# Course Outline

## I. Introduction

## II. Performance Evaluation

## III. Processor Design and Analysis

- I. Single-cycle implementation
- II. Multi-cycle implementation
- III. Pipelined implementation
- IV. Hazards
- V. Forwarding

## IV. Memory Design and Analysis

- I. Cache block size and associativity
- II. Virtual memory

## V. I/O Design and Analysis

- I. Parity
- II. ECC
- III. Scheduling

## VI. Basic OS Functions
Introduction

This course is all about how computers work

But what do we mean by a computer?
- Different types: desktop, servers, embedded devices
- Different uses: automobiles, graphics, structural analysis, finance, genomics…
- Different manufacturers: Intel, Apple, IBM, Microsoft, Sun…
- Different underlying technologies and different costs!

Microprocessor market development ($US Million) by Supplier – 2003-2014

A Logarithmic scale is used in this chart

Source: ITCandor, 2014

Microprocessor Revenue ($US Billion) Forecast by type and quarter – 2003-2019

Revenue Growth

Source: ITCandor, 2014
Variety of Architectures

Microprocessor market development ($US Million) by Supplier – 2003-2014

A Logarithmic scale is used in this chart

Source: ITCandor, 2014
Death of the PC?

Microprocessor Revenue ($US Billion) Forecast by type and quarter – 2003-2019

Source: ITCandor, 2014
Analogy: Consider a course on “automotive vehicles”

- Many similarities from vehicle to vehicle (e.g., wheels)
- Huge differences from vehicle to vehicle (e.g., gas vs. electric)

Best way to learn:

- Focus on a specific instance and learn how it works
- While learning general principles and historical perspectives
Why learn this stuff?

- You want to call yourself a “computer scientist”
- You want to build software people use (need performance)
- You need to make a purchasing decision or offer “expert” advice

Debugging skills often benefit from understanding architecture
- better understand system error messages
- better understand translators (compilers and interpreters)

Both hardware and software affect performance:
- Algorithm determines number of source-level statements
- Language/Compiler/Architecture determine machine instructions (Chapters 2 and 3)
- Processor/Memory determine how fast instructions are executed (Chapters 5, 6, and 7)

Assessing and Understanding Performance in Chapter 4
Focus: the Processor

Overview

Control
Instruction cache
Enhanced floating point and multimedia
Advanced pipelining hyperthreading support
Control
Control
Data cache
Integer datapath
Secondary cache and memory interface

I/O Interface

Memory
Processor
Processor interface
Disk and USB interfaces
Graphics
I/O bus slots

Computer Organization II
©2005-2015 McQuain
Focus: the Memory Hierarchy

- **Speed**
  - Fastest: Memory (Smallest, Highest, SRAM)
  - Slowest: Memory (Biggest, Lowest, Magnetic disk)

- **Cost ($/bit)**
  - Fastest: Highest
  - Slowest: Lowest

- **Current technology**
  - Fastest: SRAM
  - Slowest: Magnetic disk
Focus: I/O

Overview

- Processor
- Cache
- Memory-I/O Interconnect
  - Main memory
  - I/O controller
    - Disk
  - I/O controller
    - Graphics output
  - I/O controller
    - Network
Focus: Connections to the OS

- Process management
- Virtual memory
- File systems
Language Abstractions

Delving into the depths reveals more information

An abstraction omits unneeded detail, helps us cope with complexity

What are some of the details that appear in these familiar abstractions?

```
swap(int v[], int k) {
    int temp;
    temp = v[k];
    v[k] = v[k+1];
    v[k+1] = temp;
}
```

High-level language (C)

Assembly language (MIPS)

```
swap:
    multi $2, $5, 4
    add  $2, $4, $2
    lw   $15, 0($2)
    lw   $16, 4($2)
    sw   $16, 0($2)
    sw   $16, 4($2)
    jr    $31
```

Binary machine language (for MIPS)

```
00000000101000010000000000011000
000000000011000001100000100001
1000110011001000000000000000000
10001100111111000100000000000100
10101100111110001000000000000000
10101100110001000000000000001000
00000011111000000000000001000
```
Below Your Program

Application software
   Written in high-level language

System software
   Compiler: translates HLL code to machine code
   Operating System: service code
      Handling input/output
      Managing memory and storage
      Scheduling tasks & sharing resources

Hardware
   Processor, memory, I/O controllers
How do computers work?

Need to understand abstractions such as:

- Applications software
- Systems software
- Assembly Language
- Machine Language
- Architectural Issues: i.e., Caches, Virtual Memory, Pipelining
- Sequential logic, finite state machines
- Combinational logic, arithmetic circuits
- Boolean logic, 1s and 0s
- Transistors used to build logic gates (CMOS)
- Semiconductors/Silicon used to build transistors
- Properties of atoms, electrons, and quantum dynamics

So much to learn!
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1951</td>
<td>UNIVAC I</td>
<td>1,000</td>
<td>125,000</td>
<td>2,000</td>
<td>48</td>
<td>$1,000,000</td>
<td>1</td>
<td>$6,107,600</td>
<td>1</td>
</tr>
<tr>
<td>1964</td>
<td>IBM S/360 model 50</td>
<td>60</td>
<td>10,000</td>
<td>500,000</td>
<td>64</td>
<td>$1,000,000</td>
<td>263</td>
<td>$4,792,300</td>
<td>318</td>
</tr>
<tr>
<td>1965</td>
<td>PDP-8</td>
<td>8</td>
<td>500</td>
<td>330,000</td>
<td>4</td>
<td>$16,000</td>
<td>10,855</td>
<td>$75,390</td>
<td>13,135</td>
</tr>
<tr>
<td>1976</td>
<td>Cray-1</td>
<td>58</td>
<td>60,000</td>
<td>166,000,000</td>
<td>32,000</td>
<td>$4,000,000</td>
<td>21,842</td>
<td>$10,756,800</td>
<td>51,604</td>
</tr>
<tr>
<td>1981</td>
<td>IBM PC</td>
<td>1</td>
<td>150</td>
<td>240,000</td>
<td>256</td>
<td>$3,000</td>
<td>42,105</td>
<td>$5,461</td>
<td>154,673</td>
</tr>
<tr>
<td>1991</td>
<td>HP 9000/750</td>
<td>2</td>
<td>500</td>
<td>50,000,000</td>
<td>16,384</td>
<td>$7,400</td>
<td>3,556,188</td>
<td>$9,401</td>
<td>16,122,356</td>
</tr>
<tr>
<td>1996</td>
<td>Intel PPro PC (200 MHz)</td>
<td>2</td>
<td>500</td>
<td>400,000,000</td>
<td>16,384</td>
<td>$4,400</td>
<td>47,846,890</td>
<td>$4,945</td>
<td>239,078,908</td>
</tr>
<tr>
<td>2003</td>
<td>Intel Pentium 4 PC (3.0 GHz)</td>
<td>2</td>
<td>500</td>
<td>6,000,000,000</td>
<td>262,144</td>
<td>$1,600</td>
<td>1,875,000,000</td>
<td>$1,600</td>
<td>11,452,000,000</td>
</tr>
</tbody>
</table>
### Units of Measurement

#### Spatial units:

<table>
<thead>
<tr>
<th>Unit</th>
<th>Traditional</th>
<th>Secondary Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte (B)</td>
<td>8 bits</td>
<td>8 bits</td>
</tr>
<tr>
<td>kilobyte (KB)</td>
<td>1024 or $2^{10}$ bytes</td>
<td>1000 or $10^3$ bytes</td>
</tr>
<tr>
<td>megabyte (MB)</td>
<td>1024 kilobytes or $2^{20}$ bytes</td>
<td>1000 kilobytes or $10^6$ bytes</td>
</tr>
<tr>
<td>gigabyte (GB)</td>
<td>1024 megabytes or $2^{30}$ bytes</td>
<td>1000 megabytes or $10^9$ bytes</td>
</tr>
</tbody>
</table>

#### Time units:

- nanosecond (ns): one-billionth ($10^{-9}$) of a second
- microsecond (μs): one-millionth ($10^{-6}$) of a second
- millisecond (ms): one-thousandth ($10^{-3}$) of a second

---

*IEC standard*  

---

* kilo binary byte