



MSc Informatics Eng.

2012/13

A.J.Proença

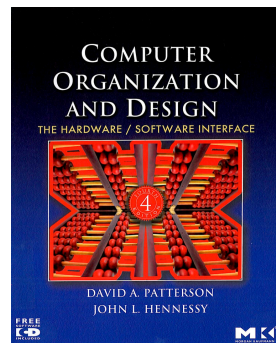
Concepts from undergrad Computer Systems (1)

(most slides are borrowed)



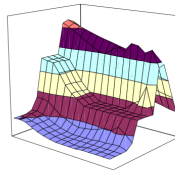
Concepts from undergrad Computer Systems

– most slides are borrowed from



and some from

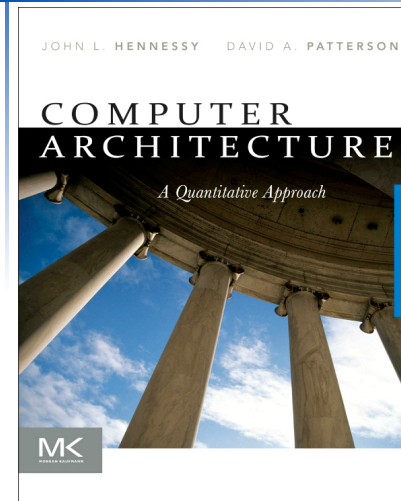
Computer Systems
A Programmer's Perspective¹
(Beta Draft)



Randal E. Bryant
David R. O'Hallaron

August 1, 2001

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



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



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, superscalarity, multi-threading, vector processing, ...
 - memory hierarchy: cache levels, ...

Understanding Performance

- Algorithm
 - Determines number of operations executed
- 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

$$\begin{aligned} \text{CPU Time} &= \text{CPU Clock Cycles} \times \text{Clock Cycle Time} \\ &= \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}} \end{aligned}$$

- 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

$$\begin{aligned} \text{Clock Cycles} &= \text{Instruction Count} \times \text{Cycles per Instruction} \\ \text{CPU Time} &= \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time} \\ &= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}} \end{aligned}$$

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



Performance Summary

The BIG Picture

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

- 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



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



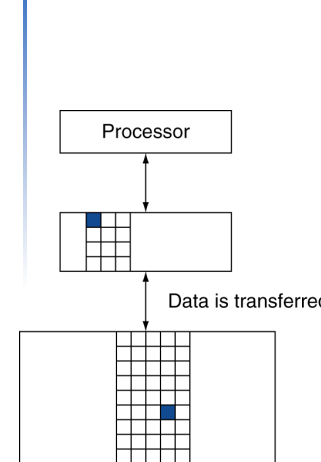
Does Multiple Issue Work?

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 = $1 - \text{hit ratio}$
 - Then accessed data supplied from upper level



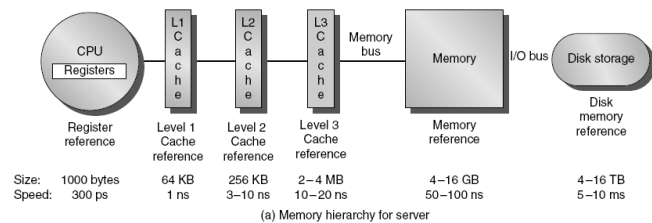
The Memory Hierarchy

The BIG Picture

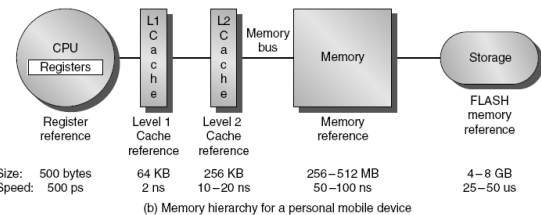
- Common principles apply at all levels of the memory hierarchy
 - Based on notions of caching
- At each level in the hierarchy
 - Block placement
 - Finding a block
 - Replacement on a miss
 - Write policy



Memory Hierarchy



(a) Memory hierarchy for server

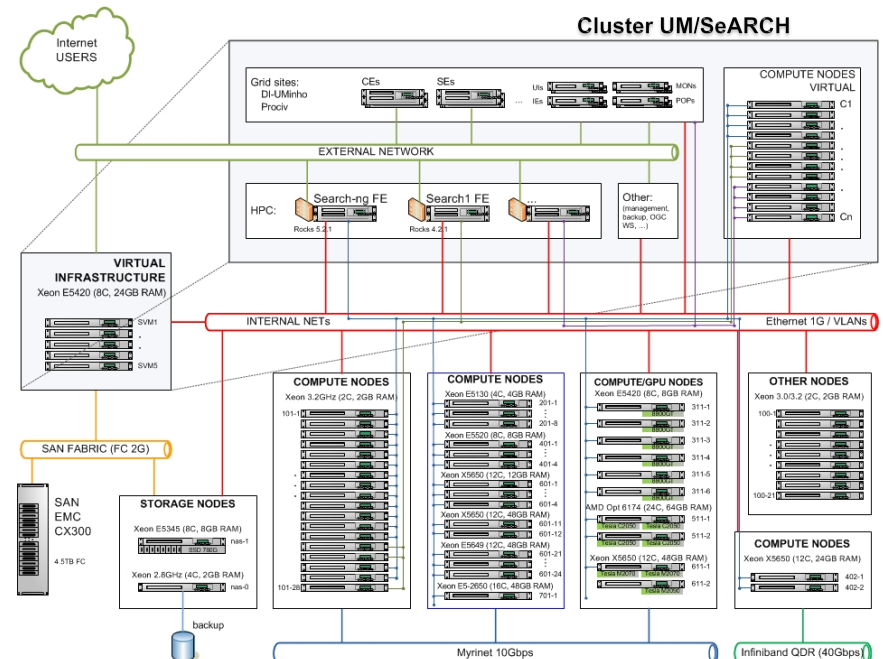


(b) Memory hierarchy for a personal mobile device



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
- Main memory services L-2 cache misses
- Some high-end systems include L-3 cache



Homework



- To identify all AMD and Intel processors' microarchitecture from Hammer and 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*)
- To 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