Computer Architecture
A Quantitative Approach, Fifth Edition


## Chapter 1

## Fundamentals of Quantitative Design and Analysis

*These slides are based on the slides provided by the publisher.*
*The slides will be modified, annotated, explained on the board, and sometimes corrected in the class*

 1

## EECS4201 Computer Architecture

- Instructor
  - Mokhtar Aboelaze
  - Office LAS2026 Phone ext: 40607
- Research interests
  - Computer Architecture
  - Low power architecture
  - Embedded systems
  - FPGA (in embedded applications)

 2

## EECS4201 Computer Architecture

- Text
  - Computer Architecture: A Quantitative Approach Patterson & Hennessey 5th Ed.
- Class Meeting
  - Tuesdays, Thursdays 10:00-11:30 CB122
- Office Hours
  - Tuesdays, Thursdays 1:00-3:00pm or by appointment

 3

## EECS4201 Topics

- Introduction
- Instruction level parallelism
- Data level parallelism (SIMD and GPU)
- Thread level parallelism
- Memory hierarchy design
- Introduction to warehouse-scale computers
- SOC and MPSOC *if time permits*

 4

## Course Learning Outcomes

- After successful completion of the course, students are expected to be able to:
- Design cache, memory hierarchy, and virtual memory using different techniques to improve cost/performance ratio.
- Demonstrate how dynamic scheduling and speculative execution can improve the system performance and explain how it is implemented in modern processors.
- Evaluate different design alternatives and make quantitative/qualitative argument for one design over the other.
- Identity the different types of parallelism (data, instruction, thread, transaction) for a given application.
- Compare and evaluate different techniques (such as multithreading, multicore, or vector) to improve CPU performance

 5

## Grading EECS4201

- Grades are distributed as follows
  - HW/Assignments **10%**
  - Quizzes **15%**
  - Midterm **25%**
  - Paper review – groups of 2 **10%**
  - Final **40%**

 6

## Grading EECS5501

- Grades are distributed as follows
  - HW/Assignments **10%**
  - Quizzes **15%**
  - Midterm **20%**
  - Project **20%**
  - Final **35%**

 7

## Assumptions

- I assume that you already completed EECS2021 or equivalent (you know about these topics).
  - Assembly language
  - RISC architecture
  - ALU architecture
  - Pipelining and hazards
  - Memory hierarchy and cache organization !?

 8

## Computer Architecture

- Why study computer architecture
- Hardware/Architecture
  - Design better, faster, cheaper computers that use as little energy as possible
- Software
  - Understand the architecture to squeeze as much performance for your code as possible

 9

## Computer Technology

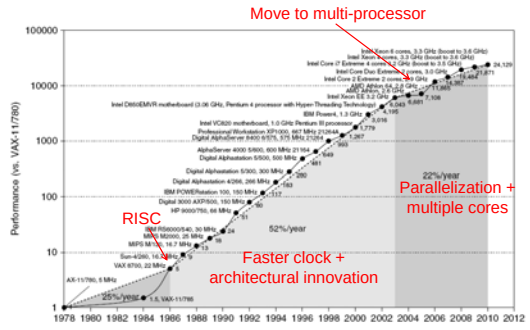


- Performance improvements:
  - Improvements in semiconductor technology
    - Feature size, clock speed
  - Improvements in computer architectures
    - Enabled by HLL compilers, UNIX
    - Lead to RISC architectures
  - Together have enabled:
    - Lightweight computers
    - Cheap computers




## Single Processor Performance



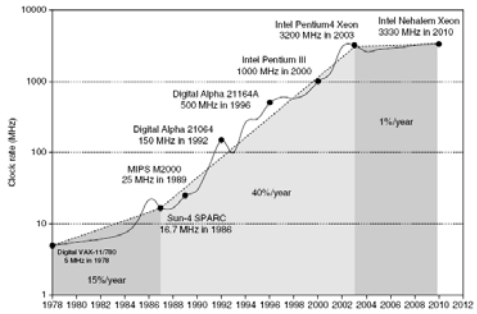






## Clock Speed





## Current Trends in Architecture



- Cannot continue to leverage Instruction-Level parallelism (ILP)
  - Single processor performance improvement ended around 2003
- New models for performance:
  - Data-level parallelism (DLP)
  - Thread-level parallelism (TLP)
  - Request-level parallelism (RLP)
- These require explicit restructuring of the application

## Classes of Computers



- Personal Mobile Device (PMD)
  - e.g. smart phones, tablet computers
  - Emphasis on energy efficiency and real-time
- Desktop Computing
  - Emphasis on price-performance
- Servers
  - Emphasis on availability, scalability, throughput
- Clusters / Warehouse Scale Computers
  - Used for “Software as a Service (SaaS)”
  - Emphasis on availability and price-performance
  - Sub-class: Supercomputers, emphasis: floating-point performance and fast internal networks
- Embedded Computers
  - Emphasis: price

## Parallelism



- Classes of parallelism in applications:
  - Data-Level Parallelism (DLP)
  - Task-Level Parallelism (TLP)
- Classes of architectural parallelism:
  - Instruction-Level Parallelism (ILP)
  - Vector architectures/Graphic Processor Units (GPUs)
  - Thread-Level Parallelism – Highly coupled
  - Request-Level Parallelism – Decoupled

## Flynn's Taxonomy



- Single instruction stream, single data stream (SISD)
- Single instruction stream, multiple data streams (SIMD)
  - Vector architectures
  - Multimedia extensions
  - Graphics processor units
- Multiple instruction streams, single data stream (MISD)
  - No commercial implementation
- Multiple instruction streams, multiple data streams (MIMD)
  - Tightly-coupled MIMD
  - Loosely-coupled MIMD

## Defining Computer Architecture



- "Old" view of computer architecture:
  - Instruction Set Architecture (ISA) design
  - i.e. decisions regarding:
    - registers, memory addressing, addressing modes, instruction operands, available operations, control flow instructions, instruction encoding
- "Real" computer architecture:
  - Specific requirements of the target machine
  - Design to maximize performance within constraints: cost, power, and availability
  - Includes ISA, microarchitecture, hardware

## Trends in Technology



- Integrated circuit technology
  - Transistor density: 35%/year
  - Die size: 10-20%/year
  - Integration overall: 40-55%/year
- DRAM capacity: 25-40%/year (slowing)
- Flash capacity: 50-60%/year
  - 15-20X cheaper/bit than DRAM
- Magnetic disk technology: 40%/year
  - 15-25X cheaper/bit then Flash
  - 300-500X cheaper/bit than DRAM

## Bandwidth and Latency

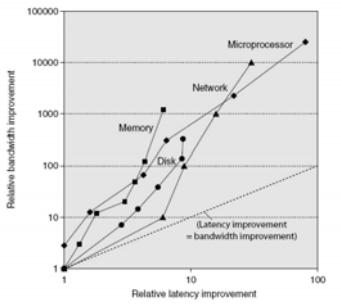


- Bandwidth or throughput
  - Total work done in a given time
  - 10,000-25,000X improvement for processors
  - 300-1200X improvement for memory and disks
- Latency or response time
  - Time between start and completion of an event
  - 30-80X improvement for processors
  - 6-8X improvement for memory and disks




## Bandwidth and Latency






Log-log plot of bandwidth and latency milestones