

EECE 417 Computer Systems Architecture

**Department of Electrical and Computer Engineering
Howard University**

Charles Kim

Spring 2007

Computer Organization and Design (3rd Ed)

-The Hardware/Software Interface

by

**David A. Patterson
John L. Hennessy**

CHAPTER 1

Computer Abstractions and Technology

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, finance, genomics...**
 - **Different manufacturers: Intel, Apple, IBM, Microsoft, Sun...**
 - **Different underlying technologies and different costs!**
- **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
- Both Hardware and Software affect performance:
 - Algorithm determines number of source-level statements
 - Language/Compiler/Architecture determine machine instructions (Chapter 2 and 3)
 - Processor/Memory determine how fast instructions are executed (Chapter 5, 6, and 7)
- Assessing and Understanding Performance in Chapter 4

What is a computer?

- **Components:**
 - input (mouse, keyboard)
 - output (display, printer)
 - memory (disk drives, DRAM, SRAM, CD)
 - network
- **Our primary focus: the processor (datapath and control)**
 - implemented using millions of transistors
 - Impossible to understand by looking at each transistor
 - We need...

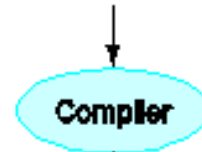
Abstraction

- 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?

High-level language program (in C)

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



Assembly language program (for MIPS)

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



Binary machine language program (for MIPS)

```
0000000010100010000000000011000
000000000000110000001100000100001
10001100011000100000000000000000
100011001111001000000000000000100
10101100111100100000000000000000
101011000110001000000000000000100
00000011111000000000000000001000
```

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!**

Instruction Set Architecture

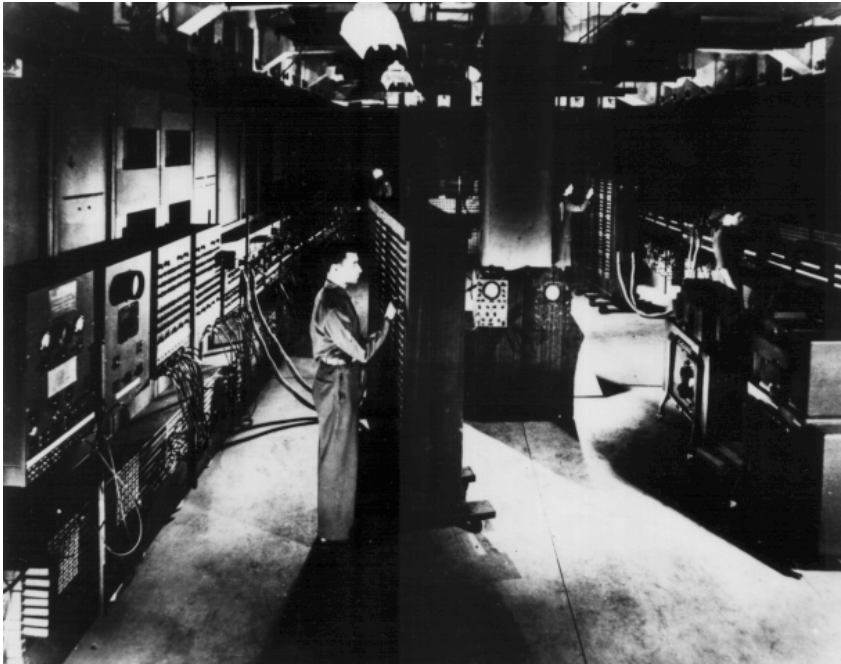
- A very important abstraction
 - interface between hardware and low-level software
 - standardizes instructions, machine language bit patterns, etc.
 - advantage: *different implementations of the same architecture*
 - disadvantage: *sometimes prevents using new innovations*

True or False: Binary compatibility is extraordinarily important?

- Modern instruction set architectures:
 - IA-32, PowerPC, MIPS, SPARC, ARM, and others

Historical Perspective

- **ENIAC built in World War II was the first general purpose computer**
 - **Used for computing artillery firing tables**
 - **80 feet long by 8.5 feet high and several feet wide**
 - **Each of the twenty 10 digit registers was 2 feet long**
 - **Used 18,000 vacuum tubes**
- **Performed 1900 additions per second**

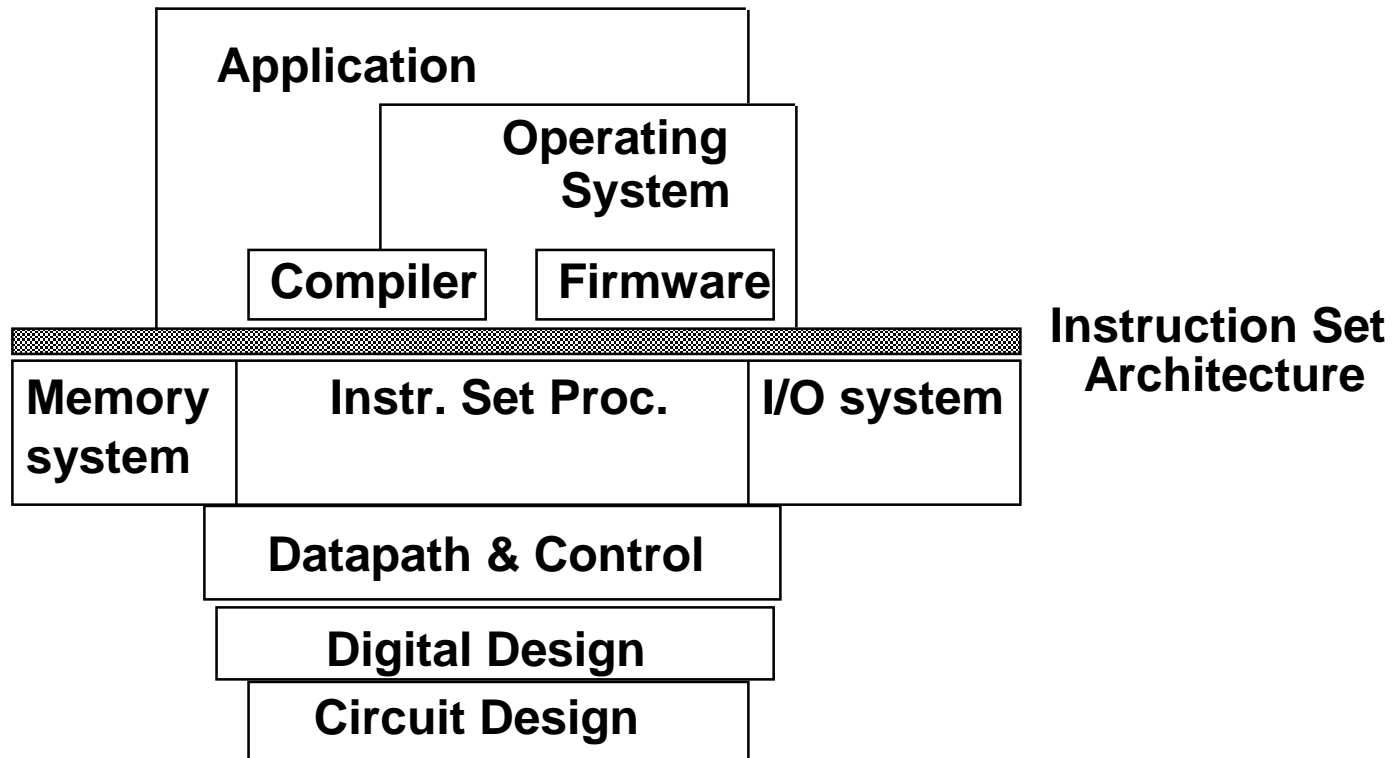


–Since then:

Moore's Law:

**transistor capacity
doubles
every 18-24 months**

How Does Each Piece Fit Together



[Irwin, PSU 2005]