# Master Informatics Eng.

2019/20

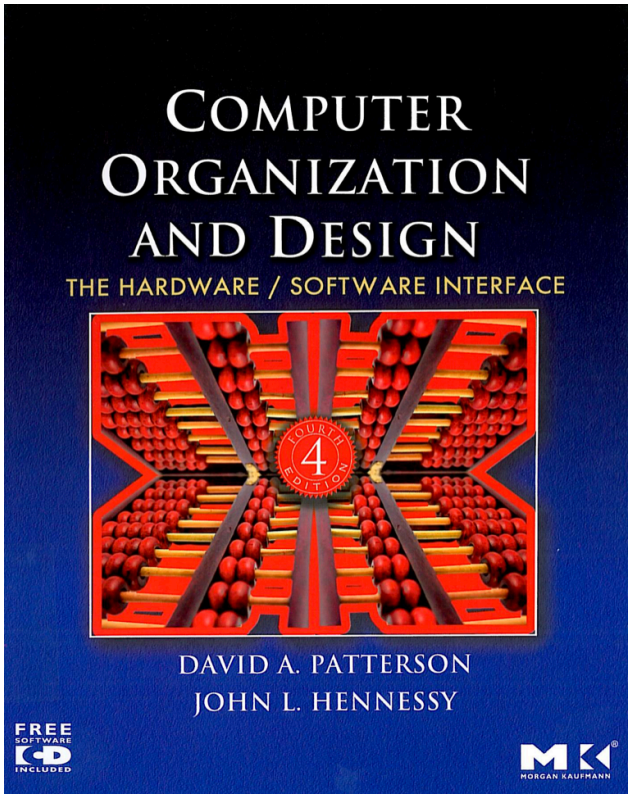
*A.J.Proença*

## Concepts from undegrad Computer Systems

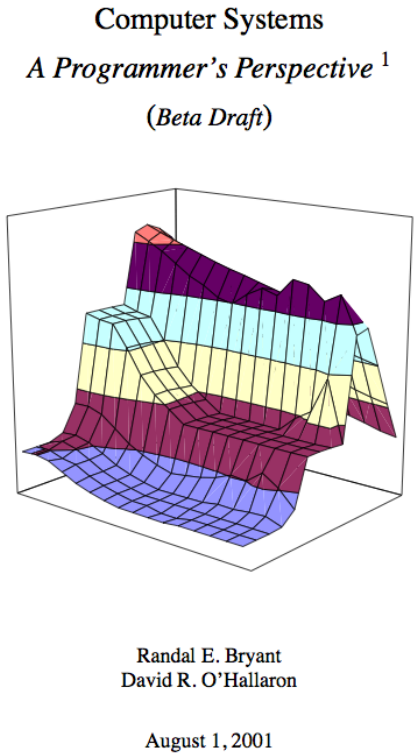
***(most slides are borrowed, mod's in green)***

# Concepts from undegrad Computer Systems

*– most slides are borrowed from*




*and some from*




*more details at*
*http://gec.di.uminho.pt/miei/sc/*

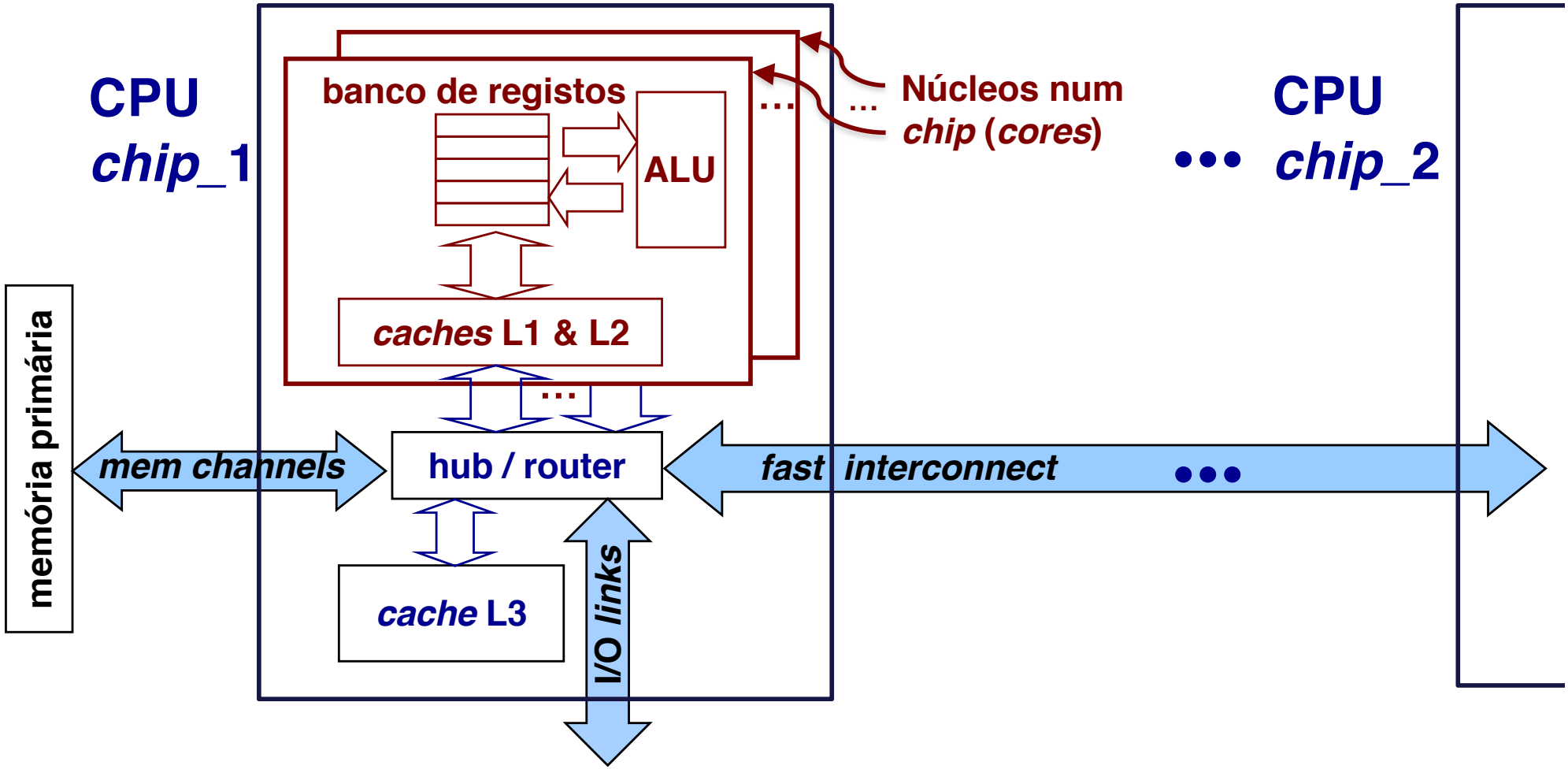
# Background for Advanced Architectures

## Key concepts to revise:

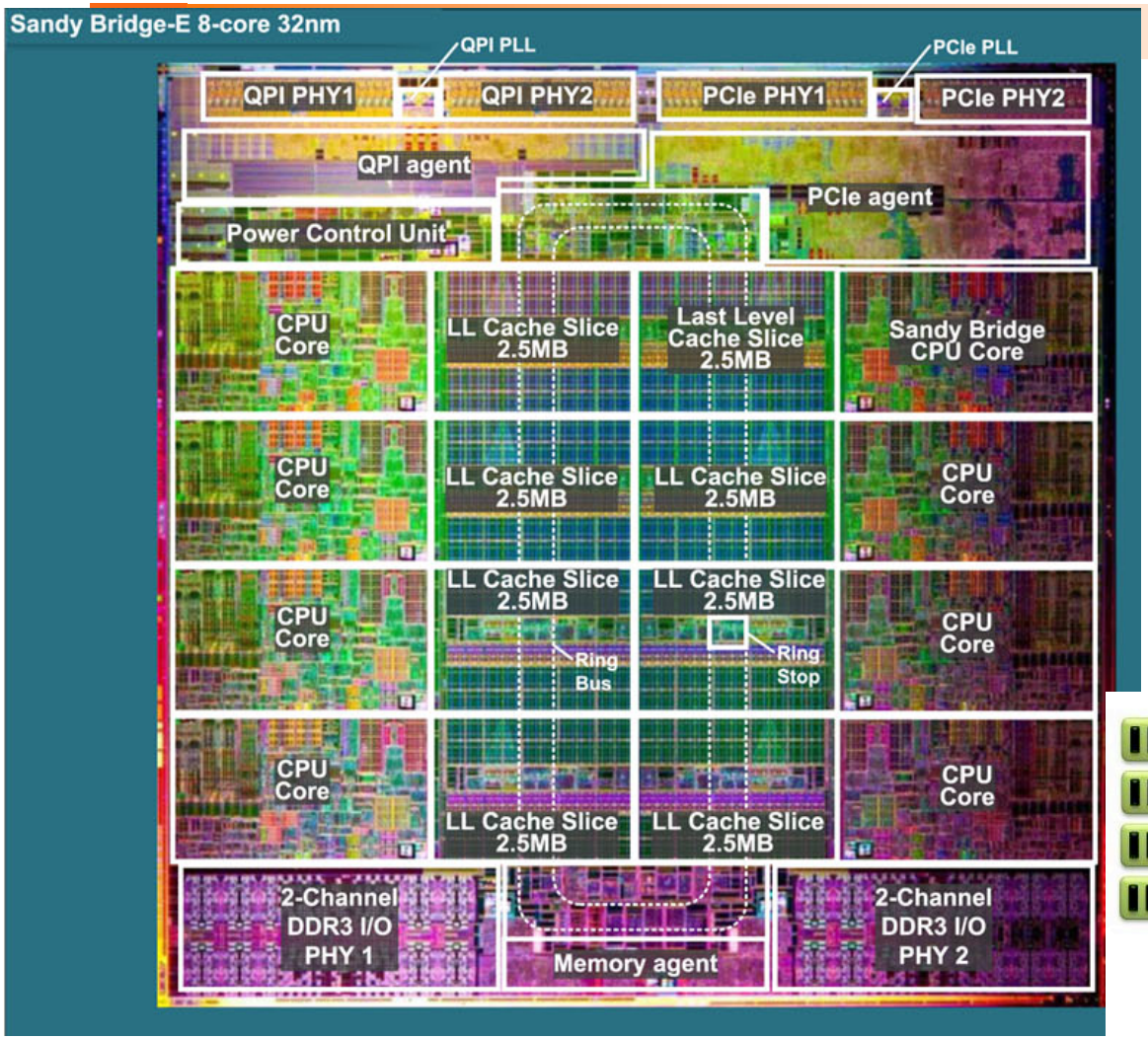
- *numerical data representation (for error analysis)*
- *ISA (Instruction Set Architecture)*
- *how C compilers generate code (a look into assembly code)*
  - *how scalar and structured data are allocated*
  - *how control structures are implemented*
  - *how to call/return from function/procedures*
  - *what architecture features impact performance*
- ***Improvements to enhance performance in a single CPU***
  - ***ILP: pipeline, multiple issue, …***
  - ***data parallelism: SIMD/vector processing, ...***
  - ***thread-level parallelism***
  - ***memory hierarchy: cache levels, ...***

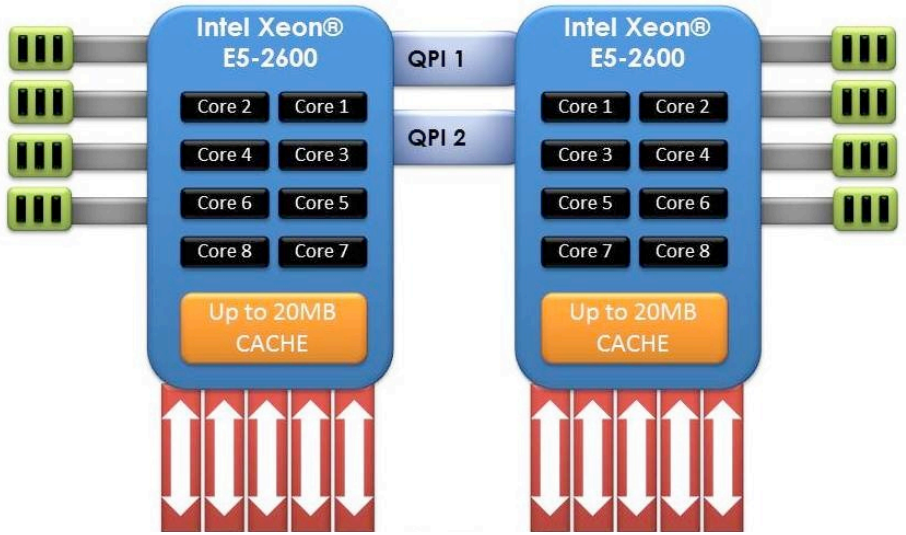
# *A hierarquia de cache em arquiteturas multicore*

## As arquiteturas *multicore* mais recentes:

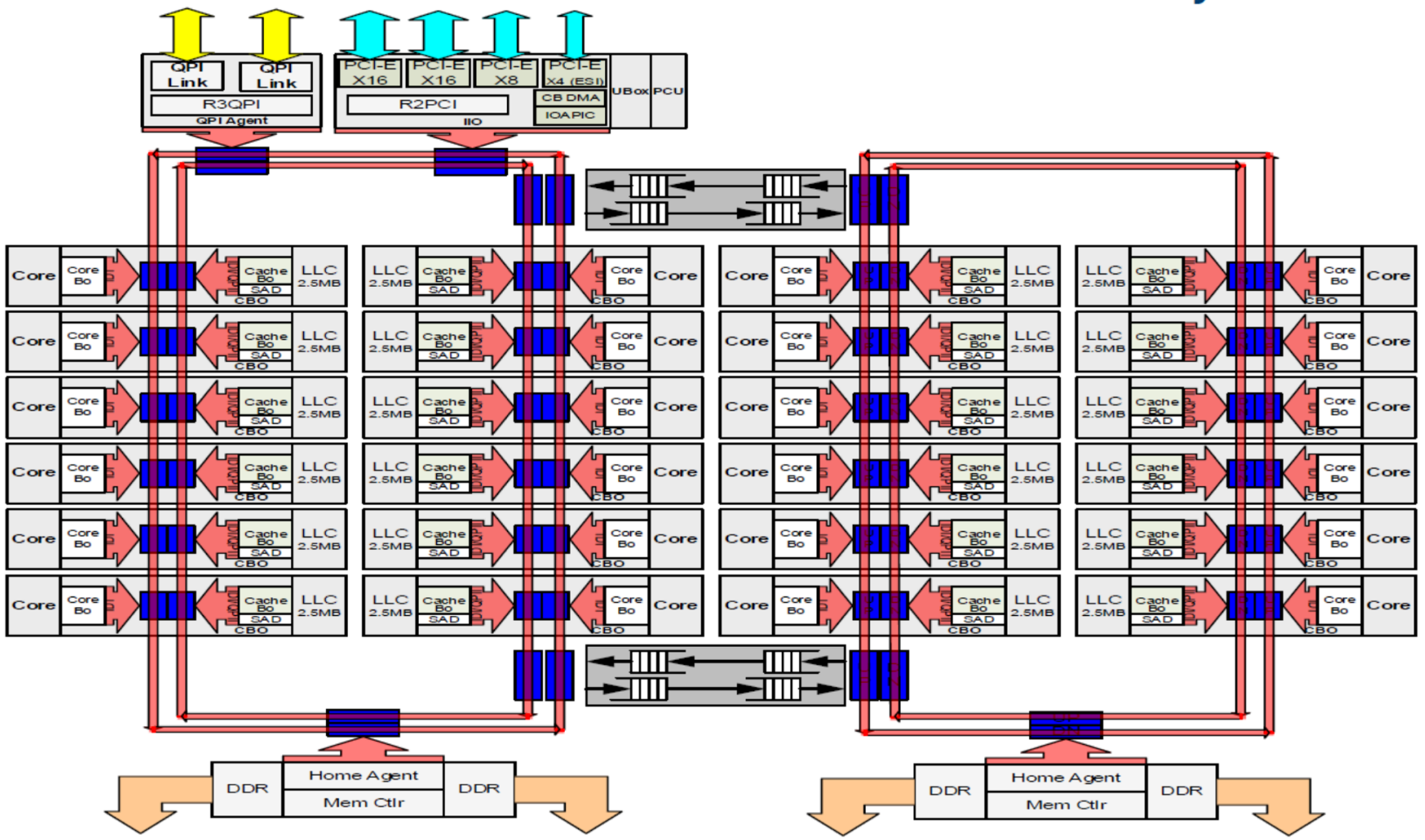


# *Lançamento da Intel em 2012: Sandy/Ivy Bridge (8-core)*

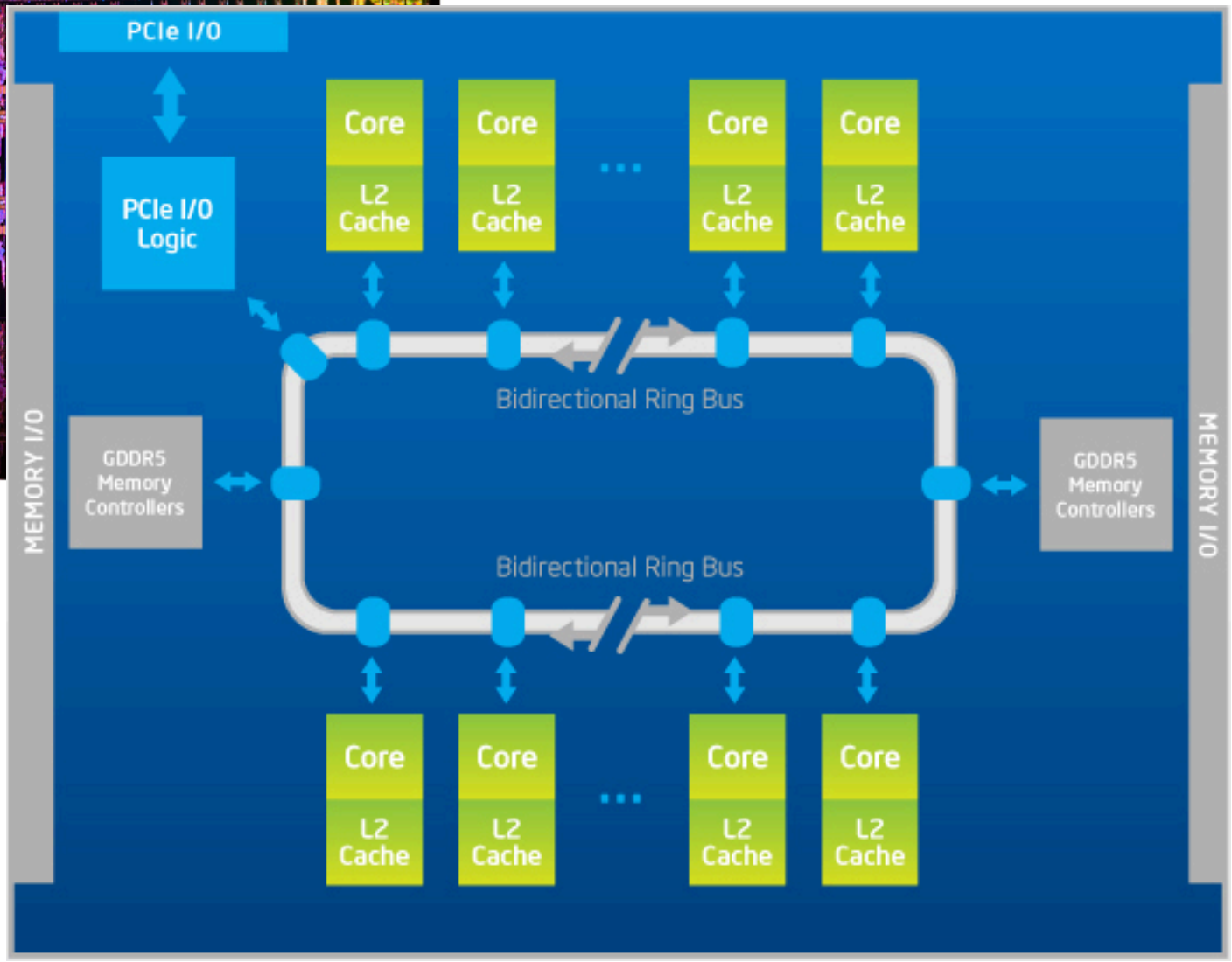


# *Intel in 2016:*
# *Broadwell-EP Xeon (22-core)*

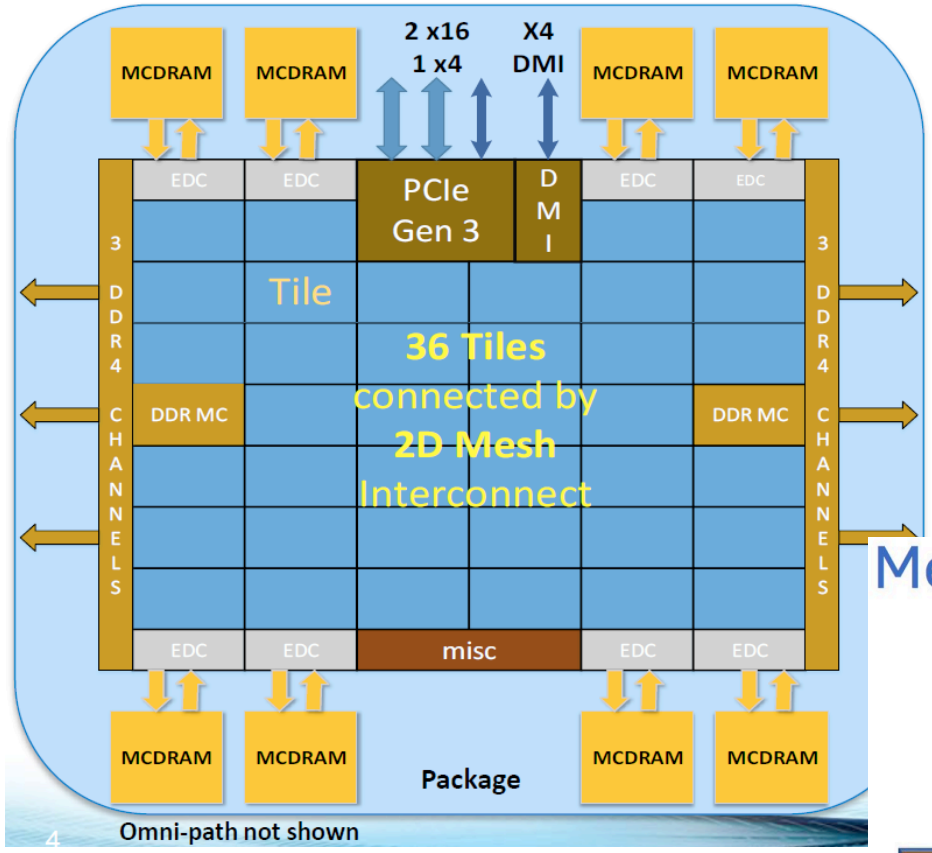


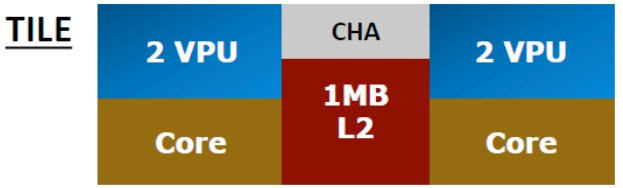
# Chips da Intel em 2012/13: Xeon Phi com 60 cores



# Intel new Phi in 2016: KNL with 72 cores

## Knights Landing Overview



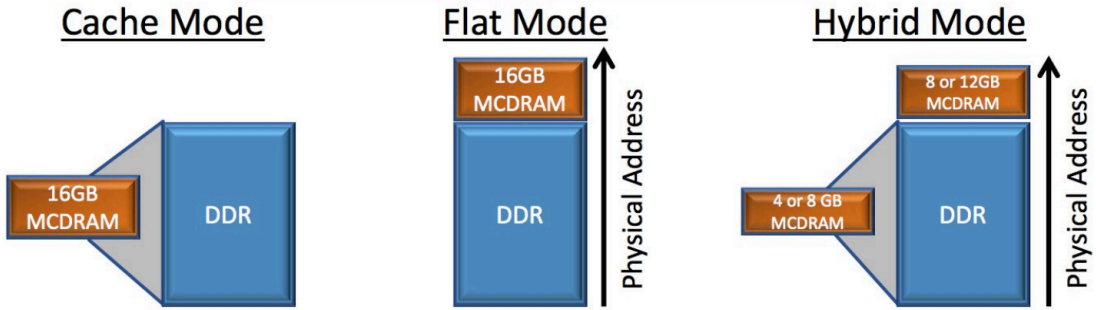




**Chip: 36 Tiles** interconnected by **2D Mesh**
**Tile**: 2 Cores + 2 VPU/core + 1 MB L2

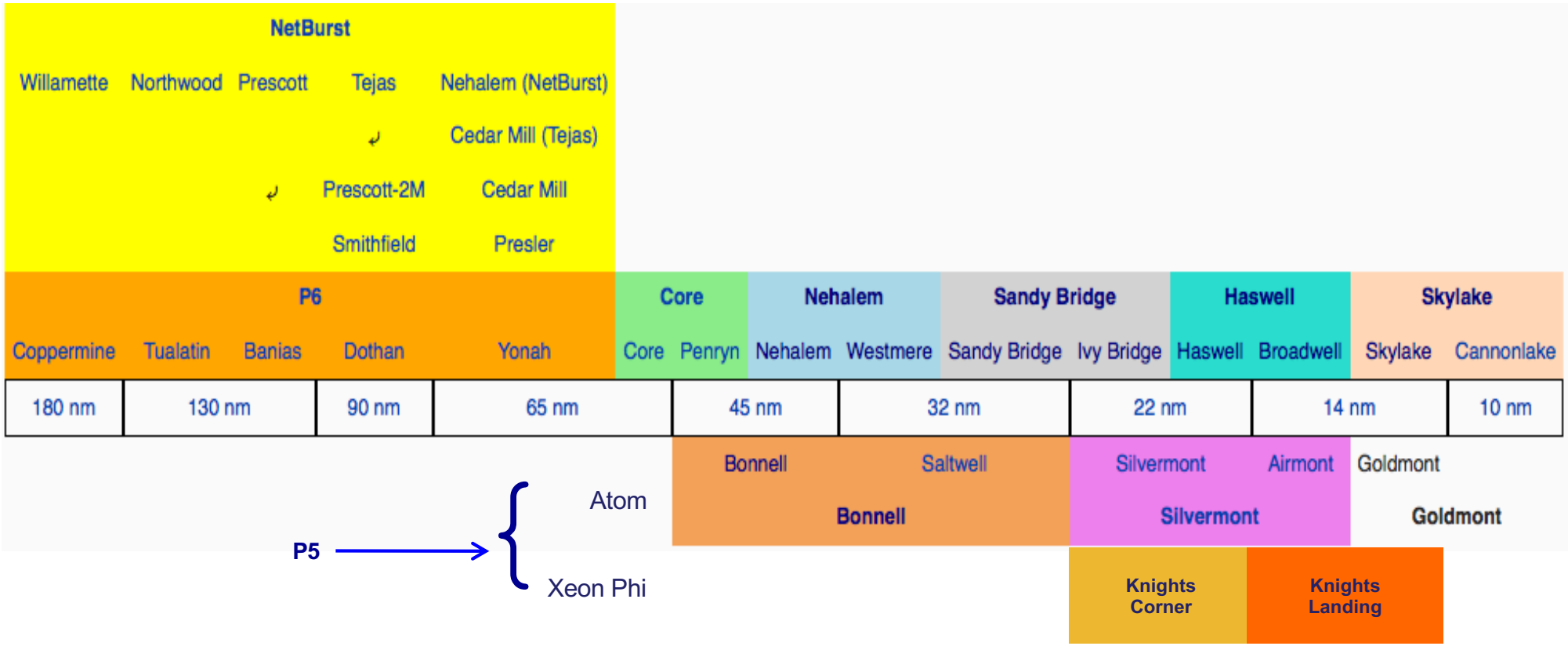
**Memory: MCDRAM:** 16 GB on-package; High BW
**DDR4**: 6 channels @ 2400 up to 384GB
**IO:** 36 lanes PCIe Gen3. 4 lanes of DMI for chipset
**Node**: 1-Socket only
**Fabric:** Omni-Path on-package (not shown)

## Memory Modes

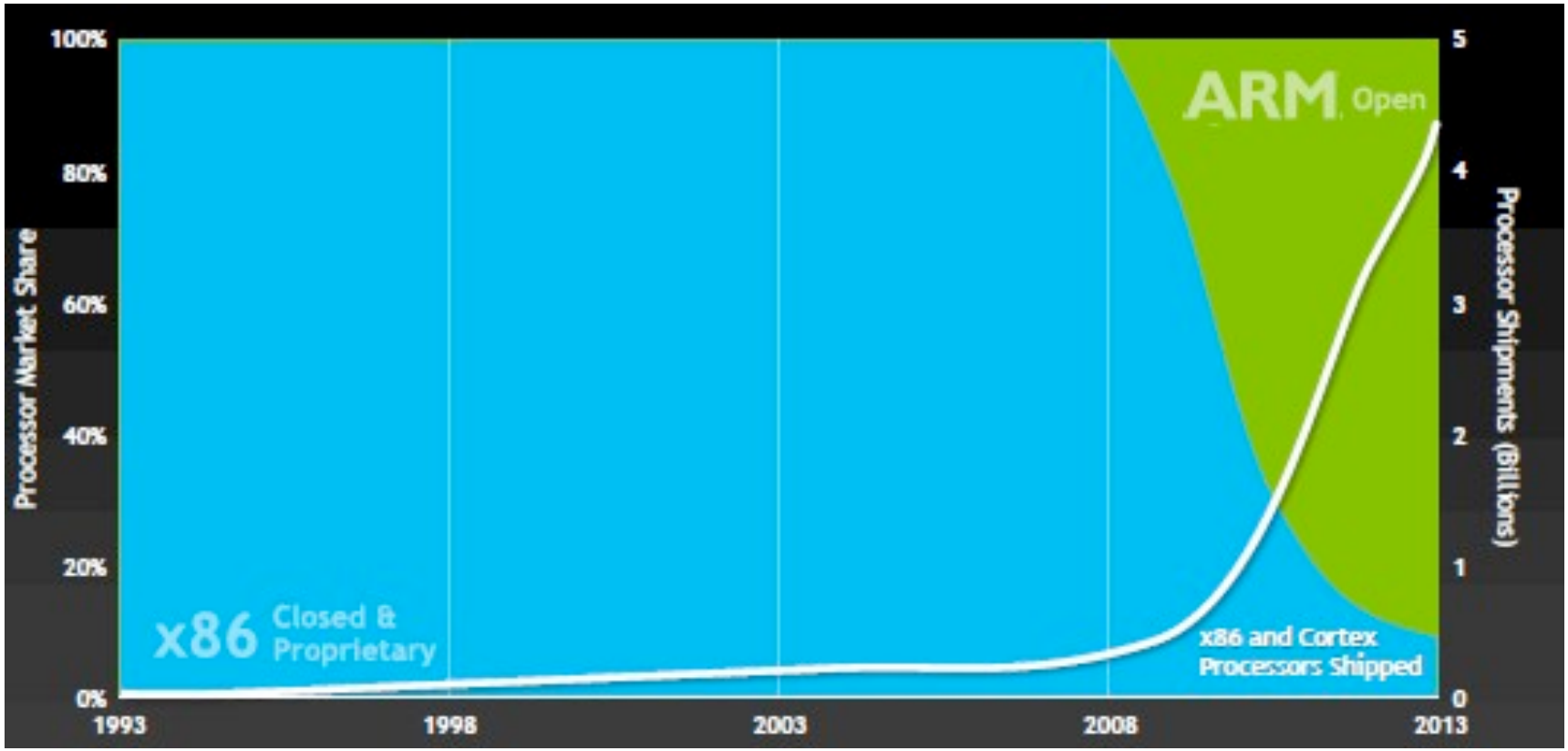
**Three** Modes. Selected at boot



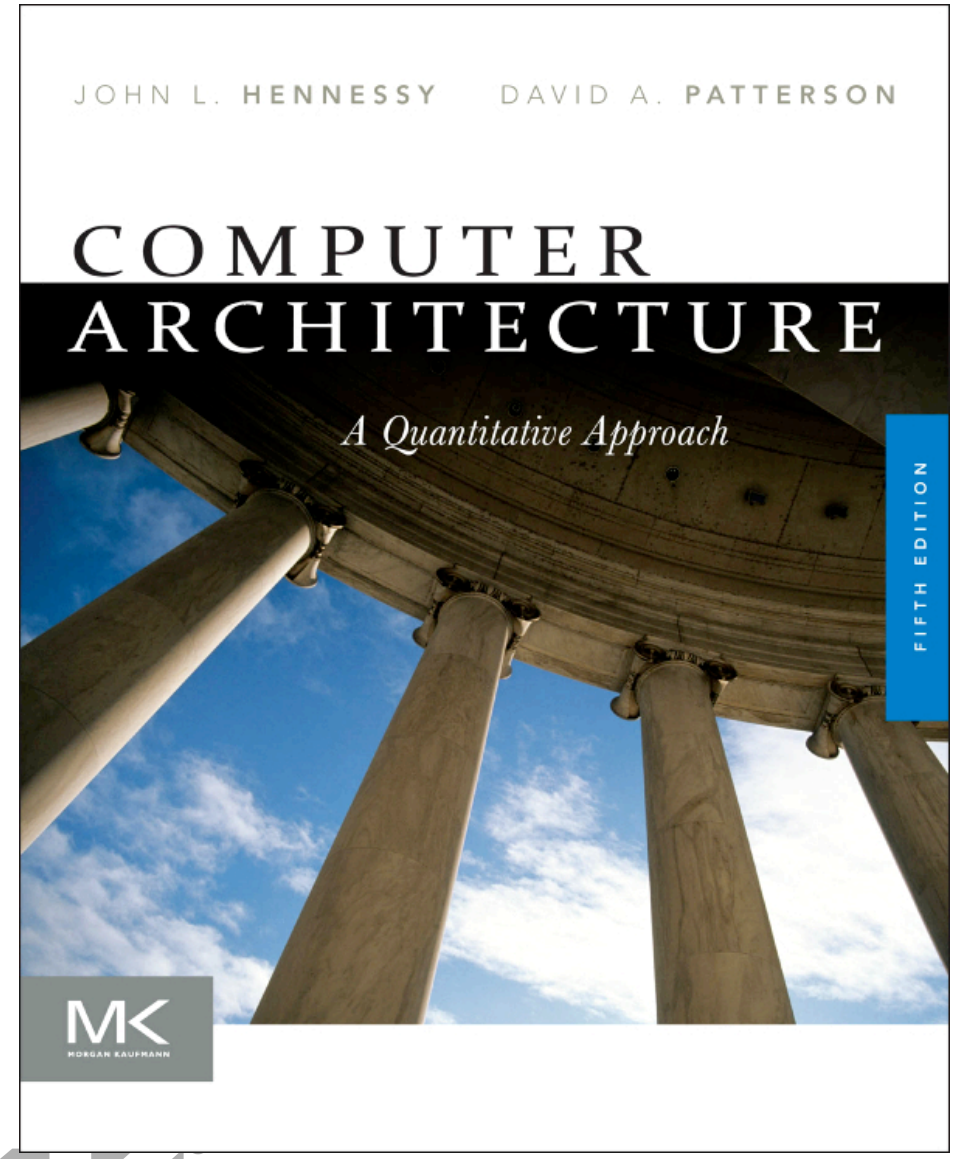
# Internal x86 roadmap



# Processadores Intel x86 versus ARM



# Key textbook in Advanced Architecture




**Computer Architecture, 5th Edition**

**Hennessy & Patterson**

## Table of Contents

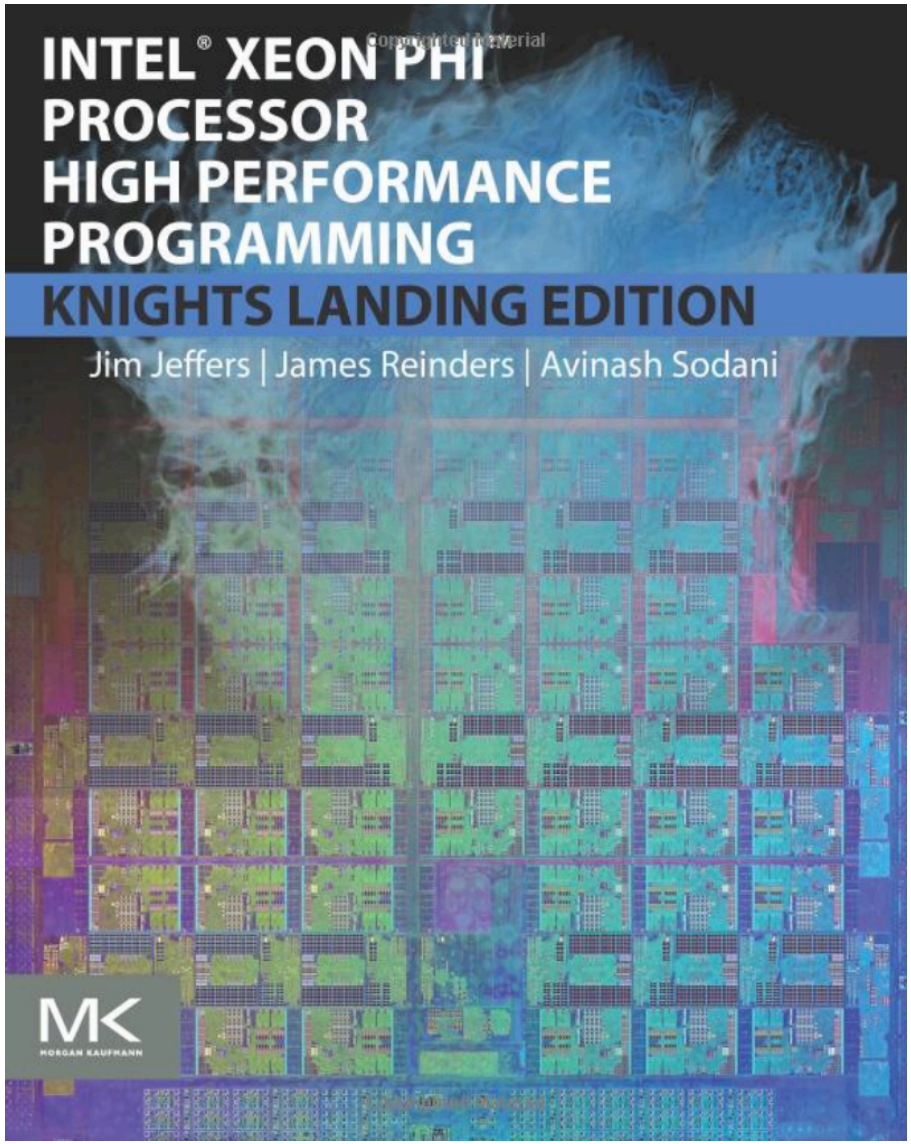
**Printed Text**

Chap 1: Fundamentals of Quantitative Design and Analysis
Chap 2: Memory Hierarchy Design
Chap 3: Instruction-Level Parallelism and Its Exploitation
Chap 4: Data-Level Parallelism in Vector, SIMD, and GPU Architectures
Chap 5: Multiprocessors and Thread-Level Parallelism
Chap 6: The Warehouse-Scale Computer
App A: Instruction Set Principles
App B: Review of Memory Hierarchy
App C: Pipelining: Basic and Intermediate Concepts

**Online**

App D: Storage Systems
App E: Embedded Systems
App F: Interconnection Networks
App G: Vector Processors
App H: Hardware and Software for VLIW and EPIC
App I: Large-Scale Multiprocessors and Scientific Applications
App J: Computer Arithmetic
App K: Survey of Instruction Set Architectures
App L: Historical Perspectives

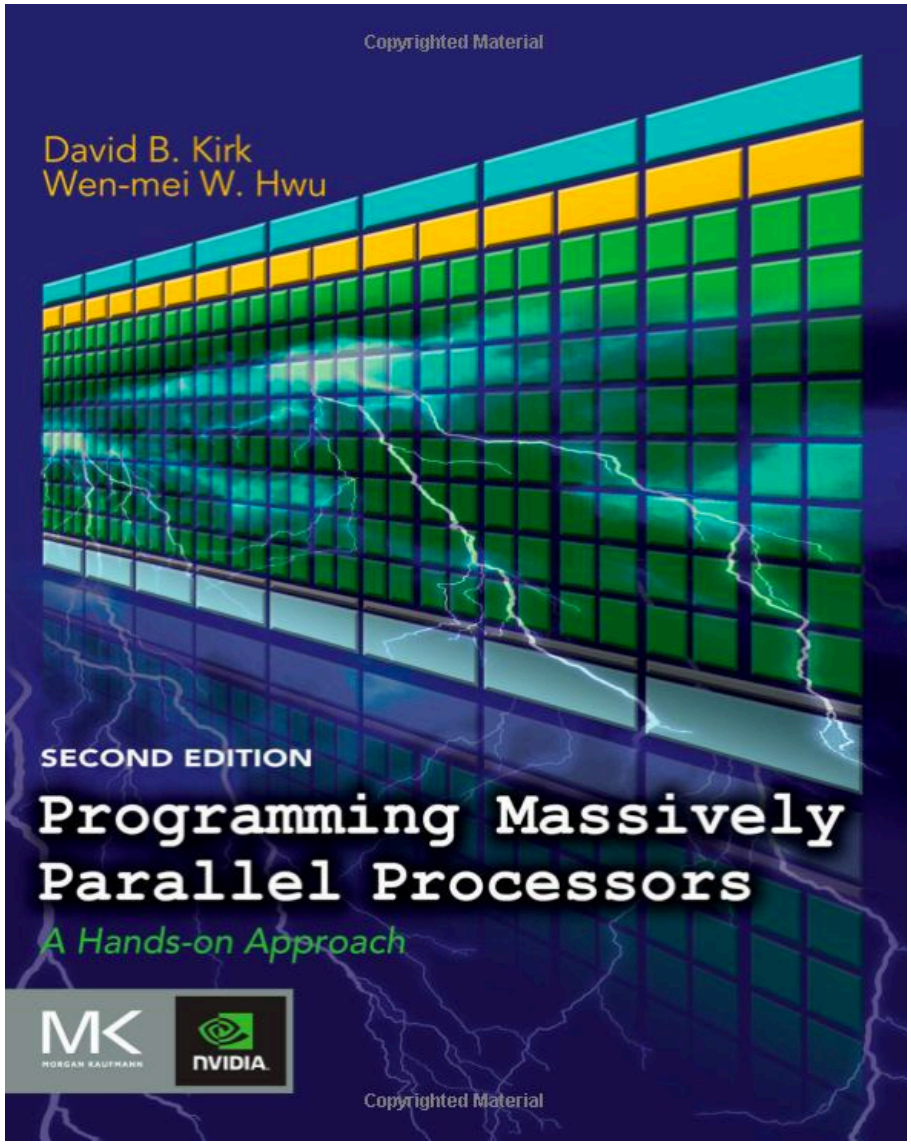
# Recommended textbook (1)




## Table of Contents



# Recommended textbook (2)




## Contents

- Algorithm + Data Structures
  - Determines number of operations executed
  - Determines how efficient data is assessed
- Programming language, compiler, architecture
  - Determine number of machine instructions executed per operation
- Processor and memory system
  - Determine how fast instructions are executed
- I/O system (including OS)
  - Determines how fast I/O operations are executed

# Response Time and Throughput

- Response time
  - How long it takes to do a task
- Throughput
  - Total work done per unit time
    - e.g., tasks/transactions/… per hour
- How are response time and throughput affected by
  - Replacing the processor with a faster version?
  - Adding more processors?
- We'll focus on response time for now…

$$\text{CPU Time} = \text{CPU Clock Cycles} \times \text{Clock Cycle Time}$$

$$= \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}}$$

- Performance improved by
  - Reducing number of clock cycles
  - Increasing clock rate
  - Hardware designer must often trade off clock rate against cycle count

$$\text{Clock Cycles} = \text{Instruction Count} \times \text{Cycles per Instruction}$$

$$\text{CPU Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

$$= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}$$

- Instruction Count, **IC**, for a program
  - Determined by program, ISA and compiler
- Average cycles per instruction (**CPI**)
  - Determined by CPU hardware
  - If different instructions have different CPI
    - Average CPI affected by instruction mix

## The BIG Picture

$$\text{CPU Time} = \underbrace{\frac{\text{Instructions}}{\text{Program}}}_{\text{IC}} \times \underbrace{\frac{\text{Clock cycles}}{\text{Instruction}}}_{\text{CPI}} \times \underbrace{\frac{\text{Seconds}}{\text{Clock cycle}}}_{T_c}$$

- Performance depends on
  - Algorithm: affects IC, possibly CPI
  - Programming language: affects IC, CPI
  - Compiler: affects IC, CPI
  - Instruction set architecture: affects IC, CPI, $T_c$
  - Processor design: ILP, memory hierarchy, ...

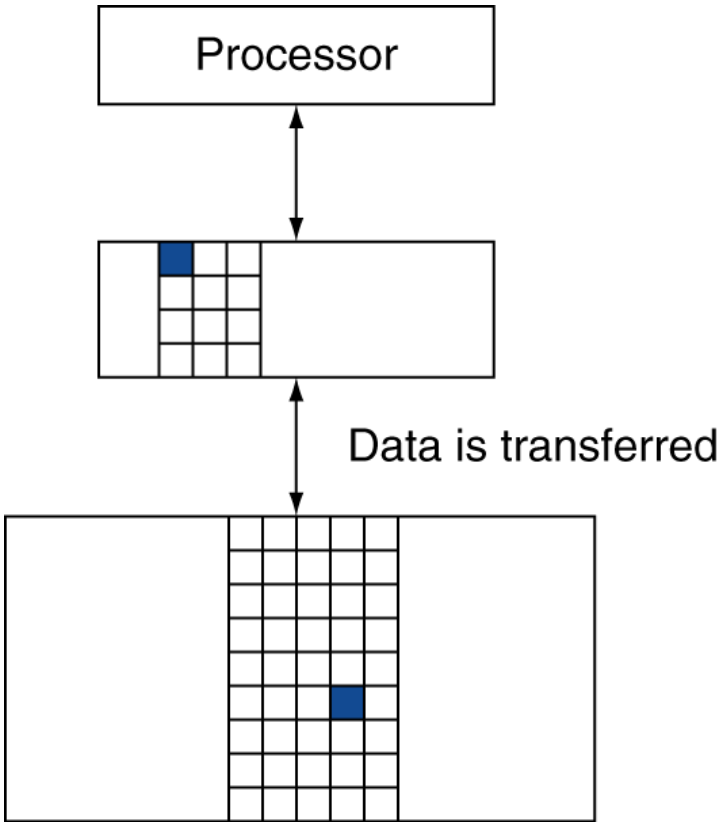
## The BIG Picture

- Pipelining improves performance by increasing instruction throughput
  - Executes multiple instructions in parallel
  - Each instruction has the same latency
- Subject to hazards
  - Structure, data, control
- Instruction set design affects complexity of pipeline implementation

## The BIG Picture

- Yes, but not as much as we'd like
- Programs have real dependencies that limit ILP
- Some dependencies are hard to eliminate
  - e.g., pointer aliasing
- Some parallelism is hard to expose
  - Limited window size during instruction issue
- Memory delays and limited bandwidth
  - Hard to keep pipelines full
- Speculation can help if done well

# Memory Hierarchy Levels




- Block (aka line): unit of copying
  - May be multiple words
- If accessed data is present in upper level
  - Hit: access satisfied by upper level
    - Hit ratio: hits/accesses
- If accessed data is absent
  - Miss: block copied from lower level
    - Time taken: miss penalty
    - Miss ratio: misses/accesses
      = 1 – hit ratio
  - Then accessed data supplied from lower level

## The BIG Picture

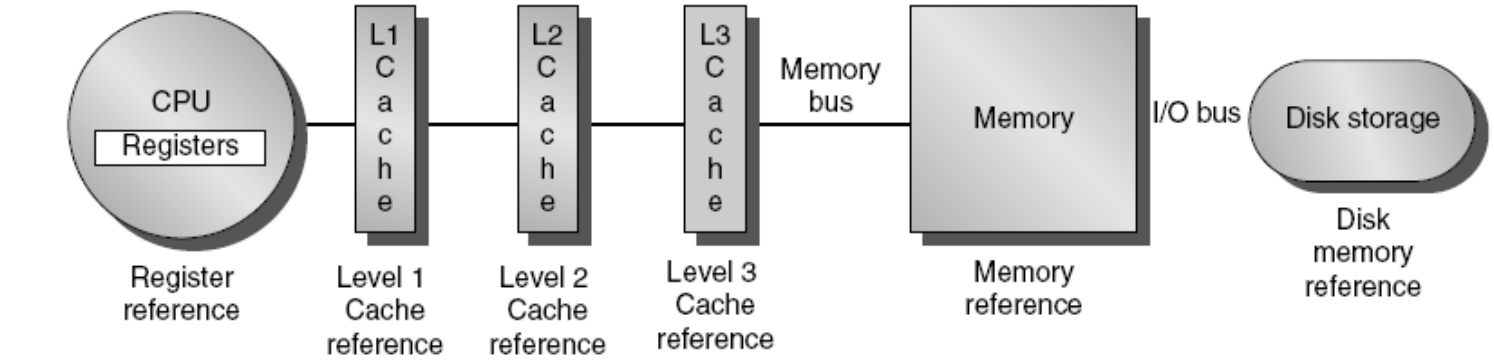
- Common principles apply at all levels of the memory hierarchy
  - Based on notions of caching
- Decisions at each level in the hierarchy
  - Block placement
  - Finding a block
  - Replacement on a miss
  - Write policy

§ 5.5 A Common Framework for Memory Hierarchies

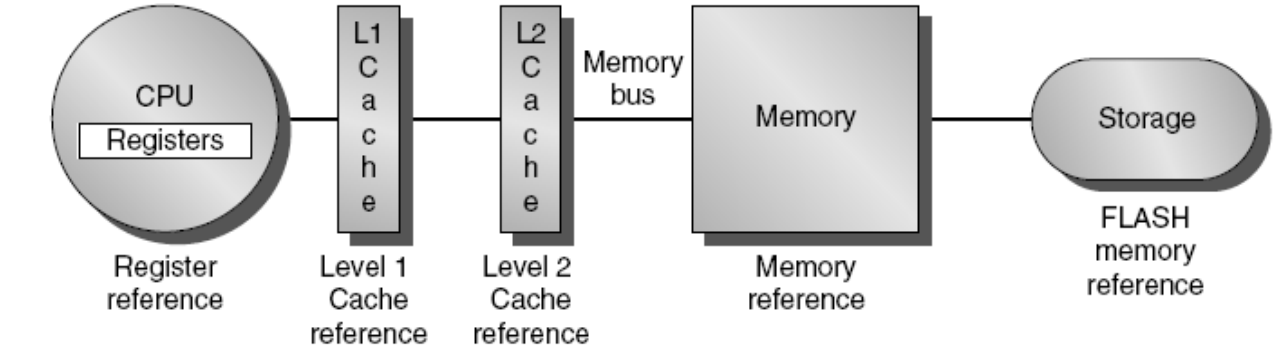
# Multilevel Caches

- Primary cache private to CPU/core
  - Small, but fast
- Level-2 cache services misses from primary cache
  - Larger, slower, but still faster than main memory
- High-end systems include L3 cache
- Main memory services L2/3 cache misses

# Memory Hierarchy




(a) Memory hierarchy for server




(b) Memory hierarchy for a personal mobile device