Chapter 1: Computer Abstractions and Technology
Course Objectives

In this course, you will learn about:

- **How computers work**
  
  - How high-level language programs (software) are translated into a machine (hardware) language & how the hardware executes the resulting program → *program performance*

  - The interface between the software and the hardware. How does software instruct hardware to perform a desired function

  - What determines the performance of a program

  - Basic concepts of modern computer designs for improved performance
This course is all about how computers work

But what do we mean by a computer?
- Types: desktop, servers, embedded devices
- Use-case Scenarios: automobiles, graphics, finance, genomics...
- Vendors/manufacturers: Intel, Apple, IBM, Microsoft, Sun, HP...

Different underlying technologies, Different uses, Different vendors

Different costs!....

Analogy: Consider a course on “automotive vehicles”

Best way to learn:
- Focus on a specific instance and learn how it works
Why learn this stuff?

- Both Software and Hardware 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)

- You want to be a “computer scientist”
  - You want to build software people use (need performance)
  - You need to make a purchasing decision or offer “expert” advice
The Components of a Computer System?

- Von Neumann architecture has five functional units:
  1. **Memory**: disk drives, DRAM, SRAM, CD
  2. **Input**: mouse, keyboard
  3. **Output**: display, printer
  4. **Arithmetic/Logic unit**
  5. **Control unit**

- Sequential execution of instructions
- Stored program concept
The Concept of Abstraction (Hierarchical layers)

Layer – 1:

Layer -2:

Layer -3:
Components of the Von Neumann Architecture
Memory and Cache

- Information stored and fetched from memory subsystem
- Random Access Memory maps addresses to memory locations
- Cache memory keeps values currently in use in faster memory to speed access times
Memory and Cache (continued)

- RAM (Random Access Memory)
  - Memory made of addressable “cells”
  - Current standard cell size is 8 bits
  - All memory cells accessed in equal time
  - Memory address
    - Unsigned binary number N long
    - Address space is then $2^N$ cells
Structure of Random Access Memory
Input/Output

- Communication with outside world and external data storage
  - Human interfaces: monitor, keyboard…
  - Archival storage: not dependent on constant power
- External devices vary tremendously from each other
Volatile storage
- Information disappears when the power is turned off
  - Example: RAM

Nonvolatile storage
- Information does not disappear when the power is turned off
  - Example: mass storage devices such as disks and tapes, pen drives
Input/Output and Mass Storage (continued)

- **Mass storage devices**
  - Direct access storage device
    - Hard drive, CD-ROM, DVD, etc.
    - Uses its own addressing scheme to access data
  - Sequential access storage device
    - Tape drive, etc.
    - Stores data sequentially
    - Used for backup storage these days
Organization of an I/O Controller
The Datapath

- Actual computations are performed
- Primitive operation circuits
  - Arithmetic (ADD, etc.)
  - Comparison (CE, etc.)
  - Logic (AND, etc.)
- Data inputs and results stored in registers
- Multiplexor selects desired output
The Datapath (continued)

- Datapath process:
  - Values for operations copied into Datapath’s input register locations
  - All circuits compute results for those inputs
  - Multiplexor selects the one desired result from all values
  - Result value copied to desired result register
The Control Unit

- Manages stored program execution

- Task
  - Fetch from memory the next instruction to be executed
  - **Decode it**: determine what is to be done
  - **Execute it**: issue appropriate command to Datapath, memory, and I/O controllers
Primary Focus

- Our primary focus: the processor (Datapath and Control)
  - implemented using millions of transistors
  - Impossible to understand by looking at each transistor
  - There is a better way ...
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?
How 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
  - Boolean logic, 1s and 0s
  - Transistors used to build logic gates
  - Semiconductors/Silicon used to build transistors
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