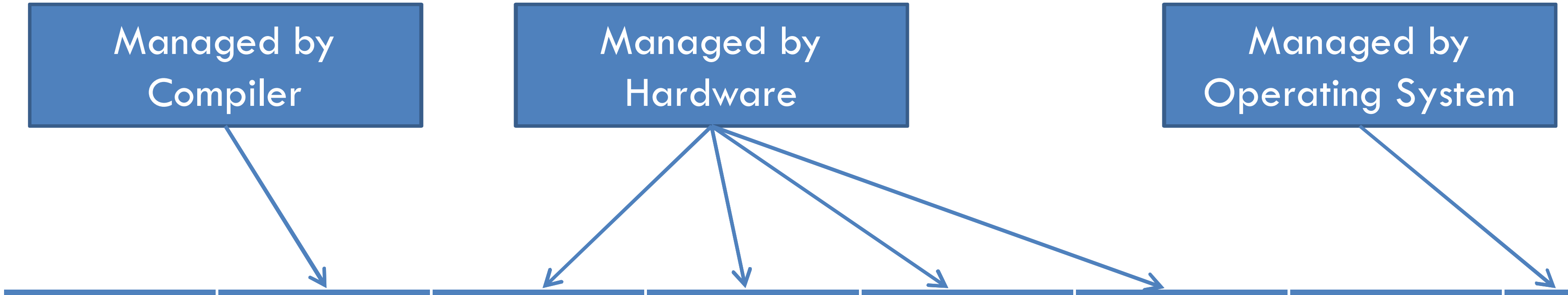


- **Block or line**: Unit of copy.
  - Usually several words.
- If accessed data present in higher level:
  - **Hit**: Delivered by higher level.
    - Hit rate = Hits / accesses.
- If accessed data is missing.
  - **Miss**: Block copied from lower level.
    - Needed time → Miss penalty.
    - Miss rate = Misses / accesses = 1 – Hit rate

Cache interconnection



Example: Haswell Core i7



	Registers	L1I	L1D	L2	L3	DRAM	Disk
Size	16	32KB	32KB	4x256KB	8MB	32GB	1.8TB
Latency	1 cycle	4-5 cycles	4-5 cycles	12 cycles	36 cycles	36 cycles + 57 ns	2.8 ms

Goal: Give the illusion of large, fast, and cheap memory

Allow programs to address space scalable to disk size at register file speed.

Contents

- □ Introduction.
- □ **Cache performance.**
- □ Trade-offs in cache design.
- □ Basic cache optimizations.

Measurements

- Average memory access time:

  $t_{avg} = t_H + (1-h) \cdot t_M$

- **Miss penalty**:
  - Time to replace a block and deliver to CPU.
  - Access time:
    - Time to get from lower level.
    - Dependent on lower level latency.
  - Transfer time:
    - Time to transfer a block.
    - Depending on bandwidth across levels.

Measurements

- **CPU execution time**
  - (CPU cycles + Memory stall cycles) x Cycle time
- **CPU cycles**
  - IC x CPI
- **Memory stall cycles**
  - Misses number x Miss penalty
  - IC x Misses per instruction x Miss penalty
  - IC x Instruction Memory access x Miss rate x Miss penalty

Beware: May be different for reads and writes.

- Where:
  - IC → Instruction Count.
  - CPI → Cycles per instruction.

Contents

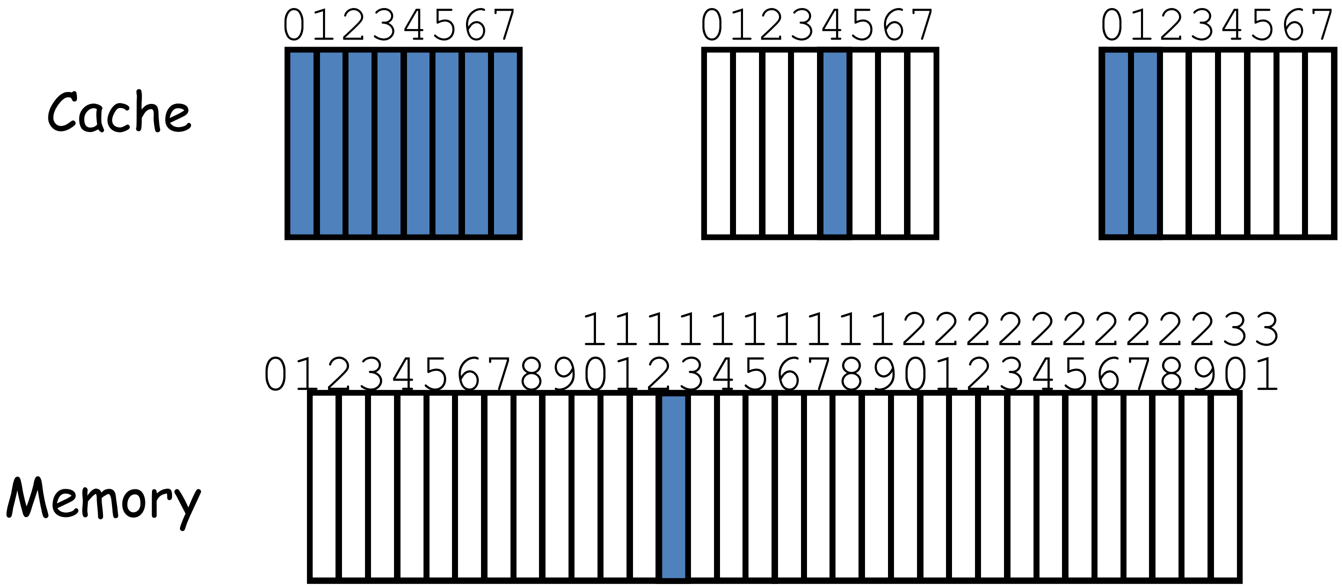
- □ Introduction.
- □ Cache performance.
- □ **Trade-offs in cache design.**
- □ Basic cache optimizations.

Four questions on memory hierarchy

- **Q1**: Where can a block be placed in the upper level?
  - Block placement.
- **Q2**: How is a block found in the upper level?
  - Block identification.
- **Q3**: Which block should be replaced on a miss?
  - Block replacement.
- **Q4**: What happens on a write?
  - Write strategy.

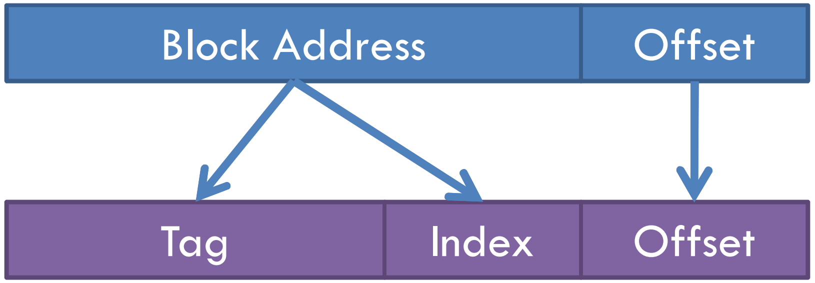
Q1: Block Placement

- **Direct mapping.**
  - Placement → block MOD blocks_in_cache
- **Fully associative mapping.**
  - Placement → Anywhere.
- **Set associative mapping.**
  - Set placement → block MOD number_of_sets.
  - Placement within set → Anywhere.



Q2: Block Identification

- **Block Address:**
  - **Tag**: Identifies the entry address.
    - Validity bit in every entry to flag if content is valid.
  - **Index**: Selects address.
- **Block offset:**
  - Select data within a block.
- **Higher associativity** means:
  - Less bits for Index.
  - More bits for Tag.



Q3: Block Replacement

- Relevant for associative and set associative mappings:
  - **Random.**
    - Easy to implement.
  - **LRU**: Less Recently Used.
    - Increasing complexity as associativity increases.
  - **FIFO**: First In First Out.
    - Approximates LRU with lower complexity.

Misses per1000 instr., SPEC 2000

	2 ways			4 ways			8 ways		
Size	LRU	Rand	FIFO	LRU	Rand	FIFO	LRU	Rand	FIFO
16 KB	114.1	117.3	115.5	111.7	115.1	113.3	109.0	111.8	110.4
64 KB	103.4	104.3	103.9	102.4	102.3	103.1	99.7	100.5	100.3
256 KB	92.2	92.1	92.5	92.1	92.1	92.5	92.1	92.1	92.5

From Hennessy & Patterson, 5Ed Appendix C

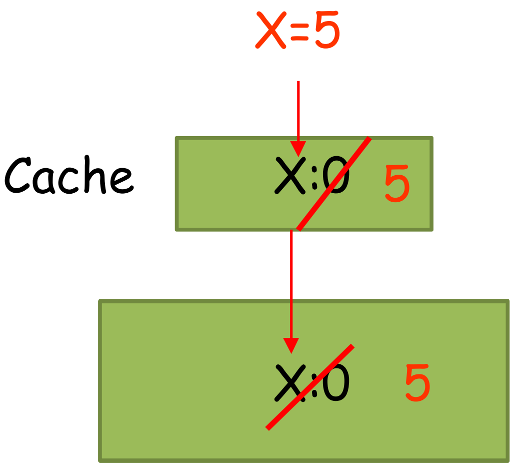
Q4: Write strategy

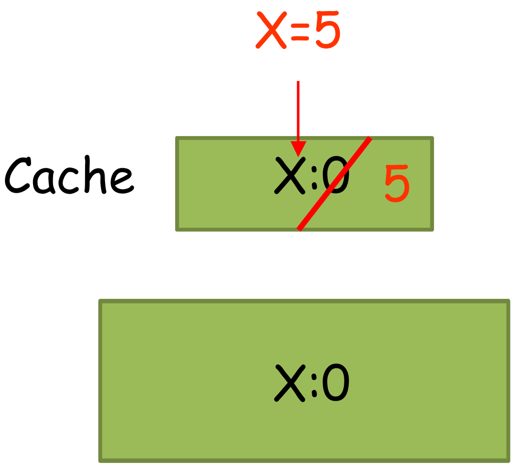
Write through

- □ All writes sent to bus and memory.
- □ Easy to implement.
- □ Performance issues in SMPs .

Write back

- □ Many writes are a hit.
- □ Write hits **not** sent to bus and memory.
- □ Propagation and serialization issues.
- □ More complex.

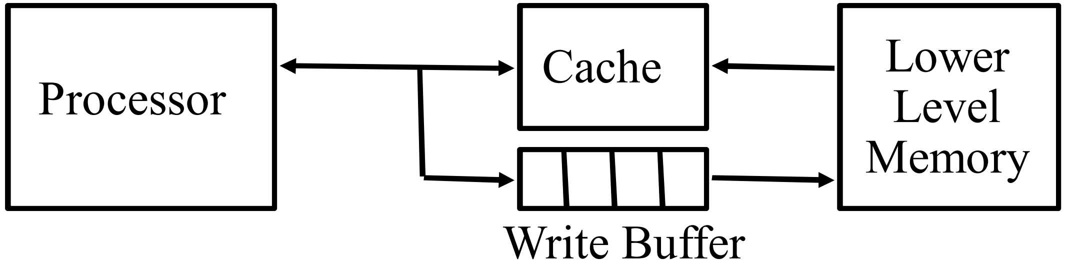




Q4: Write strategy

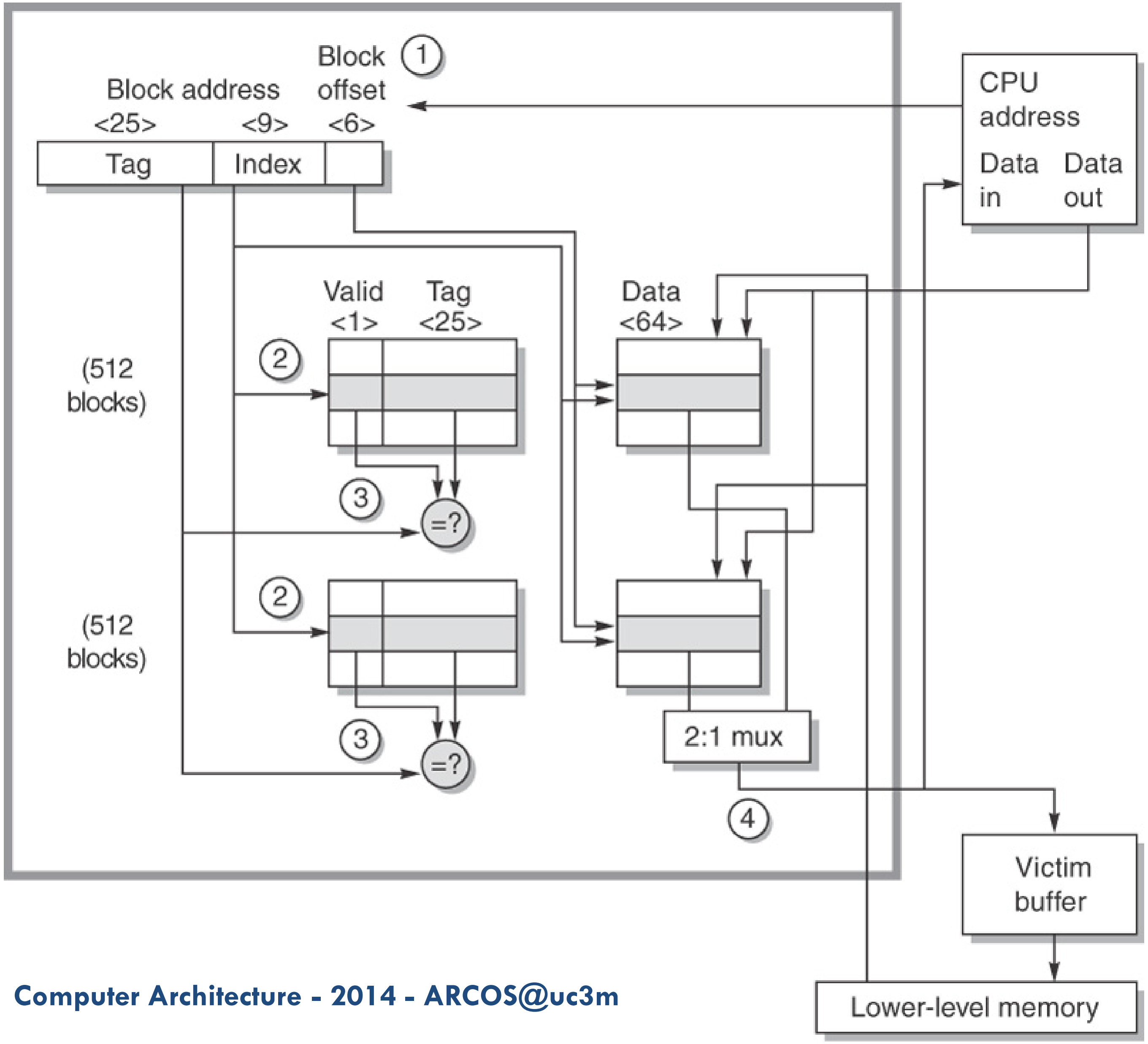
	Write-Through	Write-Back
Policy	Data written to cache blocks. Also written to next level in memory.	Data only written to cache. Update next levels when block evicted from cache
Debugging	Easy	Difficult
Write on miss?	No	Yes
Repeated writes sent to next level?	Yes	No

Write Buffers in caches with write through



- **Why a buffer?**
  - To avoid CPU stalls.
- **Why a buffer instead of a register?**
  - Write bursts are common.
- **Are RAW hazards possible?**
  - Yes.
  - Alternatives:
    - Flush buffer before a read.
    - Check buffer before a read.

Example: Opteron



Miss penalty and Out-of-Order execution

- **Miss penalty definition:**
  - Miss total latency.
  - Exposed latency (generating CPU stall).

- **Miss penalty:**
  - Memory stalls / Instructions
  - (Misses/Instructions) x (Total latency – overlapped latency)

Contents

- □ Introduction.
- □ Cache performance.
- □ Trade-offs in cache design.
- □ **Basic cache optimizations.**

Cache basic optimizations

- **Reduce miss rate.**
  - Larger block size.
  - Larger cache size.
  - Higher associativity.

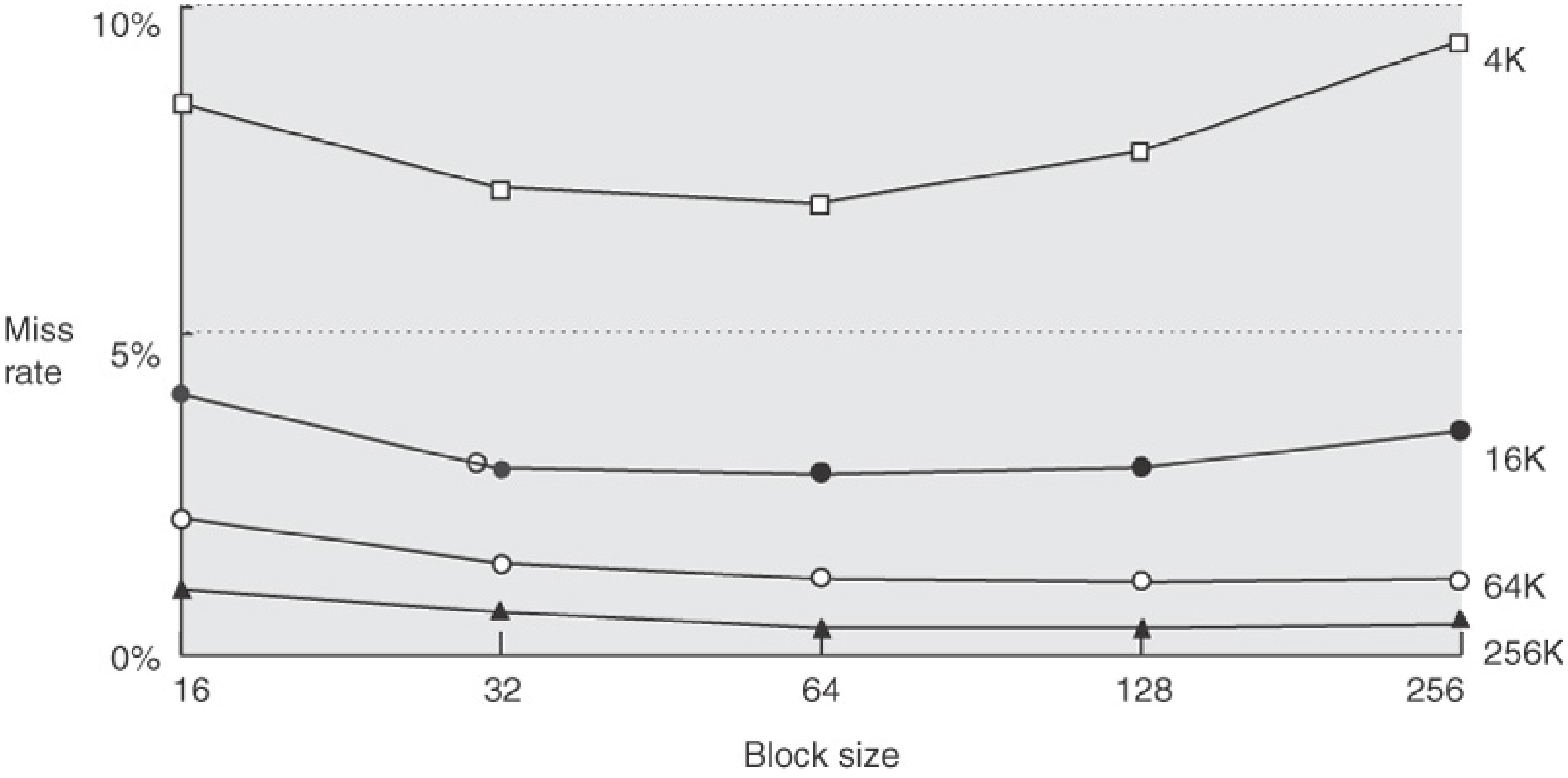
- **Reduce miss penalty.**
  - Multi-level caches.
  - Prioritize read over writes.

- **Reduce hit time.**
  - Avoid address translation in cache indexing.

1: Larger block size

- **Goal:** Reduce miss rate.
  - Improve spatial locality exploitation.
- **Increases miss penalty.**
  - Upon a miss, larger blocks need to be transferred.
  - More misses due to cache with less blocks.
- **Balance needed:**
  - High latency and high bandwidth memory:
    - Increase block size.
  - Low latency and low bandwidth memory:
    - Reduced block size.

Miss rate and block size

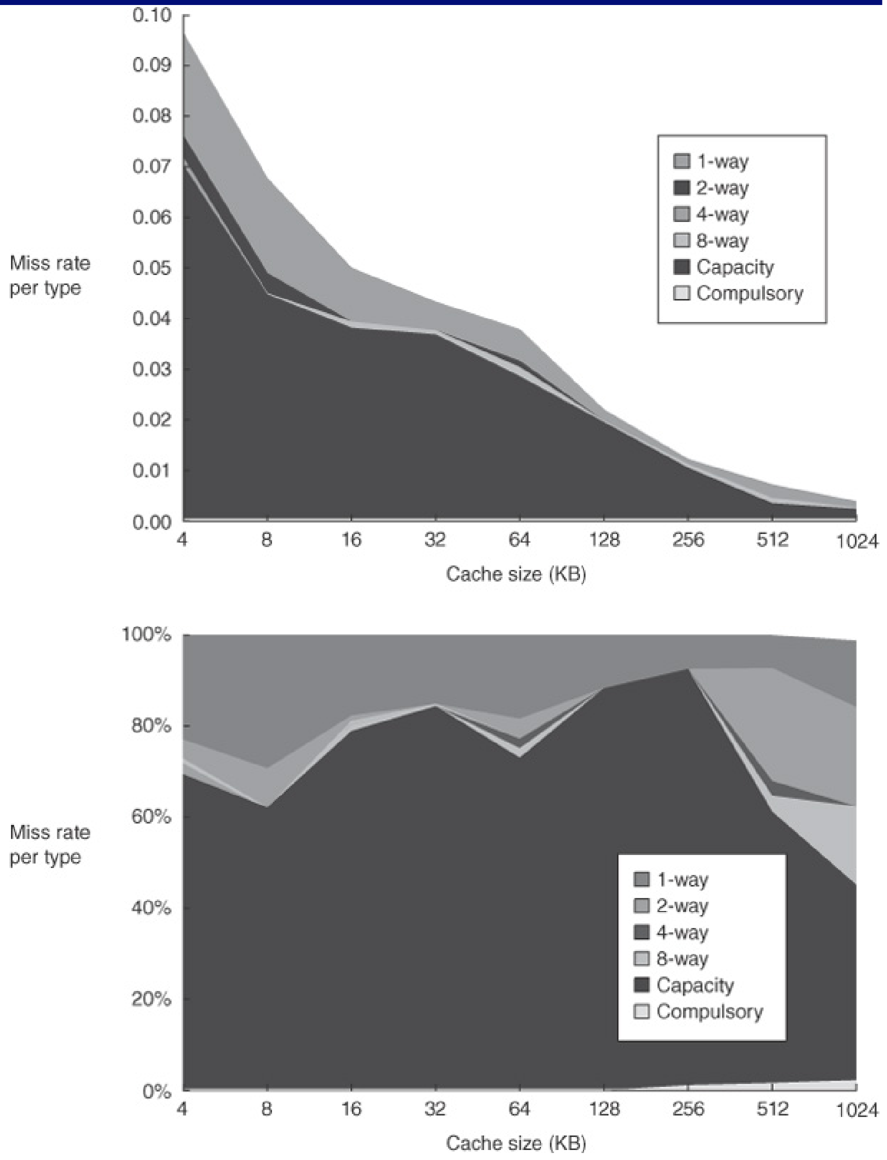


2: Larger cache size

- **Goal:** Reduce miss rate.
  - More data fit in cache.
- **May increase hit time.**
  - More time needed to find block.
- **Higher cost.**
- **Higher energy consumption.**
- Need to find balance:
  - Specially in on-chip caches.

3: Higher associativity

- □ **Goal:** Reduce miss rate.
  - Less conflicts as more ways in a set can be used.
- □ **May increase hit time.**
  - More time needed to find a block.
- □ Consequence:
  - 8 ways ≈ Fully associative



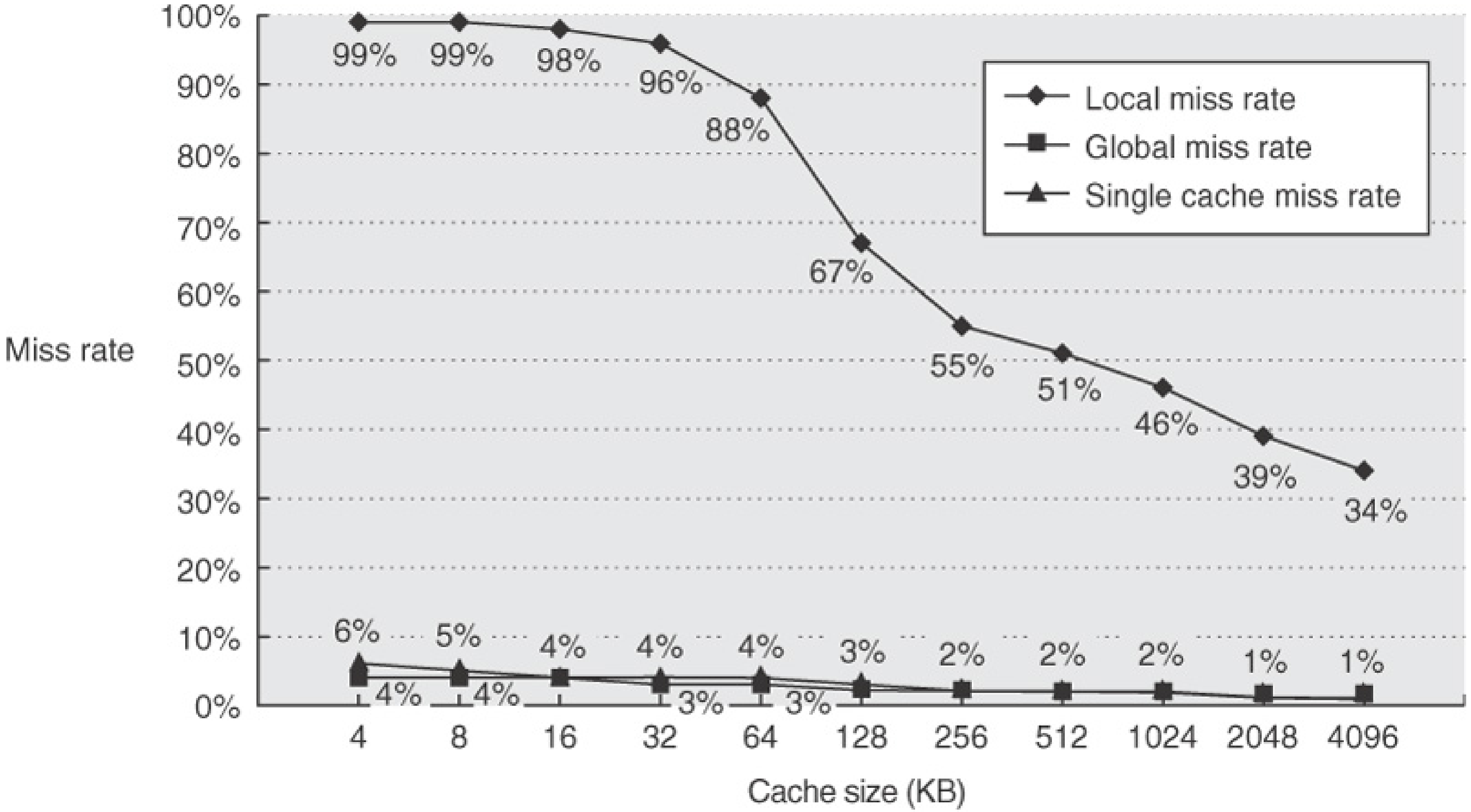
4: Multilevel caches

- **Goal:** Reduce miss penalty.
- **Evolution:**
  - Increasing performance gap between DRAM and CPU.
  - Miss penalty cost increased over time.
- **Alternatives:**
  - Faster caches.
  - Larger caches.
- **Solution:**
  - Both of them!
  - Several cache levels.

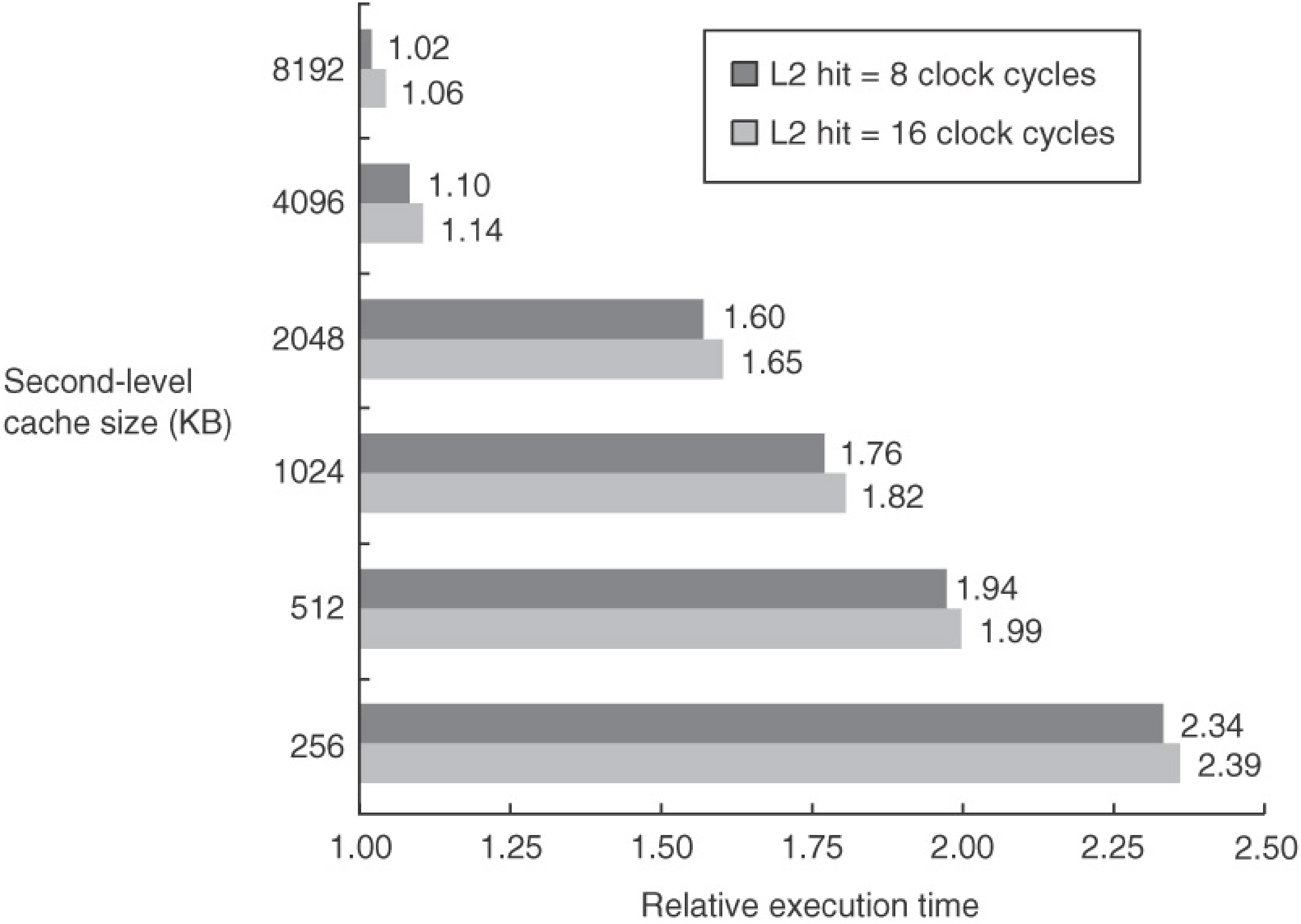
Local and global miss rate

- **Average access time:**
  - Hit time$_{L1}$ + Miss rate$_{L1}$ x Miss penalty$_{L1}$
  - Hit time$_{L1}$ + Miss rate$_{L1}$ x (Hit time$_{L2}$ + Miss rate$_{L2}$ x Miss penalty$_{L2}$)
- **Local miss rate:**
  - Misses at a cache level over accesses to that cache level.
  - L1: Miss rate$_{L1}$
  - L2: Miss rate$_{L2}$
- **Global miss rate:**
  - Misses at a cache level over **all** memory accesses.
  - L1: Miss rate$_{L1}$
  - L2: Miss rate$_{L1}$ x Miss rate$_{L2}$

Miss rate versus cache size for multilevel caches

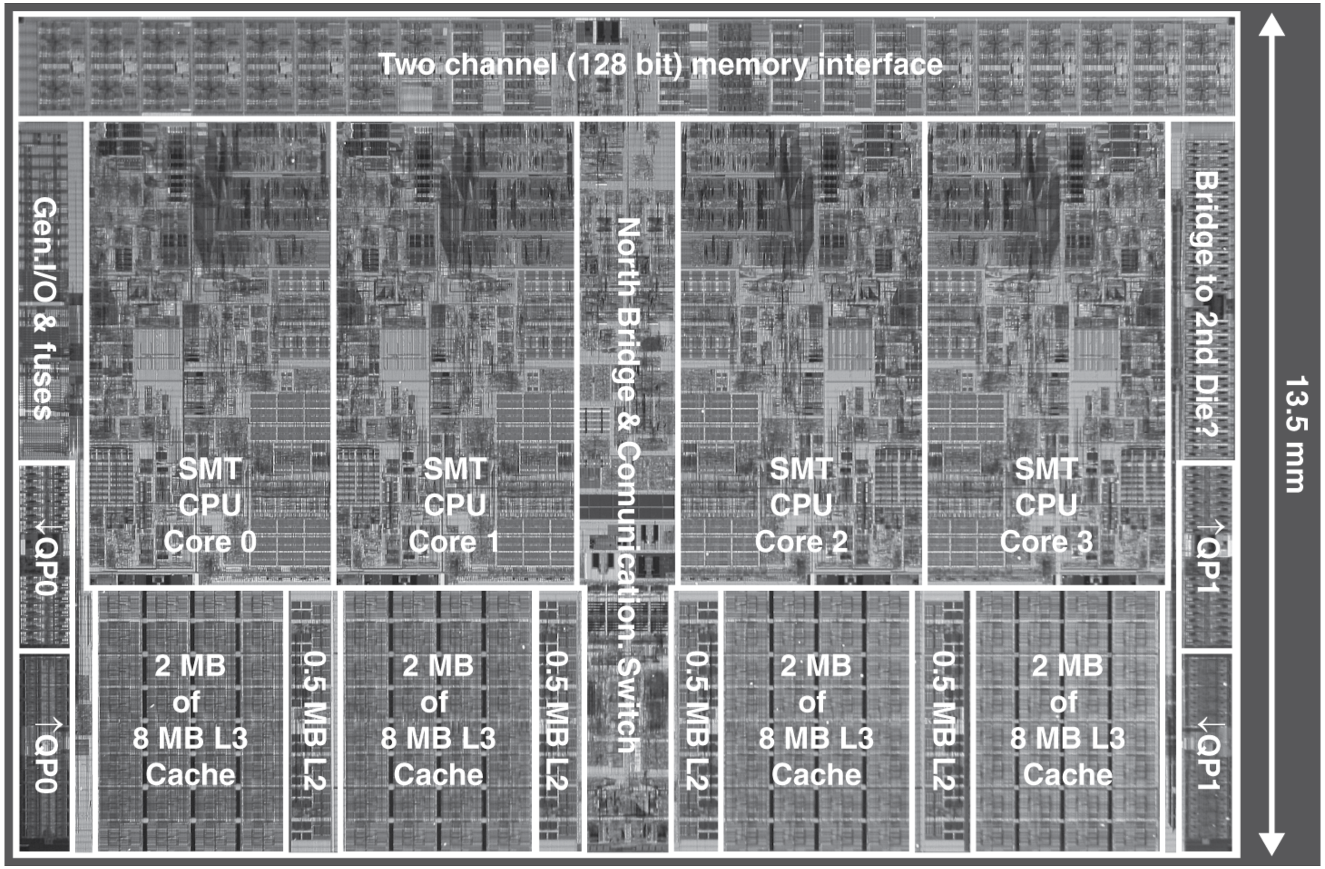


Relative execution time by second-level cache size



On-chip multilevel caches

Intel Nehalem 4-core processor



Per core: 32KB L1 I-cache, 32KB L1 D-cache, 512KB L2 cache

5: Prioritize read misses over writes

- **Goal:** Reduce miss penalties.
  - Avoid that a read miss has to wait until writes are completed.

- **Write-through caches:**
  - Write buffer could contain an updated value for the read address.
    - A) Wait until write buffer is empty.
    - B) Check contents of write buffer.
      - Continue with read miss if no conflict with buffer.

- **Write-back caches:**
  - A read miss could replace a modified block.
    - Copy modified block to buffer, read, and dump block to memory.
    - Apply options A or B to buffer.

6: Avoid Address Translation during Indexing

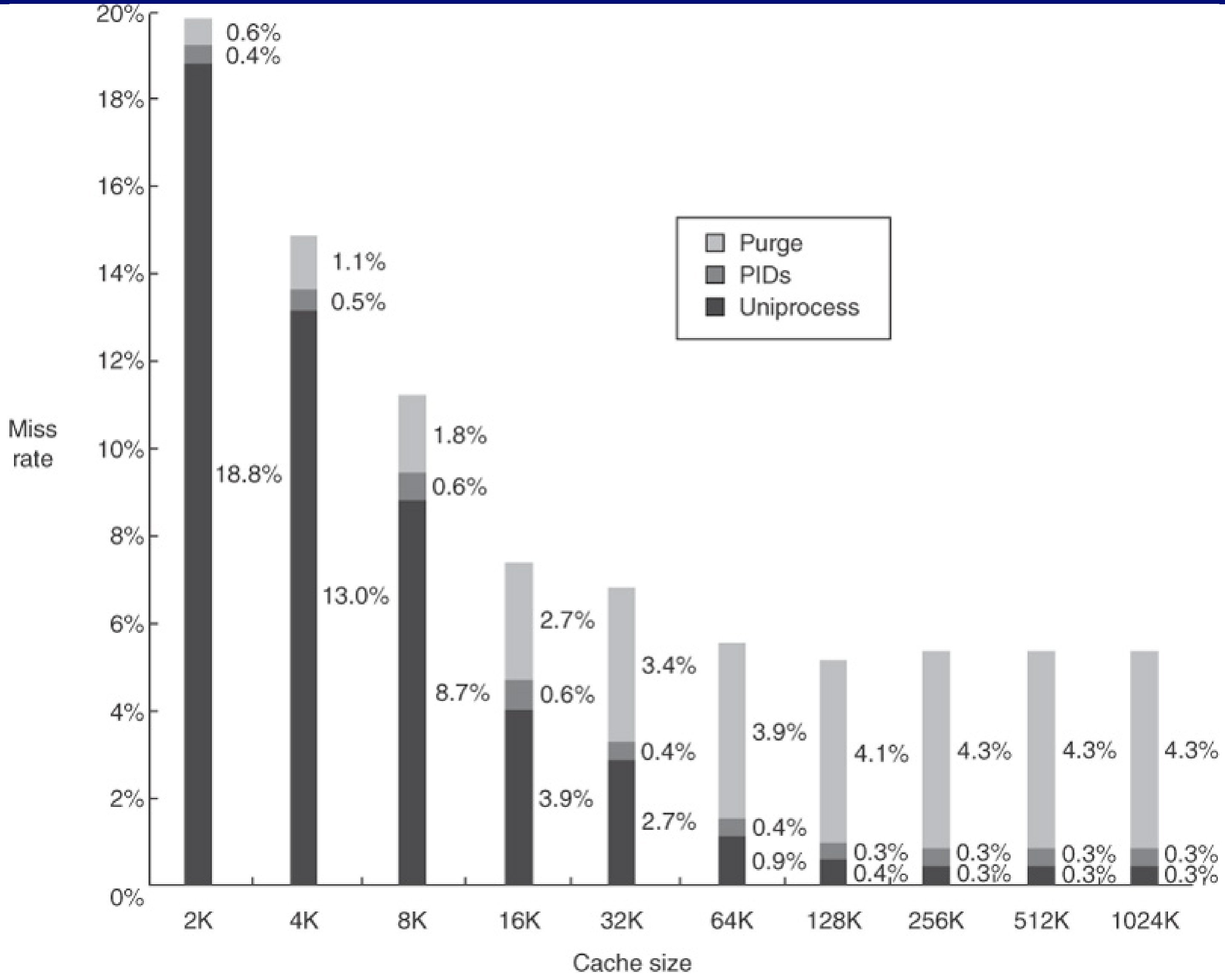
- **Goal:** Reduce hit time.
- **Translation process:**
  - Virtual address → Physical address.
  - May require additional accesses to memory.
    - Or at least to TLB.
- **Idea:** Optimize the most common case (hits).
  - Use virtual addresses for the cache.
- **Tasks:**
  - Indexing the cache.
  - Comparing tags.

Problems with virtual caches (I)

- **Protection:**
  - Page-level protection checked during virtual-to-physical translation.
  - Solution: Copy protection info from TLB o misses.

- **Process switch.**
  - Virtual addresses refer to different physical addresses.
    - Old process virtual addresses.
  - Solutions:
    - Flush the cache.
    - Add to cache address a PID tag.

Miss rate in virtual caches



Problems with virtual caches (II)

- **Aliasing:**
  - Two different virtual addresses for the same physical address.
  - Anti-aliasing hardware: to guarantee that every cache block corresponds to a unique physical address.
    - Check multiple addresses and invalidate.
  - Page coloring: Force all aliases to have identical their n last bits.
    - Makes impossible two alias to be at the same time in cache.
- **I/O addresses:**
  - I/O typically uses physical addresses.
  - Mapping to virtual addresses to interact with virtual caches.

Address translation

- **Solution:**
  - Virtual indexing and physically tagging.

- **Tasks:**
  - Indexing the cache → Use page offset.
    - This part is identical in physical and virtual addresses
  - Comparing tags → Use translated physical address.
    - Tag matching uses phyical address.

Summary

- Caching as a **solution** to mitigate the memory wall.
- Cache performance due to **locality principle** (spatial and temporal).
- **Miss penalty** dependent on access time and transfer time.
- Four key dimensions in **cache design**:
  - Block placement, block identification, block replacement and write strategy.
- Six **basic** cache **optimizations**:
  - **Reduce miss rate**: larger block size, larger cache size, higher associativity.
  - **Reduce miss penalty**: multilevel caches, prioritize reads over writes.
  - **Reduce hit time**: Avoid address translation when indexing.

References

- □ **Computer Architecture. A Quantitative Approach. Fifth Edition.**
  Hennessy y Patterson.
  Sections: B.1, B.2, B.3

- □ Exercises:
  B.1, B.2, B.3, B.4, B.5, B.6, B.7, B.8, B.9, B.10, B.11