Computer Architecture

CMSC 104 Fall 2025
September 2 2025

Administrative Notes

- All lectures will contain a slide labeled ‘administrative notes’ at the front
- This is where you will find announcements about
  - tests
  - projects
  - due dates
  - any changes in class schedule
  - anything else you need to know about the administration of this class

An overview of computer systems

Hardware and Software

- ***Hardware:*** those physical parts you can reach out and touch
- ***Software:*** the abstract logic that you can't see or touch; it executes on the hardware

Software Hierarchy

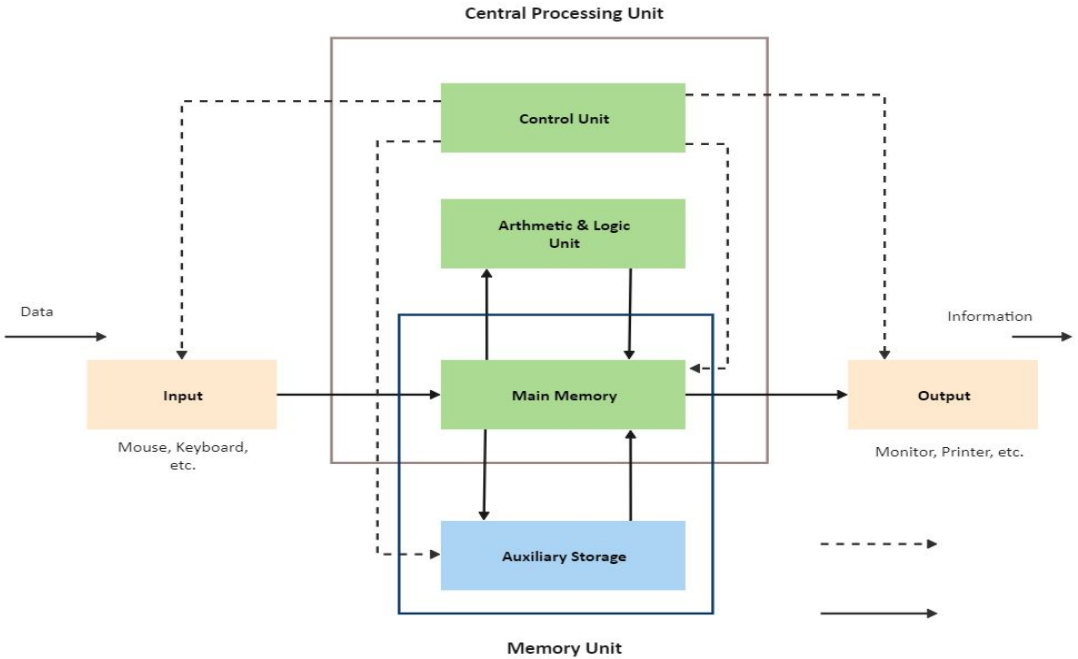
- Operating Systems
- Applications

Networks and Networking

- The Internet
- The World-Wide Web
- Web Services

Cloud computing

Typical computer architecture



From *https://edraw.wondershare.com/article/computer-architecture.html*

Phone, tablet, laptop, desktop - it really doesn't matter; the ideas are the same

Networks, Clouds and Virtual Machines

A network is a bunch of computers connected together to improve productivity

- Exchange information
- Work together to solve problems
- An “internet” was a network of networks; now there’s just “the Internet”

A “cloud” (see next slide)

Clouds

"a paradigm for enabling network access to a scalable and elastic pool of shareable physical or virtual resources with self-service provisioning and administration on-demand,"

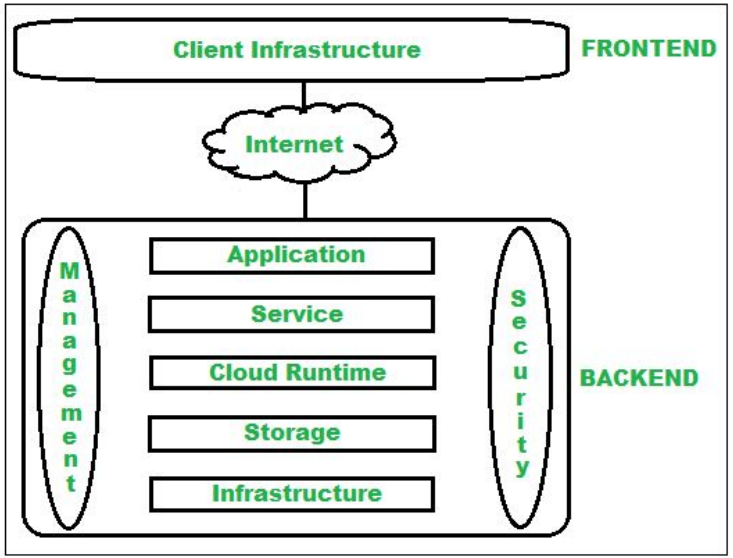
In plain English:

- A service for computing provided by somebody (Amazon, Google, Microsoft, Oracle,...) or even your own organization
- On-demand self-service - you get resources when you need them
- Resource pooling - provider has everything; any customer can get needed resources
- Broad network access - you don’t have to be sitting in a room or on a campus
- Rapid elasticity - you need more compute power? You got it (just pay for it)
- Measured service - pay for what you use

Why do you care?

- Because so much of computing today and going forward is based on clouds

From *https://www.geeksforgeeks.org/cloud-computing/architecture-of-cloud-computing/*



Virtual Machine

A “pretend” hardware/software computer constructed entirely out of software on a bigger computer

- Software emulates (pretends to be) the actual hardware, so to your program there’s no difference

Popular because computing is so cheap that most computers today are way more powerful than would be needed.

“Container” - related but different

- Figure out what each application really needs; package that together
- Less than a full VM
- Think of shipping containers on ships & trains

Where do the software and hardware fit in these pictures?

The hardware is what you can see and touch

- Display screen/monitor
- Keyboard
- Mouse
- Touch pad
- Fingerprint reader/faceID reader

There is ***system software*** that tells every piece of hardware what to do

There is ***application software*** that performs “useful” operations:

System software vs. Application software

System software is the software that controls the hardware device

- ***The Operating System*** is the overall control program
  - Windows, MacOS, Apple iOS, Android, Linux, …
- ***Device drivers*** control the interactions of the control unit with the keyboard, mouse, screen, USB drive, hard drive or SSD, etc.

Application software does “useful things”

- “Productivity apps” like Google Docs/MS Word
- Grades homework & tests
- Runs video games
- Manages money/does taxes
- Shows videos/live events

A quick digression on operating systems

1960s/1970s: project at MIT to build MULTICS - a multi-user, high performance, secure operating system

1970s: AT&T Bell Laboratories pulls out of MULTICS; Thompson, Ritchie and Kernighan write UNIX

1970s/1980s: Bell Labs shares UNIX code; UC-Berkeley writes its own “improved” version & distributes it

- lots of lawsuits follow about who owns what code

1990s: Linus Torvalds, a student in Finland, writes a version of Unix he calls “Linux” and distributes it free - “open source software”

- Decades of advancements follow, but Linux still embodies many of the original UNIX principles

Today:

Windows: Microsoft-proprietary, but has incorporated some principles from other operating systems

Apple MacOS and iOS: derived from the version of UNIX that was written at UC-Berkeley

Android: derived from Linux; purchased and controlled by Google

Linux: still open source and free; version control by Linus Torvalds and a small team. Many versions:

- RedHat
- Kali
- Debian
- Ubuntu

Data Hierarchy

Bit - one binary digit - a “1” or a “0”

Character - a sequence of bits (often, 8 or 16) that together represent one character in the language you care about - e.g, an “A” or a “9”

Fields - a collection of characters that together represent something meaningful - e.g., a “student last name” or a “student ID” or a “test grade”

Records - a collection of fields related to the same entity - e.g., everything UMBC knows about one student that’s relevant to this class

Files - a collection of records treated together - e.g., the student records of all of you in this class

Database - one or more files that contain all data relevant to some project/effort

Big Data - huge collections of data related in some way - e.g, “all Tweets sent out on Monday,” “all What’sApp messages sent from Germany

Programming

Language Hierarchy

Machine languages

- What actually runs on the hardware
- A string of 1’s and 0’s (different voltages, phases, whatever)
- Unique to every hardware manufacturer
- You’re never going to have to deal with this (be grateful!)

Assembly languages

- Quasi-human readable instruction sets
- One instruction per line of code
- Has to be assembled (translated to machine code) prior to executing

High-level languages

- What we deal with in this class
- Somewhat human-readable
- Examples:
    - Python (this class)
    - C/C++ (what this class used to cover)
    - Java
    - JavaScript

Where does Programming fit?

Aka “why do I need to know how to program?”

Everywhere

- What kind of programs do you want to write?
  - Operating systems? You can contribute to Linux distros if you want
  - Networking
  - Games
  - Financial applications
- The language you use and the operating system you write fro are tied to the kind of programming you want to do
  - C/C++ - operating systems, compilers, applications that need high performance
  - Python - big data, analytics, graphics, and even simple programs to perform a task
  - JavaScript - web services

The Python Programming Language

Python history

Released in 1991 by Guido van Rossum, a Dutch programmer

- Controlled by van Rossum and a 5-member steering committee that decides what new features to add
- There’s a new core version every few months

https://www.python.org is the location for the most current (and previous) “official versions”

Python philosophy

Extensibility is key

- Supports defining and using modules to provide new capabilities

Additionally:

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Readability counts.

Syntax and semantics

Every language has its syntax and semantics

- Syntax is the set of rules for constructing expressions in the language:
  - “Every sentence must have a subject and a predicate”
  - “Every paragraph must have a topic, and be related to that topic.”
- Semantics is understanding what expressions mean:
  - Often related to the context in which you’re speaking:
    - “Move those axes to the right”
      - Are we in the woods, and I want you to pick up some tools and carry them?
      - Are we working on a computer screen, and I want you to click on a line and drag it?

Python syntax and semantics

We'll be learning a lot about these this semester

- What to type if you want your program to display something on the screen?
- What's the difference between (, [, and {?
- Those are ***syntax*** issues
  - If you break a Python syntax rule, your program doesn't run
- What happens when you type

```
print(5+4**3)
```

- ***Semantics*** means figuring out what will happen when a statement is executed
- When you get the semantics wrong, your program runs but doesn't give you the answer you wanted or expected.

Here’s a real quick (ungraded) assignment I want everyone to do

Write your first Python program - print out the phrase “hello, world”

So what do we expect you to know from this lecture?

What's hardware, and what's software?

What's system software, and what's application software?

What's an operating system?

What language is taught in this class? Why?