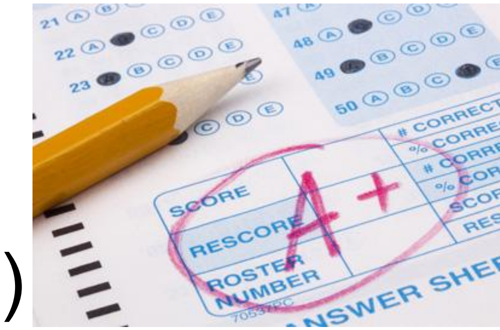
CS2507 Computer Architecture

Dr. Ahmed H. Zahran
WGB 182
a.zahran@cs.ucc.ie

Course Organization

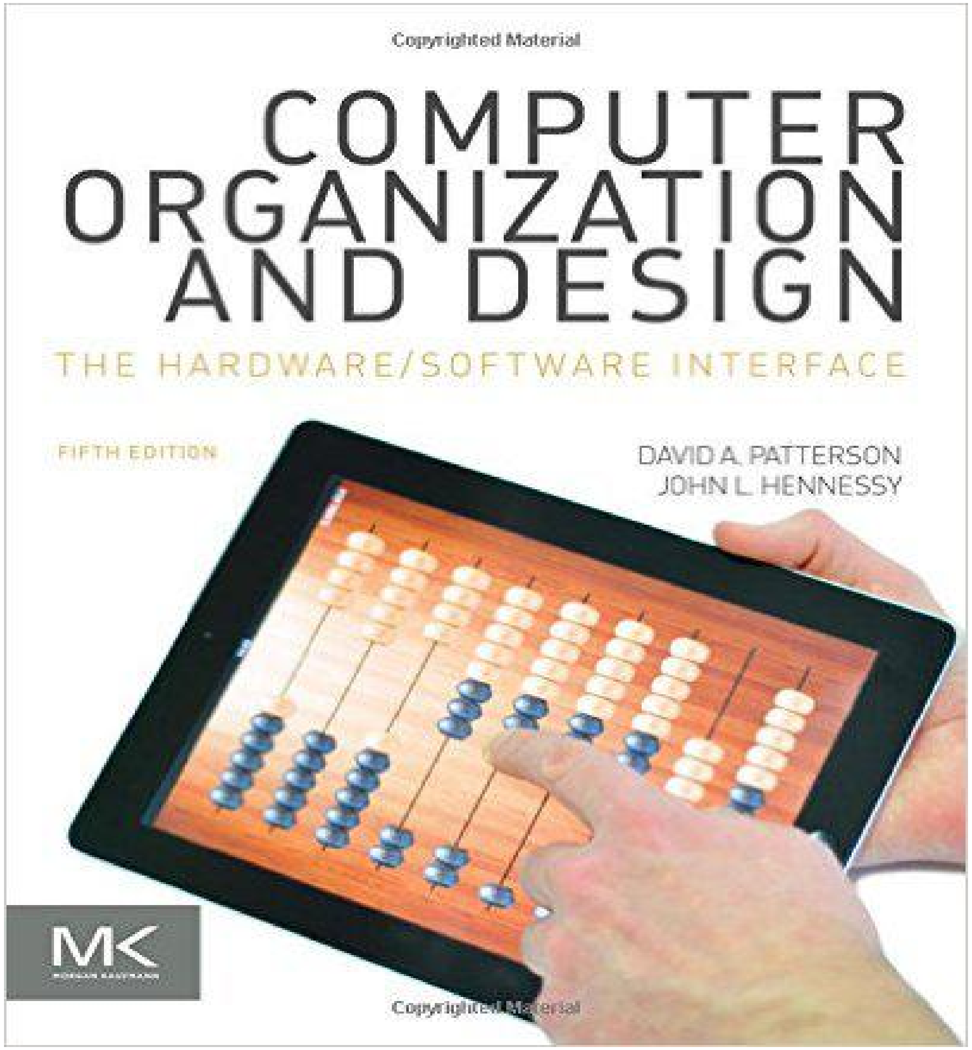
- Course webpage @ CANVAS
- Lectures
  - Tuesdays 13:00-14:00 [Online] [MS Teams]
  - Wednesdays 13:00-14:00 [Online][MS Teams]
- Office hours
  - Wednesdays 14:30 -15:30 or by appointment [MS Teams]
- Labs
  - Will be advertised later
  - 1 hours per week x 10 (Starting Week 2)

Course Evaluation



- Assignments (20 %)
  - 5 assignments (Equal weights)
- Final examination (80 %)
- Bonuses
  - Assignment and participation
- Pass mark 40 %

Textbook



- Lecture slides
  - Will be posted on CANVAS
  - Include reading material

What CS2507 offers

What determines computer performance?

How programs are translated into the machine language

hardware/software interface

Assembly Instruction

floating numbers representation

Processor Internals and Design

Hands on Assembly

Memory Design and performance

How this can benefit me?

- Write software for performance!
- Improve troubleshooting skills
- Be more appealing to many companies

Computer Architecture

Sections 1.1 - 1.4
Section 1.5 (optional)

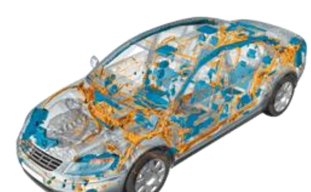
6

What computer?

- Personal computer (general purpose)

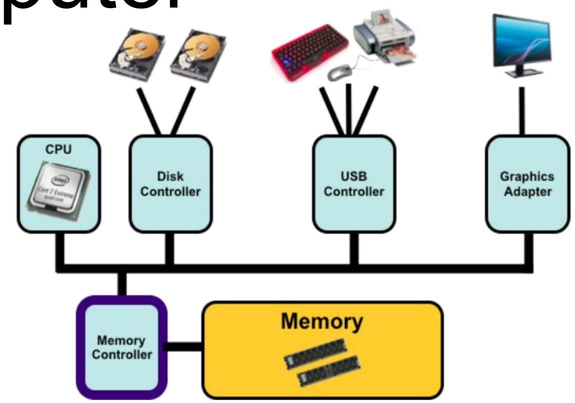
- Servers (simple to data centers)

- Embedded computers



Computer Components

- Same components for all computer
  - – Inputs/Outputs
  - – Memory/storage
  - – Processor

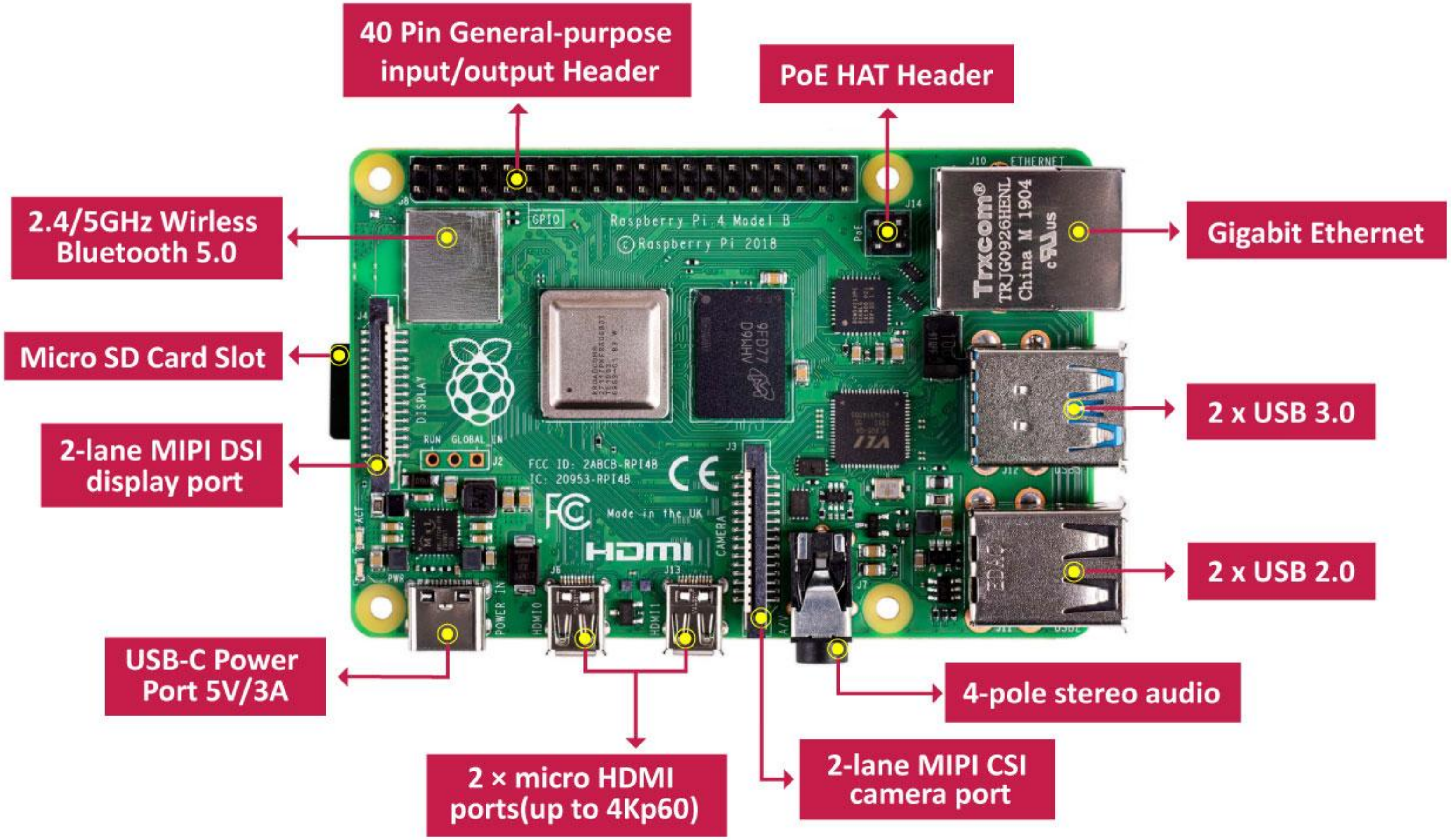


- These components would have distinct physical and logical implementations
  - the HW choice depends on many factors such as usage, cost, and energy efficiency

Openning the box

USB
ports

Openning the box

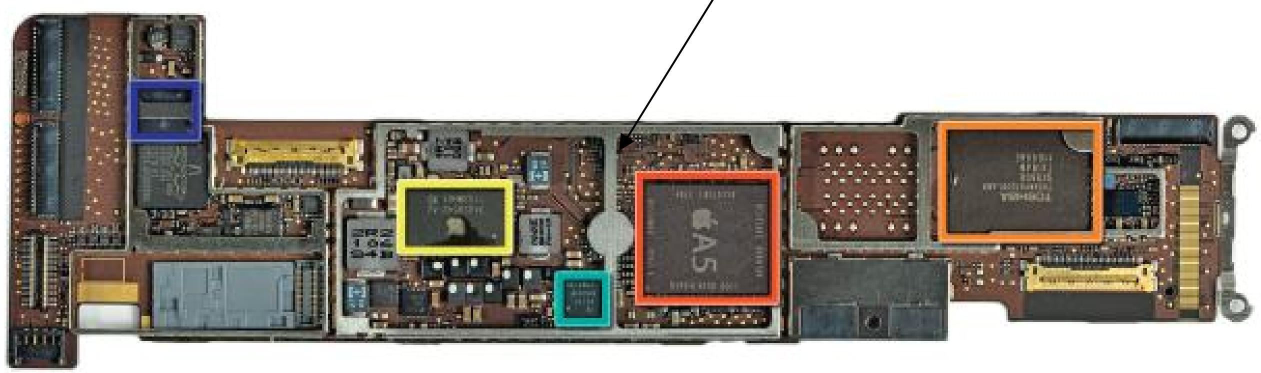


Opening the Box

Capacitive multitouch LCD screen

3.8 V, 25 Watt-hour battery

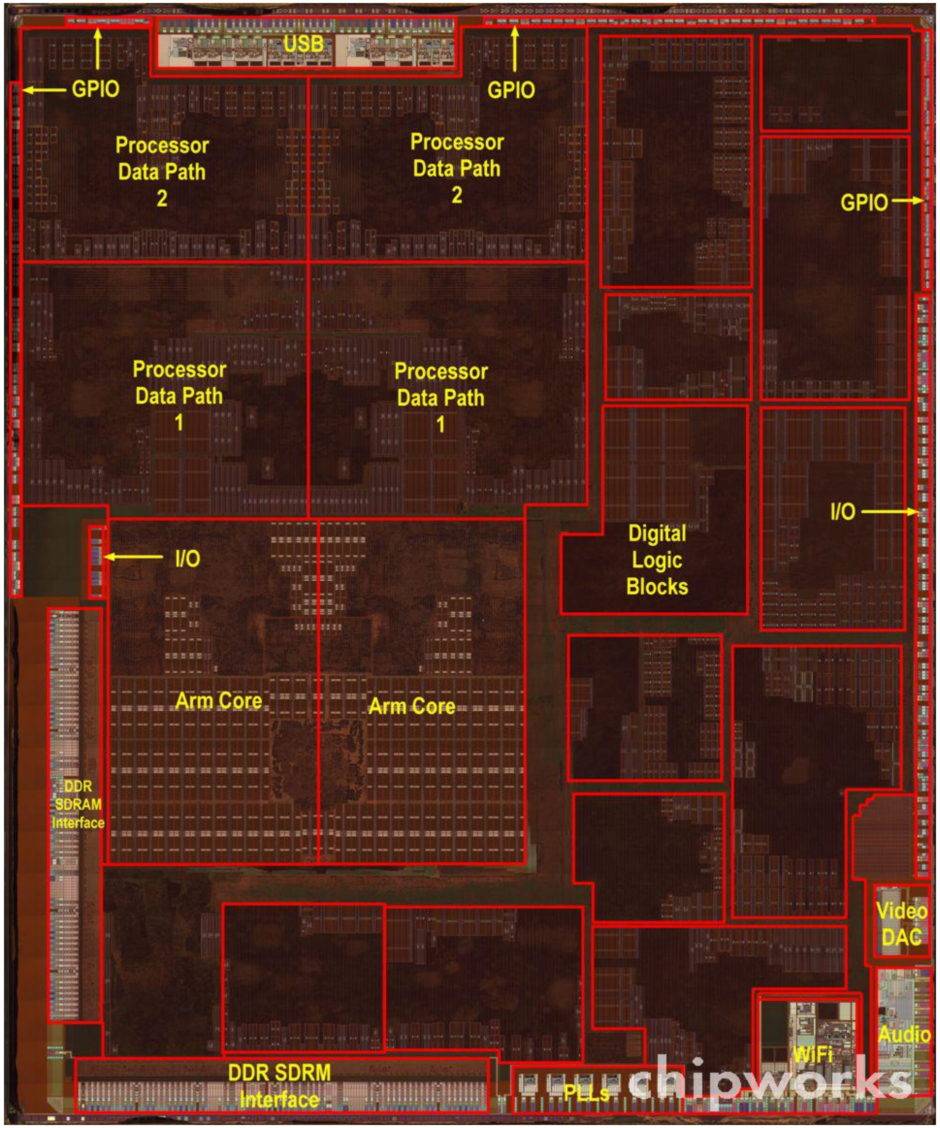
Computer board



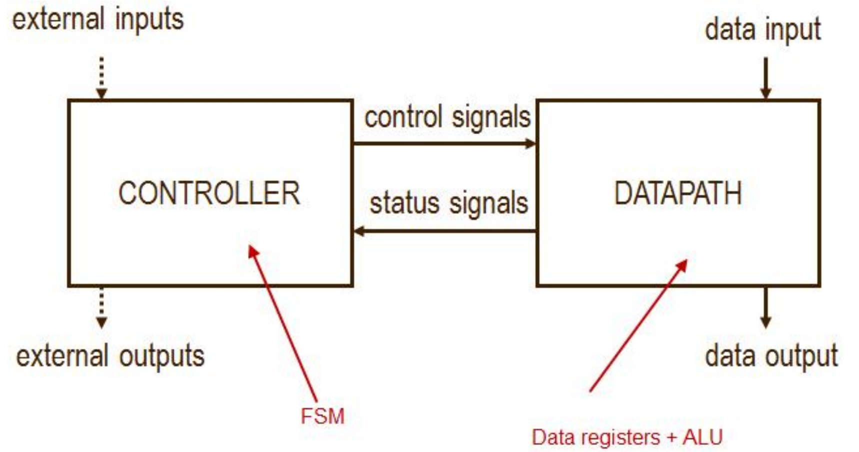
Apple iPad 2 tablet

Microprocessor Package

- Apple A5



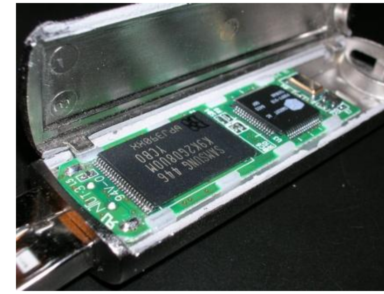
Inside the Processor

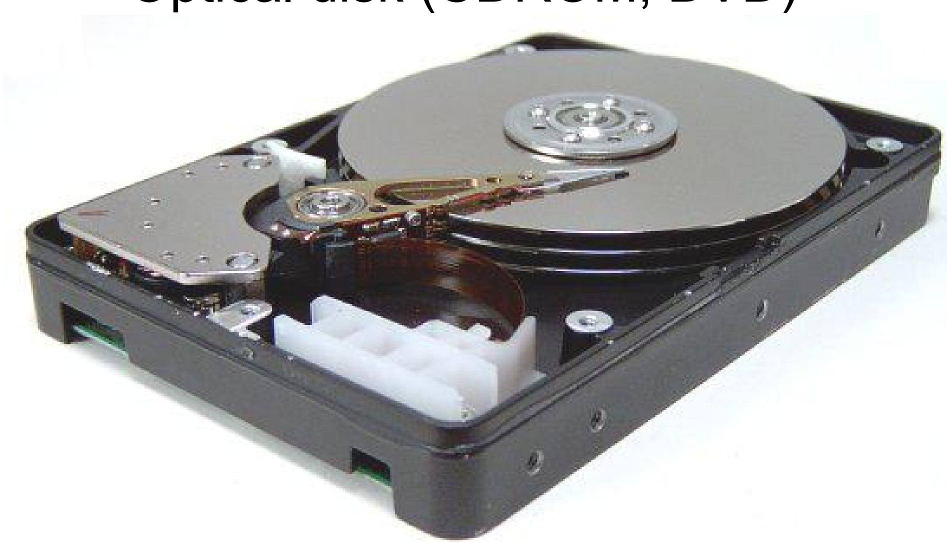


- ***Datapath***:
  - performs operations on data
- ***Controller***
  - sequences datapath, memory access
- ***Cache memory***
  - Small fast memory for immediate access to data

Memory

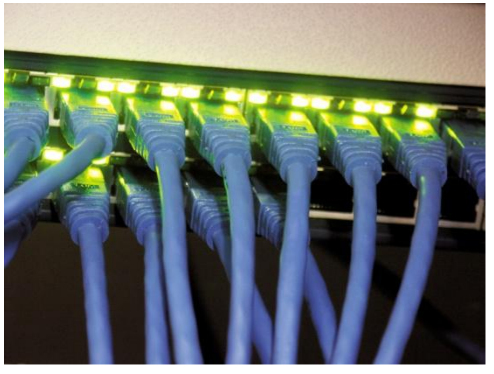
- ***Volatile*** main memory
  - Loses instructions and data when power off
  - RAM and cache
- ***Non-volatile*** secondary memory
  - Magnetic disk
  - Flash memory
  - Optical disk (CDROM, DVD)





Networks (I/O example)

- Communication
  - Ethernet, WiFi, Bluetooth



- Resource sharing (cloud computing, printers, …)
- Nonlocal access (mobile computing)

Computer Architecture

- **DEF:** *Computer architecture is the science and art of designing hardware components to create computers that meet functional, performance and cost goals*

Technology
Circuit, packaging, memory, …

Domains
PMD, server, game consoles, …

Design Goals
Performance, cost, energy efficiency, reliability, time-to-market

Eight Great Ideas

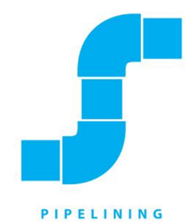
Computer Architecture: Eight Great Ideas

1. Design for ***Moore's Law***
   – Design for rapid change

2. Use ***abstraction*** to simplify design
   – Representing hardware and software a different levels

3. Make the ***common case fast***
   – Easier to improve on simple cases than complex ones

4. Performance *via* ***pipelining***
   – Sequential pattern of parallelism



Computer Architecture: Eight Great Ideas

5. *Performance via* ***parallelism***
   - *Parallel operations are faster*
6. ***Hierarchy*** of memories
   - Arranging memory according to cost/fastness
7. Performance *via* ***prediction***
   - Operating based on healthy guess
8. ***Dependability*** *via* redundancy
   - Including redundant components for addressing failure

PARALLELISM

HIERARCHY

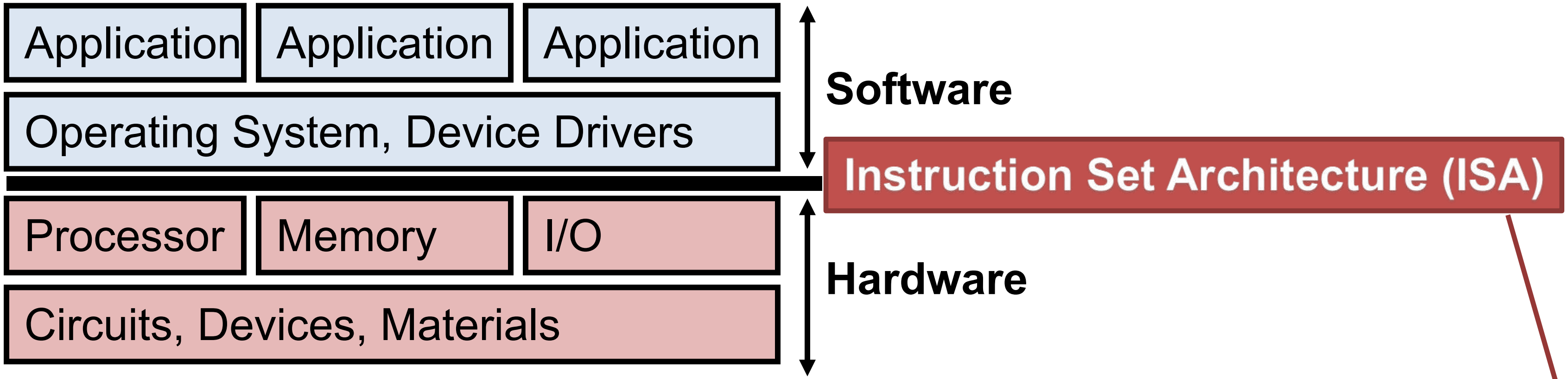
PREDICTION

DEPENDABILITY

Computer Abstraction

Software
Hardware

Computer Abstraction

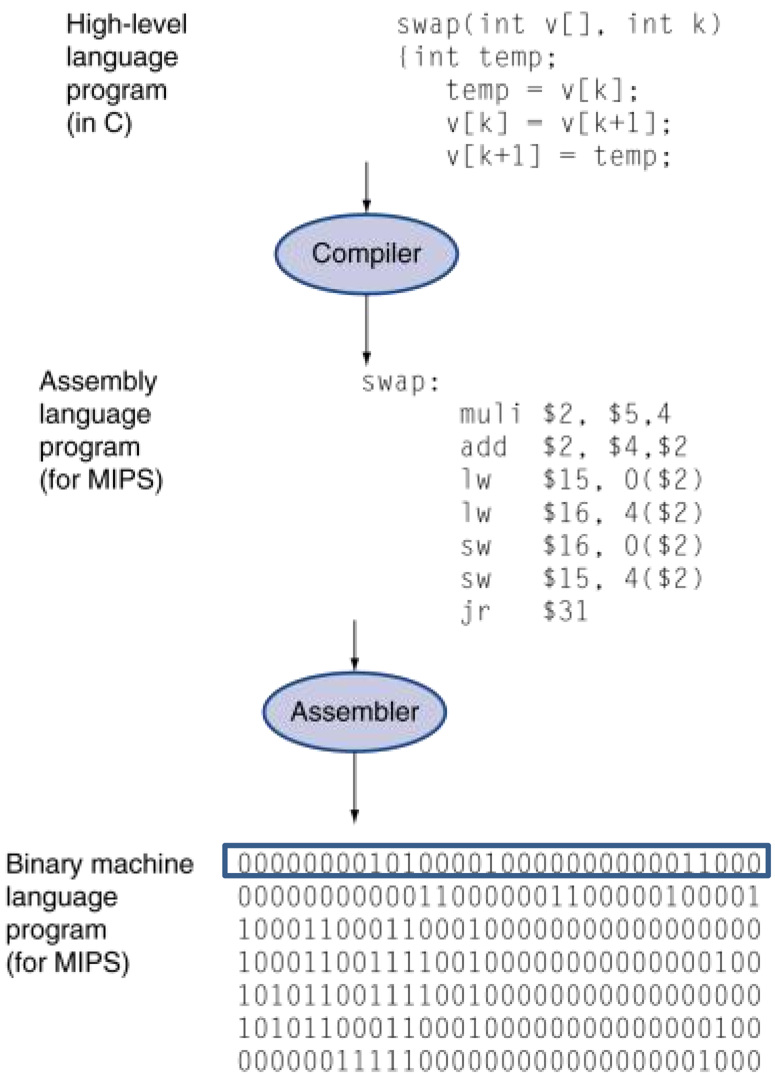


Abstraction: only way of dealing with complex systems

- Divide world into objects, each with an…
  - **Interface**: knob(s)
  - **Implementation**: "black box"
- Only specialists deal with implementation, rest of us with interface

- *The instruction set architecture is the key interface between the hardware and low-level software*
- *enables many implementations of varying cost and performance*

Levels of Program Code

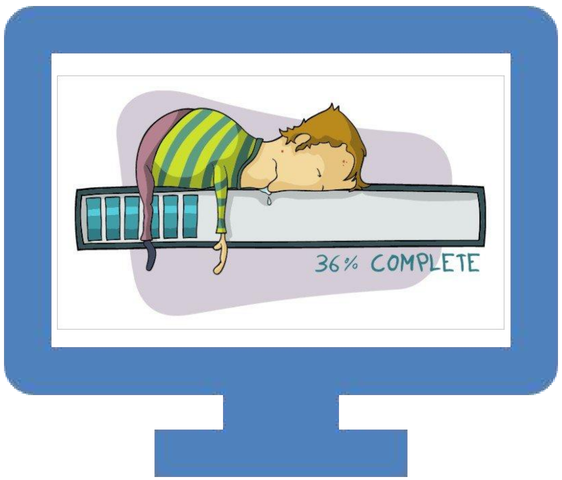


- ***High-level language***
  - Level of abstraction closer to problem domain
  - Provides for productivity and portability
- ***Assembly language***
  - Textual representation of instructions
- ***Machine language***
  - Binary digits (bits)
  - Encoded instructions and data

Summary

- Different computers share a common set of components: processor, memory, and I/O
- Eight design ideas have contributed to the improvement in computer performance over years
- Abstraction is an intrinsic principal in hardware and software design
- The instruction set architecture is the key interface between the hardware and low-level software

Computer Performance



Sections 1.6 - 1.9

Objectives

- Define ***key metrics*** used for measuring computer performance
- Identify ***factors*** affecting computer performance
- Explain ***approaches*** of boosting computer performance and their ***challenges***

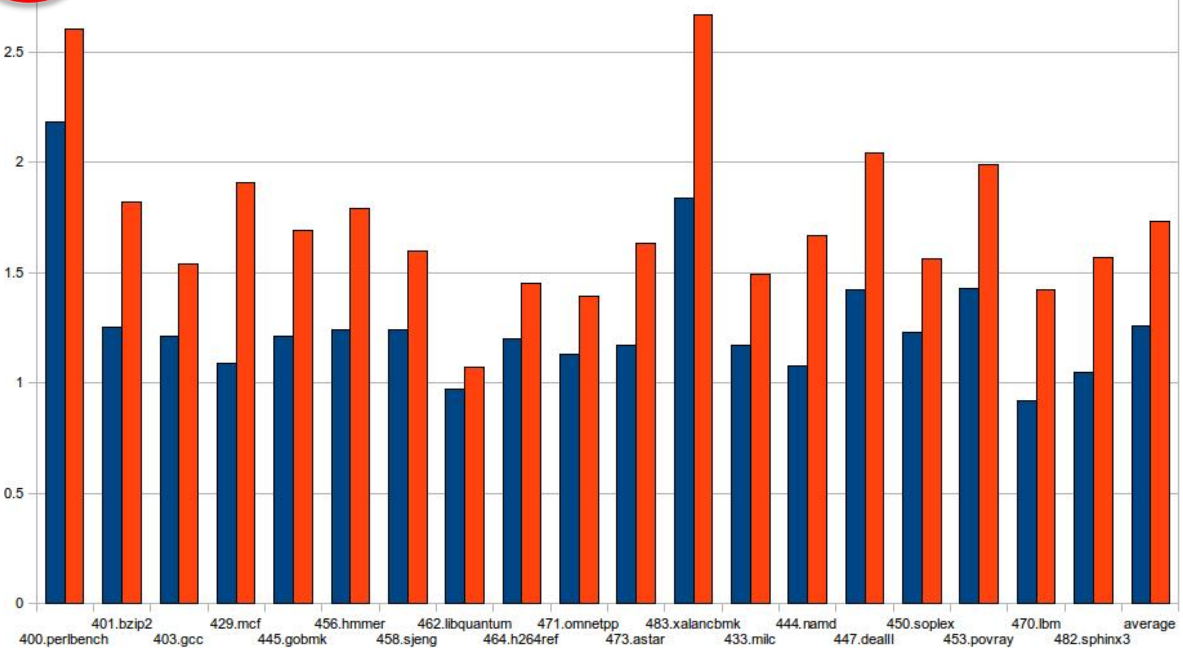
Key Performance Metrics

- **When we say computer A is better than Computer B?**
- ***Response time***
  - Also referred to as execution time
  - How long does it take to complete a task?
- ***Throughput***
  - Total work done per unit time
    - e.g., tasks/transactions/… per hour
  - *relevant to servers*

CPU Benchmark

- Benchmarks are programs specifically chosen to measure performance
- SPEC CPU2006(12 integer + 17 floating point)

A ratio normalized to a reference machine

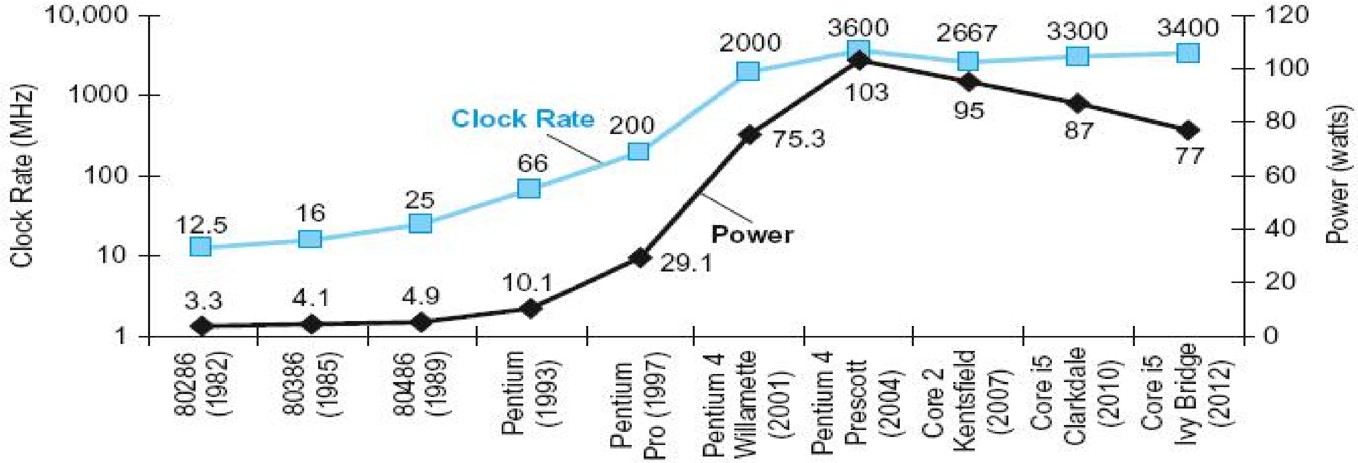


$$\sqrt[n]{\prod_{i=1}^{n} \text{Execution time ratio}_i}$$

Discussion

- How are response time and throughput affected by
  1. Replacing the processor with a faster version?
  2. Adding more processors?

Intel processors



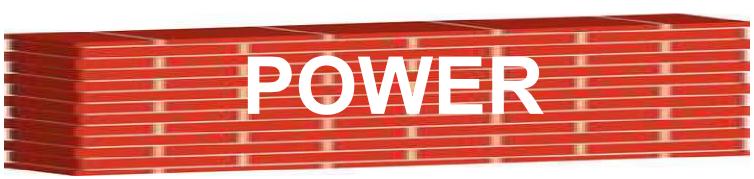
- In CMOS IC technology
  - Complementary Metal Oxide Semiconductor (CMOS)

$$\text{Power} = \text{Capacitive load} \times \text{Voltage}^2 \times \text{Frequency}$$

$\times 30$

$5\text{V} \rightarrow 1\text{V}$

$\times 1000$



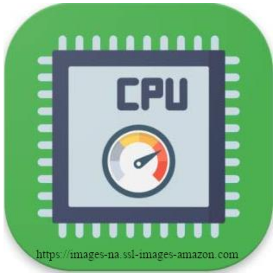
Power wall refers to the high leakage current that accompany increasing the CPU frequency without reducing the voltage level. Expensive cooling systems are needed to compensate for temperature increase.

Multiprocessors

- Multicore microprocessors
  - More than one processor per chip
  - Can improves the system throughput

- Reaping the benefits of multiple cores requires explicitly ***parallel programming***

- Why parallel programming is hard to do?
    - Load balancing
    - Optimizing communication and synchronization



Performance Analysis

Relative Performance

- Define **Performance = 1/Execution Time**
- "X is n time faster than Y"

$$\text{Performance}_X / \text{Performance}_Y$$
$$= \text{Execution time}_Y / \text{Execution time}_X = n$$

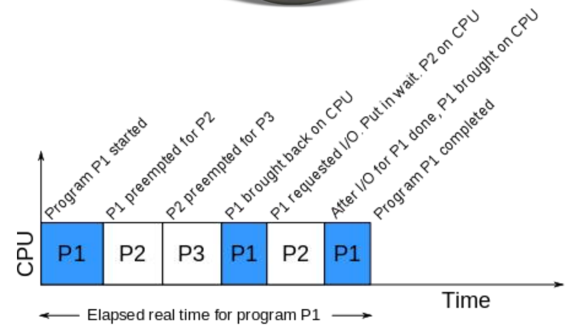
- Example: time taken to run a program
  - 10s on A, 15s on B. How much faster is A than B?

- Execution Time_B / Execution Time_A = 15s / 10s = 1.5
  - So A is 1.5 times faster than B

Measuring Performance

- Elapsed time
  - Total time to complete a task, including all aspects
    - Processing, I/O, OS overhead, idle time
  - Determines ***system performance***

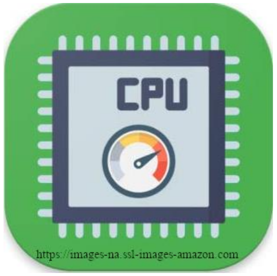
- ***CPU time***
  - Time spent processing a given job
    - Does not include: I/O time, other jobs' shares
  - Comprises user CPU time and system CPU time



Source: Wikipedia

- Different programs are affected differently by CPU and system performance
  - *a good understanding of the used system (HW & OS) can help developing more efficient programs*

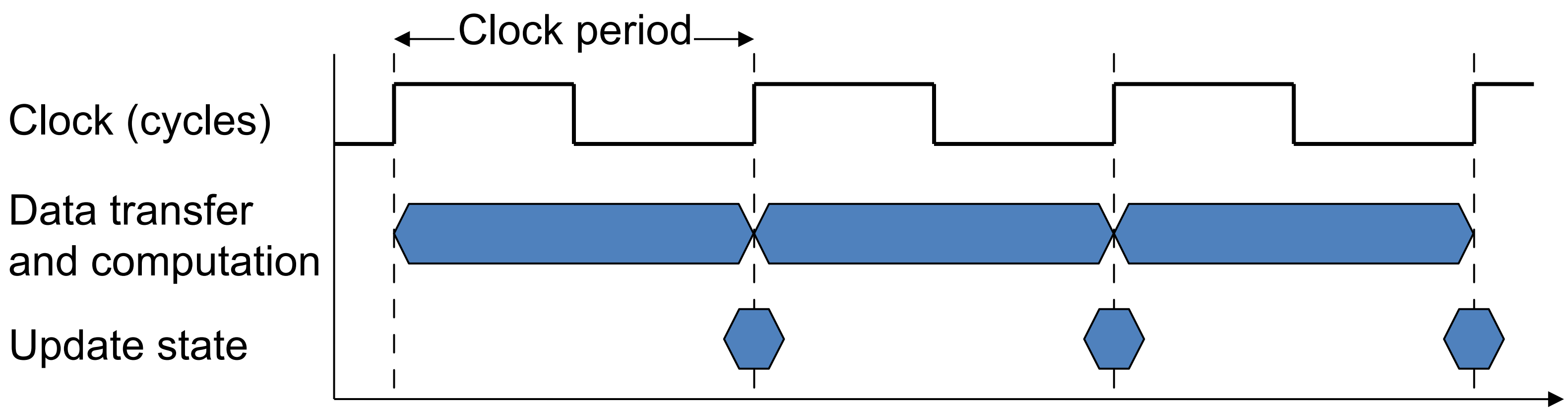
Standard Benchmarking SPEC
Personal



https://images-na.ssl-images-amazon.com

CPU Clocking

- Operation of digital hardware governed by a constant-rate clock



- ***Clock period***: duration of a clock cycle
  - E.g., 250ps = 0.25ns = 250×10^{-12}s
- ***Clock frequency (rate)***: cycles per second
  - E.g., 4.0GHz = 4000MHz = 4.0×10^{9}Hz

CPU Performance

- CPU execution time for a program

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

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

- Performance improved by
  - Reducing number of clock cycles (how?)
  - Increasing clock rate
- Hardware designer often trade off clock rate against cycle count

CPU Time Example

- Computer A: 2GHz clock, 10s CPU time
- Designing Computer B
  - Aim for 6s CPU time
  - Can do faster clock, but causes 1.2 × clock cycles
- How fast must Computer B clock rate be?

Instruction **Count (IC) & Cycles Per** ***Instruction*** **(CPI)**

An instruction usually refers to an atomic operation such as ALU calculation, memory access, I/O access

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

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

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

- Instruction Count 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***

Instructions
50% ALU
45% MEM
5% I/O

CPI Example

- **Computer A:** Cycle Time = 250ps, CPI = 2.0
- **Computer B:** Cycle Time = 500ps, CPI = 1.2
  - Same ISA
  - Which is faster, and by how much?

Performance

Time is the only complete and reliable measure of performance

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

- 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