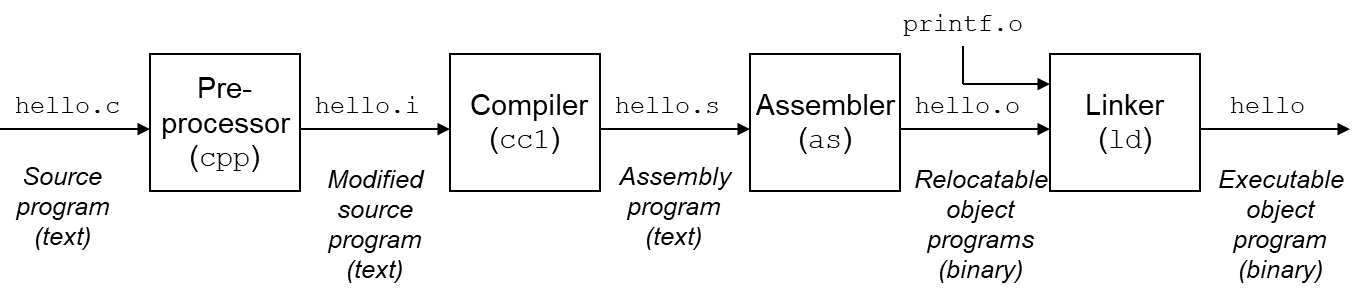
Week 1 Lecture

# B&O Chapter 1 – A Tour of Computer Systems

## Information is Bits Plus Context

* **Source program** (or **source file**)
  + A sequence of bits (either 0 or 1), organized in 8-bit chunks called **bytes**.
* Computer systems represent text characters using an encoding method such as **ASCII** or **Unicode**, which represent each character with a unique integer value
* Fundamentally, all information in a system is represented as a sequence of bits.
  + This includes disk files, programs stored in memory, user data stored in memory, data transferred across a network, etc.

## Programs are Translated by Other Programs into Different Forms

* Programs written in any higher-level language (e.g. C, Java, Python, etc.) must be translated by other programs into an executable form.
* In a C program, the high-level, human-readable code is translated into a sequence of low-level **machine-language** instructions, which are then packaged into an **executable object program** (or **executable object file**).
* On a Unix system, the translation from source file to object file is done by a **compiler driver**.
* The **GCC** compiler driver
  + Performs the translation in a sequence of four phases (listed below), with each phase being performed by four programs, which are collectively known as the **compilation system**.
    - Preprocessing
    - Compiling
    - Assembling
    - Linking

## It Pays to Understand How Compilation Systems Work

* For simple programs, we can trust that the compilation system will produce correct and efficient machine code, but for more complex programs, there are important reasons (listed below) why one might need to understand how compilation systems work.
  + Optimizing program performance
  + Understanding link-time errors
  + Avoiding security issues

## Processors Read and Interpret Instructions Stored in Memory

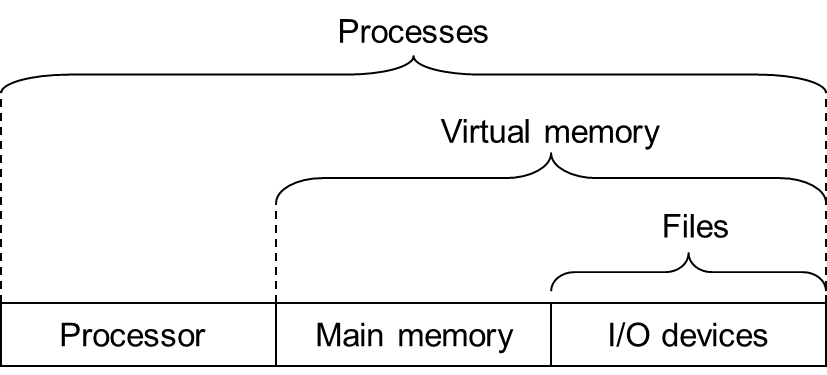
* In order to understand what happens when we run a program, we first need to understand the hardware of a typical system.
  + **Buses**
    - A collection of electrical conduits that carry bytes of information back and forth between components
  + **I/O Devices**
    - The system’s connection to the external world
    - Examples
      * keyboard (input)
      * mouse (input)
      * monitor/display (output)
      * disk drive
    - Each device is connected to the I/O bus by either a **controller** or an **adapter**, which transfer information back and forth between the I/O bus and the I/O device
      * A **controller** is a chip set in the device itself or on the system’s main printed circuit board (i.e. the motherboard)
      * An **adapter** is a card that plugs into a slot on the motherboard
  + Main Memory
    - A temporary storage device that holds both a program and the data it manipulates while the processor is executing the program
    - Physically, it consists of a collection of **dynamic random access memory** (**DRAM**) chips
    - Logically, memory is organized as a linear array of bytes, each with its own unique address starting at zero.
  + Processor
    - The central processing unit (CPU), or simply processor, is the engine that interprets (or executes) instructions stored in main memory.

## Caches Matter

* Caches serve as temporary staging areas for information that the processor is likely to need in the near future
  + Newer systems typically have L1, L2, and L3 caches
* Caches can be exploited to improve the performance of a programs by an order of magnitude
* The hardware technology used to implement these caches is know as **static random access memory** (**SRAM**).

## Storage Devices Form a Hierarchy

## The Operating System Manages the Hardware

* The **operating system** (**OS**) can be thought of as a layer of software interposed between the application program and the hardware
* All attempts that an application program makes to manipulate the hardware must go through the operating system
* The OS has two primary purposes:
  + To protect the hardware from misuse by runaway applications
  + To provide applications with simple and uniform mechanisms for manipulating complicated and often widely varying low-level hardware devices
* Both goals are achieved by the OS via processes, virtual memory, and files, which are all abstractions provided by the OS