MSc Informatics Eng.

2014/15

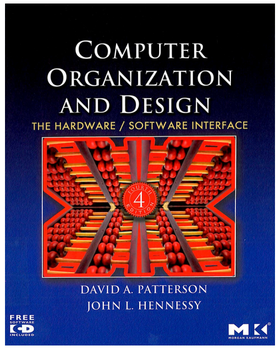
A.J.Proença

Concepts from undegrad Computer Systems (1)

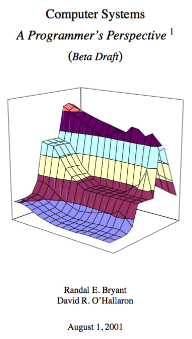
(most slides are borrowed, mod's in green)

Concepts from undegrad Computer Systems

– most slides are borrowed from


COMPUTER ORGANIZATION AND DESIGN
THE HARDWARE / SOFTWARE INTERFACE
4
DAVID A. PATTERSON
JOHN L. HENNESSY

and some from



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

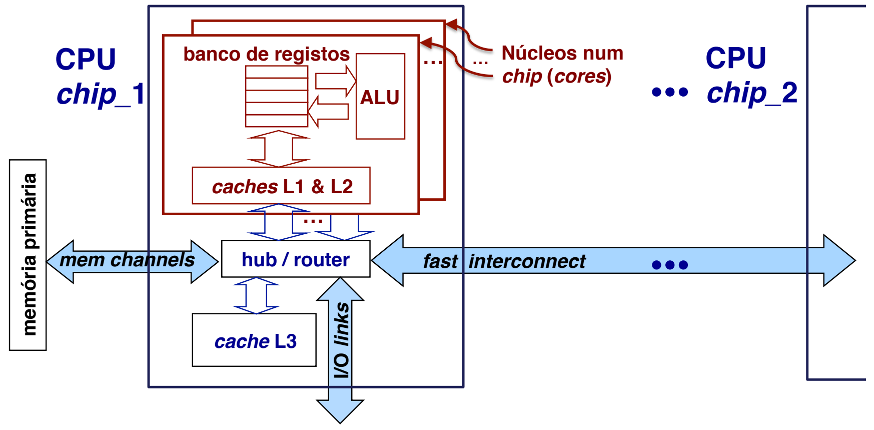
Background for Advanced Architectures

Key concepts to revise:

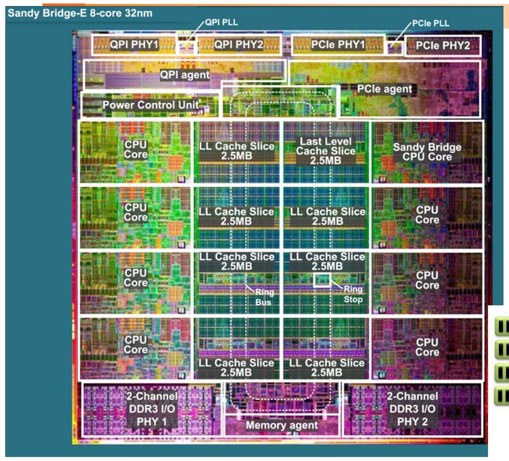
- *numerical data representation (for error analysis)*
- *ISA (Instruction Set Architecture)*
- *how C compilers generate code (a look into assembly code)*
  - *how scalar and structured data are allocated*
  - *how control structures are implemented*
  - *how to call/return from function/procedures*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

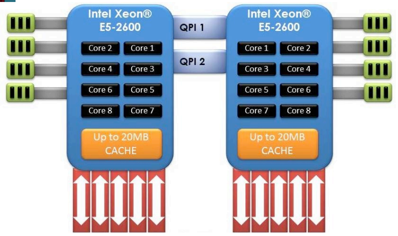
A hierarquia de cache em arquiteturas multicore

As arquiteturas *multicore* mais recentes:

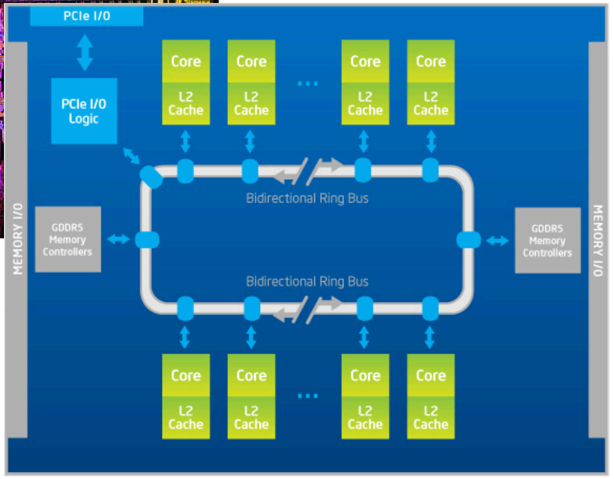


Lançamento da Intel em 2012: Sandy/Ivy Bridge (8-core)

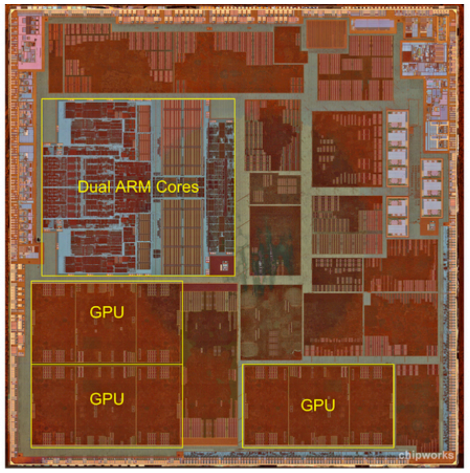




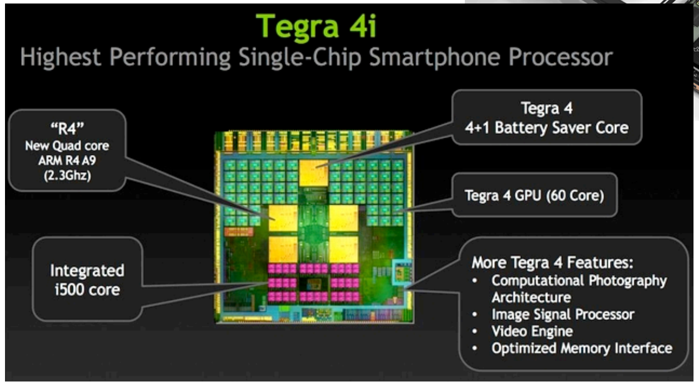
Chips da Intel em 2012/13: Xeon Phi com 60 cores



Exemplo de chip com processadores RISC: 2x ARM's no A6 do iPhone 5



Exemplo de chip com processadores RISC: 4+1 ARM's no Tegra 4i da NVidia


Tegra 4i
Highest Performing Single-Chip Smartphone Processor
"R4" New Quad core ARM R4 A9 (2.3Ghz)
Tegra 4 4+1 Battery Saver Core
Tegra 4 GPU (60 Core)
Integrated i500 core
More Tegra 4 Features:
• Computational Photography Architecture
• Image Signal Processor
• Video Engine
• Optimized Memory Interface

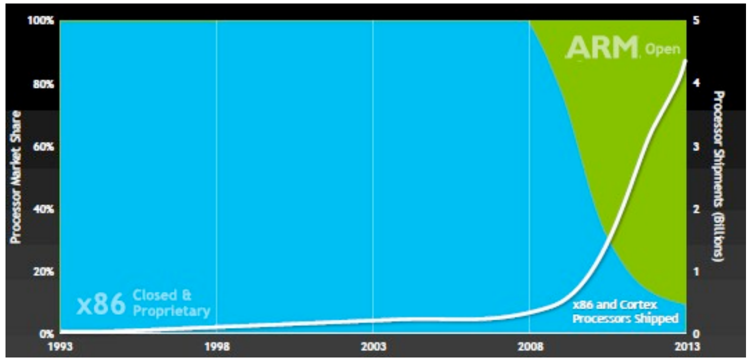
Exemplo de chip com processadores RISC: 4+4 ARM's no Exynos 5 Octa, Galaxy S 4

Performance and Energy-Efficiency

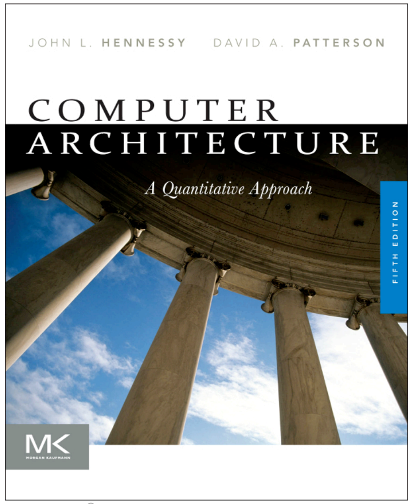
LITTLE — Most energy-efficient processor from ARM — Cortex-A7
- Simple, In-order, 8 stage pipeline
- Performance better than today's mainstream, high-volume smartphones

big — Highest performance in mobile power envelope — Cortex-A15
- Complex, out-of-order, multi-issue pipeline
- Up to 5x the performance of today's mainstream, high-volume smartphones

Processadores Intel x86 versus ARM



Key textbook for AA

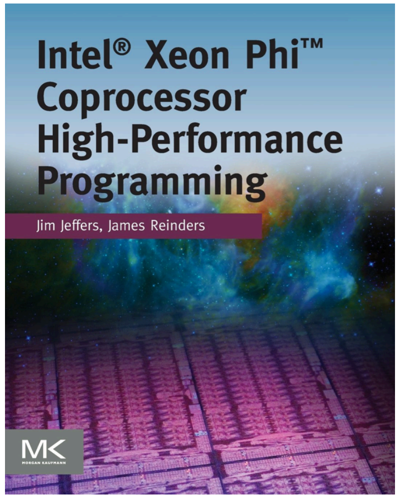


Computer Architecture, 5th Edition

Hennessy & Patterson

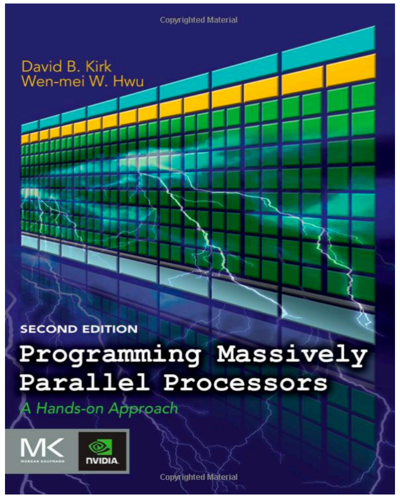
Table of Contents

Recommended textbook (1)


Intel® Xeon Phi™ Coprocessor High-Performance Programming
Jim Jeffers, James Reinders
MK

Contents

Recommended textbook (2)



Contents

Understanding Performance

- 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…

CPU Time *(single-core)*

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

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

Instruction Count and CPI

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

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

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

- 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

Performance Summary *(single-core)*

The BIG Picture

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

- 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, ...

Pipeline Summary

The BIG Picture

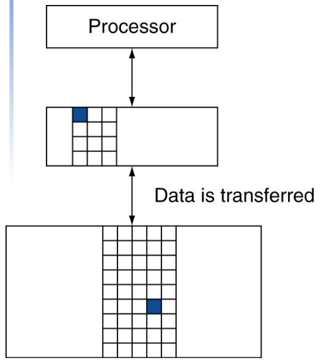
- Pipelining improves performance by increasing instruction throughput
  - Executes multiple instructions in parallel
  - Each instruction has the same latency
- Subject to hazards
  - Structure, data, control
- Instruction set design affects complexity of pipeline implementation

Does Multiple Issue Work?

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 Memory Hierarchy

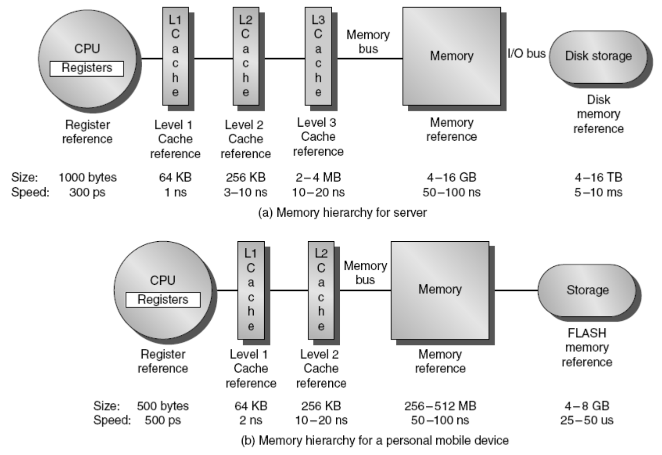
The BIG Picture

- 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

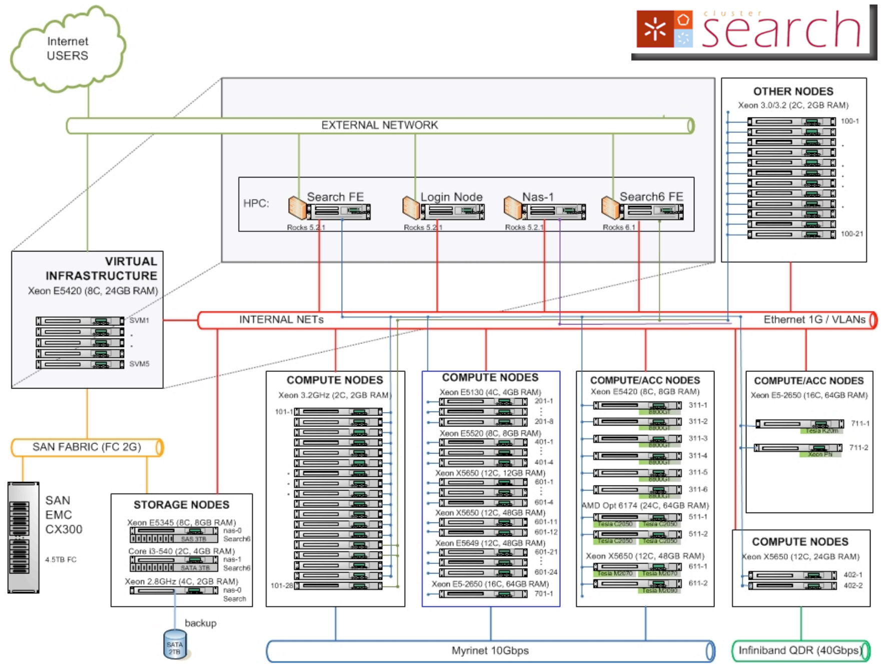
Multilevel Caches

- 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



Copyright © 2012, Elsevier Inc. All rights reserved.



Homework

- Identify all Intel Xeon processors' microarchitecture from Core till the latest releases, and build a table with:
  - year, max clock frequency, # pipeline stages, degree of superscalarity, # simultaneous threads, vector support , # cores, type/bandwidth of external interfaces, …
  - UMA/NUMA; for each cache level: size, latency, line size, direct/associative, bandwidth to access lower memory hierarchy levels, … *(homework for following week)*
- Identify the CPU generations at the SeARCH cluster
- ***Suggestion**: create a GoogleDocs table, shared by all students, and all critically contribute to build the table*