



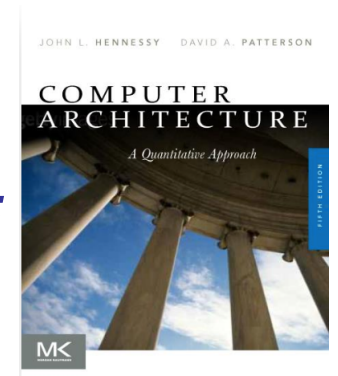
## Master Informatics Eng.

2020/21

*A.J.Proença*

### Memory Hierarchy

*(most slides are borrowed from this book:*

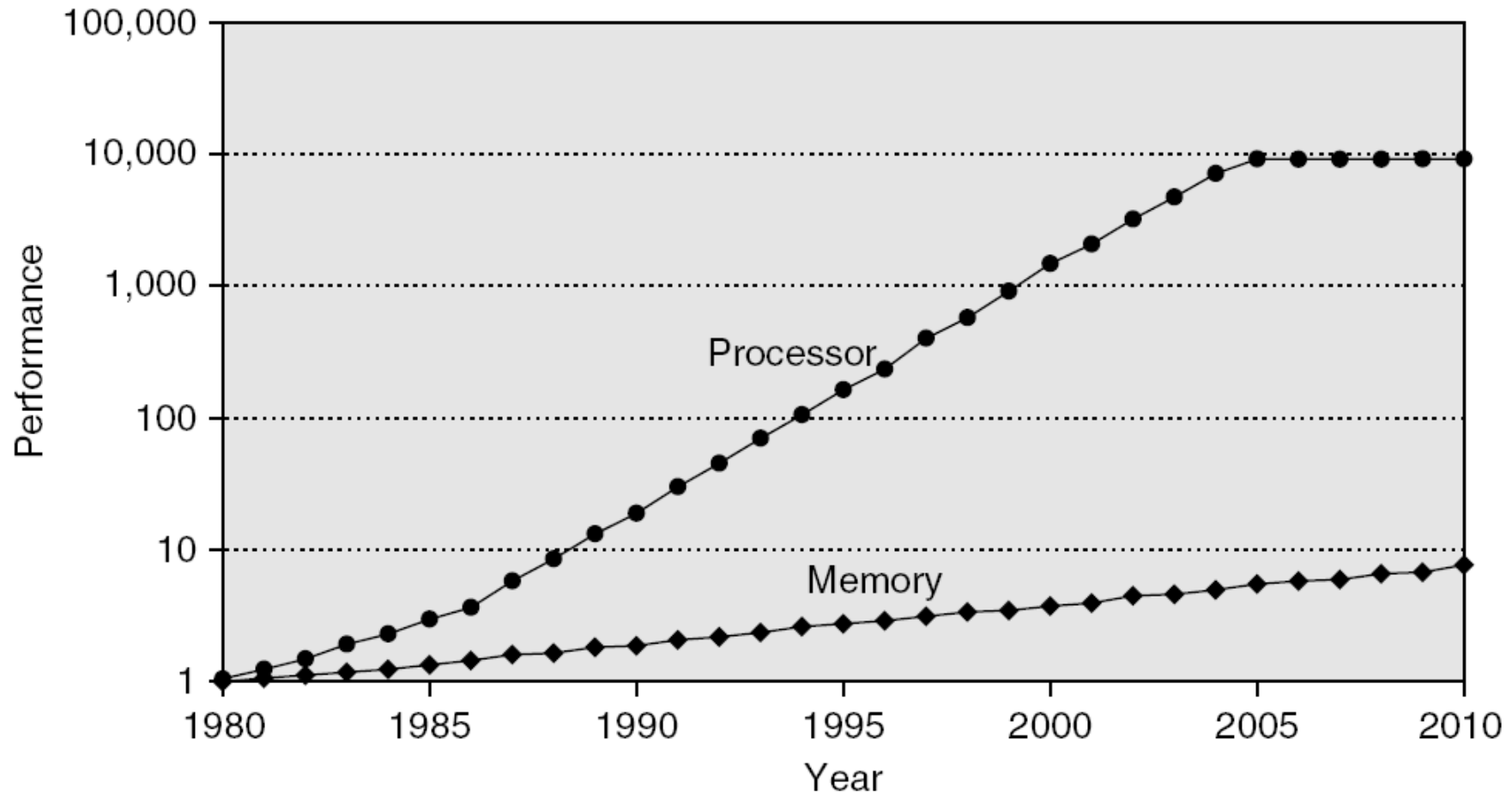


# Reading suggestions (from CAQA 5<sup>th</sup> Ed)

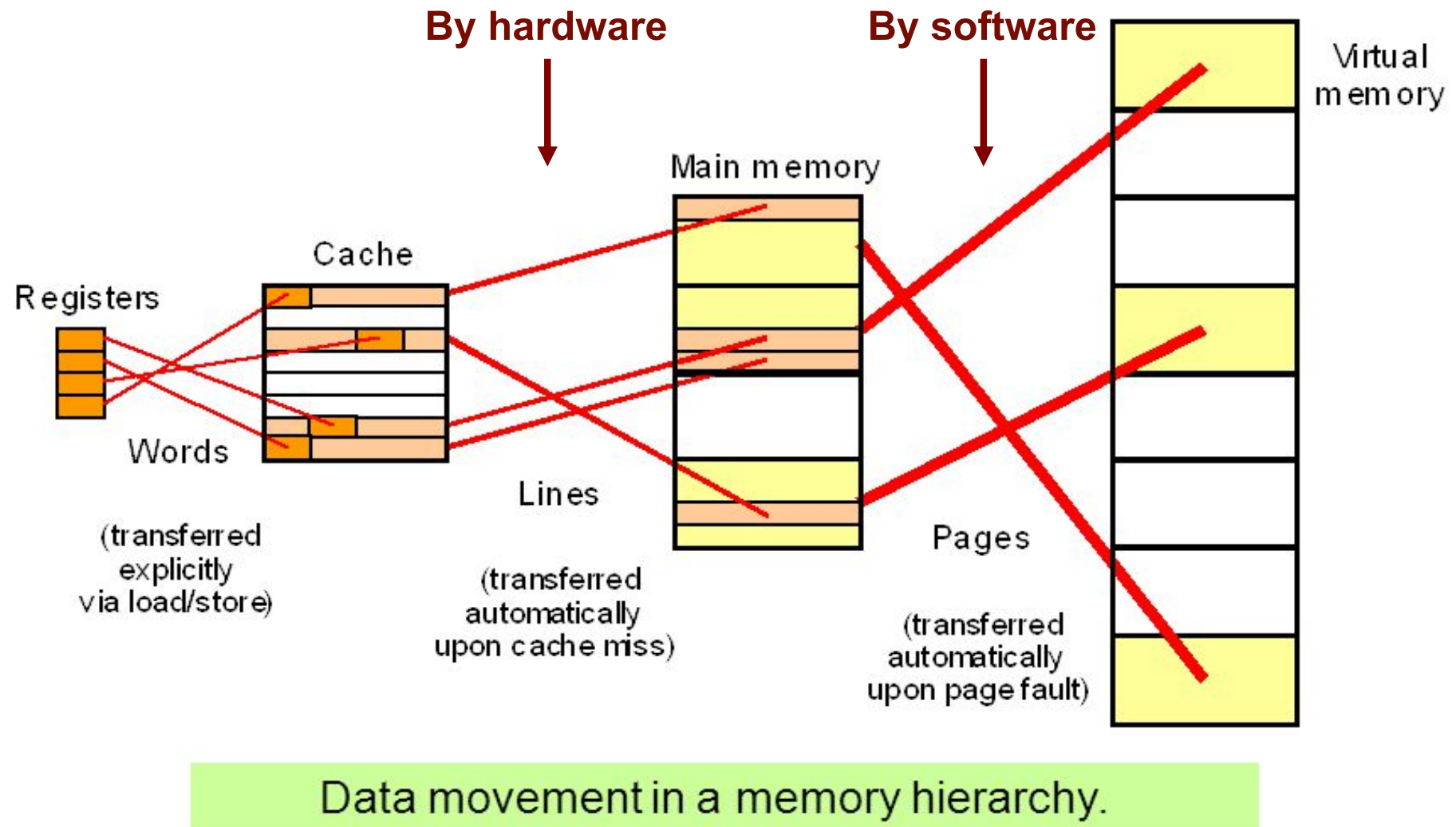
- Review of memory hierarchy: App. B
- Multiprocessor cache coherence and snooping coherence protocol with example: 5.2
- Basics on directory-based cache coherence: 5.4
- Models of memory consistency: 5.6

- Programmers want unlimited amounts of memory with low latency
- Fast memory technology is more expensive per bit than slower memory
- Solution: organize memory system into a hierarchy
  - Entire addressable memory space available in largest, slowest memory
  - Incrementally add smaller and faster memories, each containing a subset of the memory below it, proceed in steps up toward the processor
- Temporal and spatial locality insures that nearly all references can be found in smaller memories
  - Gives the illusion of a large, fast memory being presented to the processor

# Memory Performance Gap



# Memory hierarchy: the big picture



# Memory Hierarchy Design

- Memory hierarchy design becomes more crucial with recent multi-core processors:
  - Aggregate peak bandwidth grows with # cores:
    - Intel Core i7 can generate two references per core per clock
    - Four cores and 3.2 GHz clock
      - 25.6 billion\* 64-bit data references/second +
      - 12.8 billion\* 128-bit instruction references
      - = 409.6 GiB/s!
  - DRAM bandwidth is only 6% of this (25 GiB/s)
  - Requires:
    - Multi-port, pipelined caches
    - Two levels of cache per core
    - Shared third-level cache on chip

See exercise  
next slide

See exercise  
next slide

\* US billion =  $10^9$

# Exercise 1 on memory hierarchy



Consider the following study case:

- execution of a piece of code in the SeARCH node with the Xeon Skylake (Gold 6130); detailed info on this Intel microarchitecture in [https://en.wikichip.org/wiki/intel/microarchitectures/skylake\\_\(server\)](https://en.wikichip.org/wiki/intel/microarchitectures/skylake_(server)) ;
- execution of the same 2 instructions (that are already in the instruction cache) in all cores of a single chip: load in register a double followed by a multiplication by another double in a memory distant location;
- the Skylake cores are 6-way superscalar and each core has 2 load units;
- these instructions are executed with a cold data cache.

## Compute:

- a) The** max required bandwidth to access the external RAM when executing these 2 instructions (to simplify, consider clock rate = 2 GHz).
- b) The** aggregate peak bandwidth available in this Xeon device to access the installed DRAM-4 (using all memory channels).

# Exercise 2 on memory hierarchy



Similar to problem 1 (same node/chip in the cluster), but consider now:

- execution of code taking advantage of the AVX-512 facilities;
- execution of the same 2 vector instructions (that are already in the instruction cache) in all cores: load in register a vector of doubles followed by a multiplication by another vector of doubles in memory;
- the Skylake cores are 6-way superscalar and 2-way MT, and each core supports 2 simultaneous vector loads;
- the Skylake 6130 base clock rate with **AVX-512** code is **1.3 GHz**;
- these instructions are executed with a cold data cache.

## Compute/estimate:

**The** max required bandwidth to access the external RAM when executing these 2 vector instructions.

Compare with the peak bandwidth computed before.

\* [https://en.wikichip.org/wiki/intel/xeon\\_gold/6130](https://en.wikichip.org/wiki/intel/xeon_gold/6130)

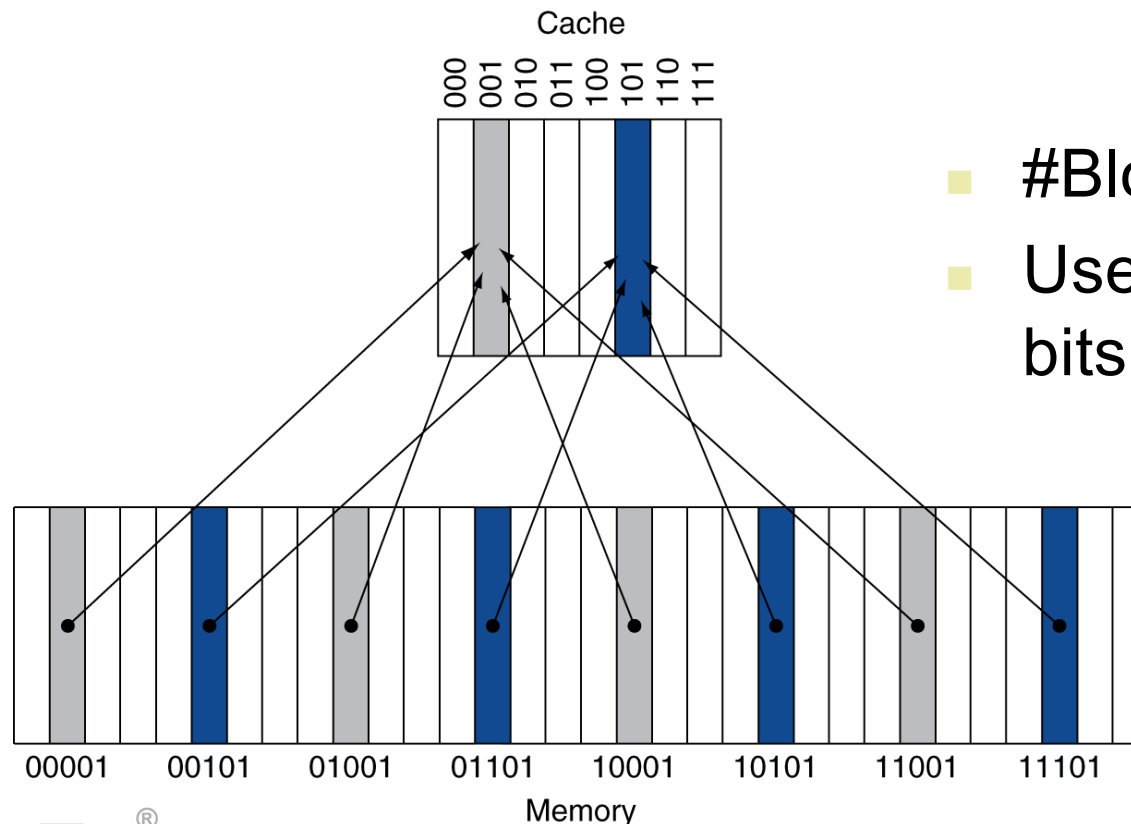
# The Memory Hierarchy

## The BIG Picture

- Common principles apply at all levels of the memory hierarchy
  - Based on notions of caching
- At each level in the hierarchy
  - Block placement
  - Finding a block
  - Replacement on a miss
  - Write policy

# Direct Mapped Cache

- Location determined by address
- Direct mapped: only one choice
  - (Block address) modulo (#Blocks in cache)



- #Blocks is a power of 2
- Use low-order address bits

# Associative Caches

- Fully associative
  - Allow a given block to go in any cache entry
  - Requires all entries to be searched at once
  - Comparator per entry (expensive)
- $n$ -way set associative
  - Each set contains  $n$  entries
  - Block number determines which set
    - (Block number) modulo (#Sets in cache)
  - Search all entries in a given set at once
  - $n$  comparators (less expensive)

# How Much Associativity

- Increased associativity decreases miss rate
  - But with diminishing returns
- Simulation of a system miss rate with 64KiB D-cache, 16-word blocks, SPEC2000
  - 1-way: 10.3%
  - 2-way: 8.6%
  - 4-way: 8.3%
  - 8-way: 8.1%

# Block Placement

- Determined by associativity
  - Direct mapped (1-way associative)
    - One choice for placement
  - n-way set associative
    - n choices within a set
  - Fully associative
    - Any location
- Higher associativity reduces miss rate
  - Increases complexity, cost, and access time

# Replacement Policy

- Direct mapped: no choice
- Set associative
  - Prefer non-valid entry, if there is one
  - Otherwise, choose among entries in the set
- Least-recently used (LRU)
  - Choose the one unused for the longest time
    - Simple for 2-way, manageable for 4-way, too hard beyond that
- Random
  - Gives approximately the same performance as LRU for high associativity

# Write Policy

- Write-through
  - Update both upper and lower levels
  - Simplifies replacement, but may require write buffer
- Write-back
  - Update upper level only
  - Update lower level when block is replaced
  - Need to keep more state
- Virtual memory
  - Only write-back is feasible, given disk write latency

# Memory Hierarchy Basics

$$CPU_{exec-time} = (CPU_{clock-cycles} + Mem_{stall-cycles}) \times \text{Clock cycle time}$$

$$CPU_{exec-time} = (IC \times CPI_{CPU} + Mem_{stall-cycles}) \times \text{Clock cycle time}$$

With introduction of a single-level cache:

$$Mem_{stall-cycles} = IC \times \dots \text{Miss rate} \dots \text{Mem accesses} \dots \text{Miss penalty} \dots$$

$$Mem_{stall-cycles} = IC \times \text{Misses/Instruction} \times \text{Miss Penalty}$$

$$\frac{\text{Misses}}{\text{Instruction}} = \frac{\text{Miss rate} \times \text{Memory accesses}}{\text{Instruction count}} = \text{Miss rate} \times \frac{\text{Memory accesses}}{\text{Instruction}}$$

For each additional cache-level  $i$  (including LLC to memory):

$$Mem\_accesses_{level\_i} = Misses/Instruction_{level\_i-1}$$

$$Miss\_penalty_{level\_i} = (\text{Hit\_rate} \times \text{Hit\_time} + \text{Miss\_rate} \times \text{Miss\_penalty})_{level\_i+1}$$

# Exercise 3 on Cache Performance

- Given
  - I-cache miss rate = 2%
  - D-cache miss rate = 4%
  - Miss penalty = 100 cycles
  - Base CPI (ideal cache) = 2
  - Load & stores are 36% of instructions
- Miss cycles per instruction
  - I-cache:  $?? \times ?? = ??$
  - D-cache:  $?? \times ?? \times ?? = ??$
- Actual CPI =  $2 + ?? + ?? = ??$

# Multilevel Caches

- Primary cache attached to CPU
  - Small, but fast
- Level-2 cache services misses from primary cache
  - Larger, slower, but still faster than ~~main memory~~ L3
- L-3 cache or main memory services L-2 cache misses
  - Larger, slower, but still faster than main memory
- Main memory services L-3 cache misses  
(and eventually L-2 cache misses, if L-3 is non-inclusive)

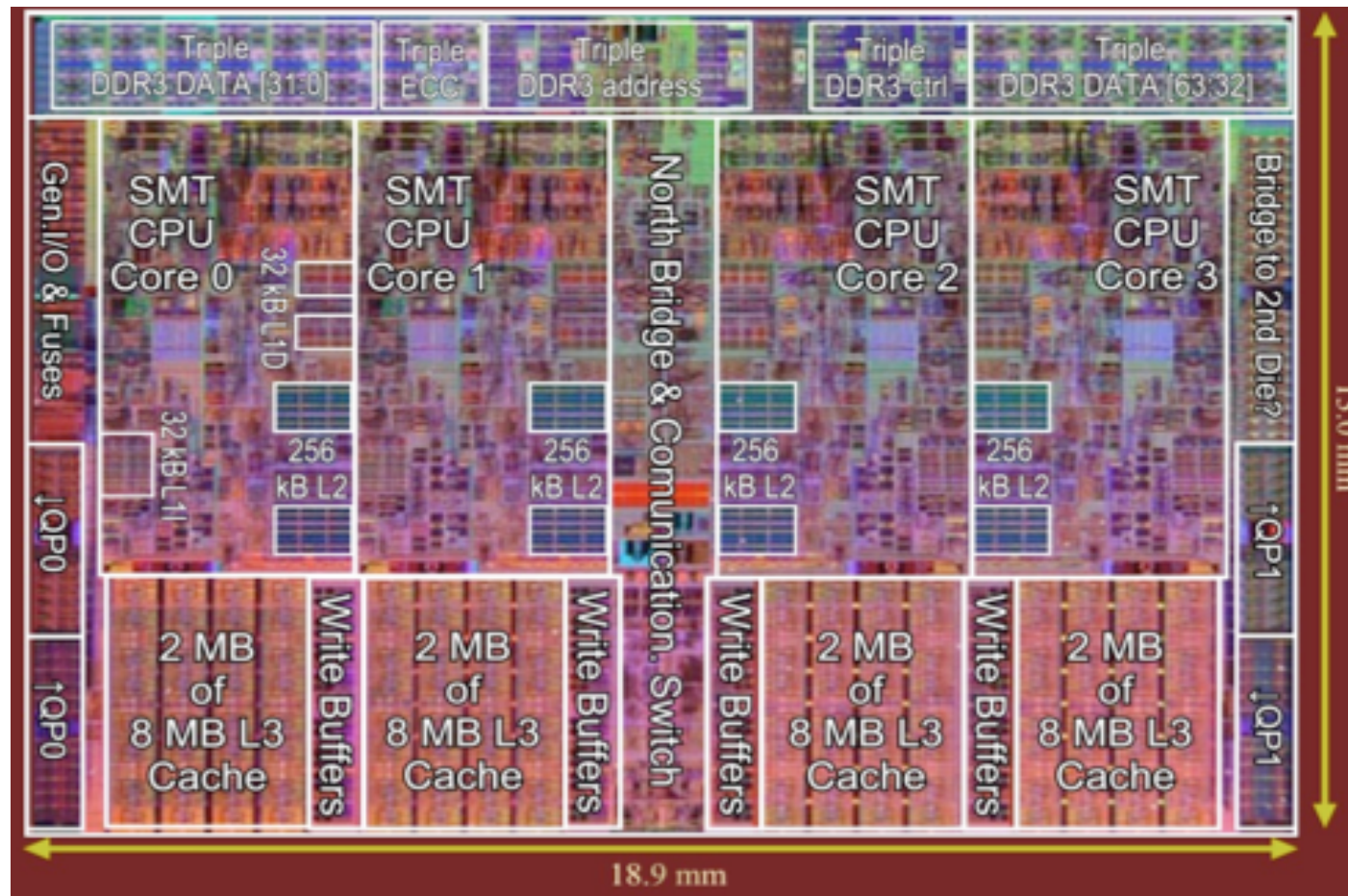


# Exercise 4 on Multilevel Cache

- Given
  - CPU base **CPI = 1**, clock rate = 4GHz
  - Miss rate/instruction = 2%
  - Main memory access time = 100ns
- With just primary cache
  - Miss penalty =  $100\text{ns} / 0.25\text{ns} = 400$  cycles
  - Effective **CPI = 9** ( $= 1 + 0.02 \times 400$ )
- Now add L-2 cache ...
  - Access time = 5ns
  - Global miss rate to main memory = 0.5%
- $\text{CPI} = 1 + \text{??} \times \text{??} + \text{??} \times \text{??} = \text{??}$
- Performance ratio =  $9 / \text{??} = \text{??}$

# Multilevel On-Chip Caches

Intel Nehalem 4-core processor



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

# 3-Level Cache Organization

	Intel Nehalem	AMD Opteron X4
L1 caches (per core)	L1 I-cache: 32KB, 64-byte blocks, 4-way, <b>approx LRU replacement</b> , hit time n/a L1 D-cache: 32KB, 64-byte blocks, 8-way, approx LRU replacement, write-back/allocate, hit time n/a	L1 I-cache: 32KB, 64-byte blocks, 2-way, <b>approx LRU replacement</b> , hit time 3 cycles L1 D-cache: 32KB, 64-byte blocks, 2-way, <b>approx LRU replacement</b> , write-back/allocate, hit time 9 cycles
L2 unified cache (per core)	256KB, 64-byte blocks, 8-way, <b>approx LRU replacement</b> , write-back/allocate, hit time n/a	512KB, 64-byte blocks, 16-way, <b>approx LRU replacement</b> , write-back/allocate, hit time n/a
L3 unified cache (shared)	8MB, 64-byte blocks, 16-way, replacement n/a, write-back/allocate, hit time n/a	2MB, 64-byte blocks, 32-way, replace block shared by fewest cores, write-back/allocate, hit time 32 cycles

n/a: data not available

# Exercise 5 on multilevel performance



## Characterize the memory system of Xeon Skylake Gold 6130:

### 1. L1 I-cache

- size ? KiB/core, ?-way set associative, ? sets, line size ? B, hit time ? cycles, ? B/cycle on transfer bandwidth L1 to the instruction fetch unit

### L1 D-cache

- size ? KiB/core, ?-way set associative, ? sets, line size ? B, hit time ? cycles, ? B/cycle on load bandwidth L1 to load buffer unit

### 2. L2 cache

- size ? KiB/core, ?-way set associative, ? sets, line size ? B, hit time ? cycles, ? B/cycle on load bandwidth L2 to L1

### 3. L3 cache

- size ? KiB/core, ?-way set associative, ? sets, line size ? B, hit time ? cycles, ? B/cycle on load bandwidth L3 to L2

### 4. DRAM, DDR4-2666

- up to ? GT/s, bandwidth ? GB/s per channel, ? mem channels, aggregate bandwidth ? GB/s, ? B/cycle on peak load bandwidth DRAM to L3, NUMA-local latency ? ns, NUMA-remote latency ? ns

# Intel new cache approach with Skylake



## Re-Architected L2 & L3 Cache Hierarchy



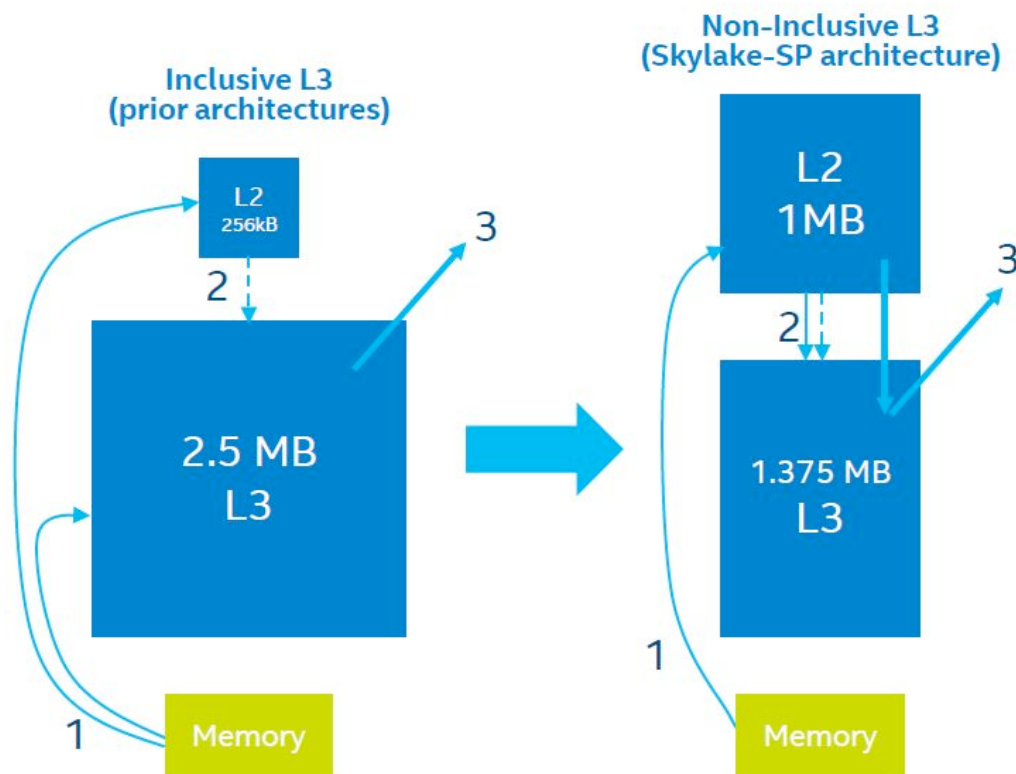
- On-chip cache balance shifted from shared-distributed (prior architectures) to private-local (Skylake architecture):
  - Shared-distributed → shared-distributed L3 is primary cache
  - Private-local → private L2 becomes primary cache with shared L3 used as overflow cache
- Shared L3 changed from inclusive to non-inclusive:
  - Inclusive (prior architectures) → L3 has copies of all lines in L2
  - Non-inclusive (Skylake architecture) → lines in L2 **may not** exist in L3

**SKYLAKE-SP CACHE HIERARCHY ARCHITECTED SPECIFICALLY FOR DATA CENTER USE CASE**

# Intel new cache approach with Skylake



## Inclusive vs Non-Inclusive L3



1. Memory reads fill directly to the L2, no longer to both the L2 and L3
2. When a L2 line needs to be removed, both modified and unmodified lines are written back
3. Data shared across cores are copied into the L3 for servicing future L2 misses

### Cache hierarchy architected and optimized for data center use cases:

- Virtualized use cases get larger private L2 cache free from interference
- Multithreaded workloads can operate on larger data per thread (due to increased L2 size) and reduce uncore activity

# Exercise 6 on multilevel performance



Similar to problem 1 (same node/chip in the cluster, code), but consider now:

- execution of scalar code in a **2 GHz** single-core (already in L1 I-cache);
- code already takes advantage of all data cache levels (**L1, L2 & L3**), where 50% of data is placed on the RAM modules in the memory channels of the other PU chip (**NUMA architecture**);
- remember: the Skylake cores are **6-way superscalar** and **2-way MT**, and each core supports **2 simultaneous loads**;
- **cache latency time on hit**: take the average of the specified values;
- **memory latency**: 80 nsec (**NUMA local**), 120 nsec (**NUMA remote**);
- **miss rate per instruction** :
  - at **L1: 2%**; at **L2: 50%**; at **L3: 80%** (these are not global values!).

## Compute/estimate:

1. **The miss penalty** per instruction at each cache level.
2. **The average memory stall cycles** per instruction that degrades CPI.