What is a Computer?
What is a Computer?

A typical computer consists of:

- a CPU
- memory
- a hard disk
- a keyboard and monitor
- and one or more communication devices.
Communication Devices

- A *phone modem* uses a phone line and can transfer data in a speed up to 56,000 bps (bits per second).
- A *DSL* (digital subscriber line) also uses a phone line and can transfer data up to 10 Mbps.
- A *cable modem* uses the TV cable line maintained by the cable company. Cable modems are usually faster than DSL (30 Mbps).
- A Network Interface Card (*NIC*) is a device to connect a computer to a local area network (LAN). For example, a typical type of NIC, called 10BaseT, can transfer data at 10 Mbps (million bits per second).
Memory is a place to store data and program instructions for the CPU to execute.

- A program and its data must be brought to memory before they can be executed.
- A memory unit is an ordered sequence of bytes, each of which holds eight bits.
- Memory is never empty, but its initial content may be meaningless to your program. The current content of a memory byte is lost whenever new information is placed in it. (Think of the whiteboard in this room.)
Bits

- A computer is really nothing more than a set of switches.
  - Each switch exists in two states: on or off.
  - Storing information is simply a matter of setting a sequence of switches on or off.
  - If a switch is on, we say its value is 1. If a switch is off, its value is 0.
- A single value of 0 or 1 is called a bit (for binary digit). Data of various kinds, such as numbers, characters, and strings, are encoded as a series of bits (zeros and ones).
Bytes

- The smallest unit of storage a programmer can access is called a **byte**. A byte consists of 8 bits.
  - A byte is large enough to hold a small number (such as 3) or a single character (such as “Q”).

- If a program needs to store a large number that cannot fit into a single byte, it may be split across a number of adjacent bytes in memory.

- The way data is encoded within a byte varies depending on whether the data represents a number, a character, or something else. Programmers need not be concerned about the encoding and decoding of data, which is performed automatically by the system based on the encoding scheme.

<table>
<thead>
<tr>
<th>Memory address</th>
<th>Memory content</th>
<th>Encoding for character ‘J’</th>
<th>Encoding for character ‘a’</th>
<th>Encoding for character ‘v’</th>
<th>Encoding for character ‘a’</th>
<th>Encoding for number 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>01001010</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2001</td>
<td>01100001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2002</td>
<td>01110110</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2003</td>
<td>01100001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2004</td>
<td>00000011</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Large numbers of bytes

Since computers have a large capacity of memory, we describe large numbers of bytes using the SI prefixes.

- A kilobyte is about 1,000 bytes.
- A megabyte is about 1,000,000 bytes.
- A gigabyte is about 1,000,000,000 bytes.
- A terabyte is about 1,000,000,000,000 bytes.

We use the same prefixes used for describing other large numbers.

- “kilo-” = 1000, ”mega-” = 1,000,000, etc.
- Megahertz (MHz) and gigahertz (GHz) are measures of frequency.
- Megabits per second (Mbps) and gigabits per second (Gbps) are measures of the speed at which data can be transferred.
Large numbers of bytes

Since computers have a large capacity of memory, we describe large numbers of bytes using the SI prefixes.

- A kilobyte is exactly $2^{10} = 1,024$ bytes.
- A megabyte is exactly $2^{20} = 1,048,576$ bytes.
- A gigabyte is exactly $2^{30} = 1,073,741,824$ bytes.
- A terabyte is exactly $2^{40} = 1,099,511,627,776$ bytes.

We use the same prefixes used for describing other large numbers.

- “kilo-” = 1000, ”mega-” = 1,000,000, etc.
- Megahertz (MHz) and gigahertz (GHz) are measures of frequency.
- Megabits per second (Mbps) and gigabits per second (Gbps) are measures of the speed at which data can be transferred.
Storage Devices

- Memory is *volatile*, meaning the information contained is lost when the power is off.

- Programs and data are permanently stored in *secondary storage* and are moved to memory when the computer actually uses them. There are three main types of storage devices:
  - Disk drives (hard disks and floppy disks)
  - CD drives (CD-R and CD-RW)
  - Tape drives
CPU (Central Processing Unit)

- the “brain” of a computer: It retrieves instructions from memory and executes them.
- CPU speed is measured in gigahertz (GHz): 1 GHz = 1,000,000,000 pulses per second.
- CPU speed has been improved continuously. Intel® Core™2 Quad Processor Q9550: 2.83 GHz.
Structure of a computer and the von Neumann architecture

- John von Neumann proposed the Stored Program Model in a 1946 paper
- Program is stored in memory (as well as data)
- CPU fetches instructions one by one and executes them
**The “Fetch-Execute Cycle”**

The Central Processing Unit (CPU) in a computer executes a “fetch-execute cycle” over and over.

- Fetch an instruction
- Execute the instruction
- Repeat

The speed of this cycle is defined by the speed of the processor chip.
Computer programs, collectively known as software, are simply sets of instructions to the computer.

- A programmer tells a computer what to do through programs. Without software, a computer is an empty machine.
- Computers do not understand human languages, so you need to use computer languages to communicate with them. Programs are written using various different programming languages.
Machine Language

• *Machine language* is a set of primitive instructions built into every computer. It is the language that that computer’s CPU chip “speaks”.

• The instructions are in the form of binary code, so you have to enter binary codes for various instructions.

• Programming in a native machine language is a very tedious process. Moreover, the programs are highly difficult to read and modify. For example: to add two numbers, you might write an instruction in binary that looks like this:

\[1101000100100011\]
Assembly Language

- **Assembly languages** were developed to make programming easier.
- Assembly language is a language that is easier for humans to understand than machine language.
- Computers do not understand assembly language. However, people write programs called *assemblers* that can convert assembly language programs into machine language.
- For example, to add two numbers, you might write an instruction in assembly code like this:

  ```assembly
  ADD R1, R2, R3
  ```

  This would get translated into the string of 0s and 1s that makes up the machine language instruction.
High-Level Programming Languages

- *High-level programming languages* such as Python are even more English-like and easy to learn and program.
- The following is an example of a high-level language statement that computes the area of a circle with radius 5:

  \[
  \text{area} = 5 \times 5 \times 3.1416
  \]
- This statement would actually be translated into multiple machine language instructions.
Popular High-Level Languages

COBOL (CCommon Business Oriented Language)
FORTRAN (FORmula TRANslation)
BASIC (Beginner’s All-purpose Symbolic Instructional Code)
Pascal (named for Blaise Pascal)
Ada (named for Ada Lovelace)
C (strongly associated with the UNIX operating system)
Visual Basic (Basic-like visual language developed by Microsoft)
Delphi (Pascal-like visual language developed by Borland)
C++ (an object-oriented language, based on C)
Java
Swift
Python
Compilers vs. Interpreters

Compilers are programs that take a source file written in a high-level language, translate them, and produce an object file written in machine language.

The object file can then be executed by the computer.

C, C++, Java are typically compiled languages.
Compilers vs. Interpreters

Interpreters are programs that take individual instructions written in a high-level language, translate them, and execute them.

You can also accumulate instructions in a file and have the interpreter automatically run them one at a time.

Python, Ruby, and Swift are typically interpreted languages.
## Interpreters vs. Compilers

<table>
<thead>
<tr>
<th>Interpreter</th>
<th>Compiler</th>
</tr>
</thead>
<tbody>
<tr>
<td>Translates one statement at a time.</td>
<td>Scans the entire program and translates it into machine code.</td>
</tr>
<tr>
<td>Takes less time to analyze the source code, but the overall execution time is slower.</td>
<td>Takes more time to analyze the source code, but the overall execution time is faster.</td>
</tr>
<tr>
<td>No intermediate code is generated, so it’s memory-efficient.</td>
<td>Generates intermediate object code requiring linking, so it requires more memory.</td>
</tr>
<tr>
<td>Continues translating the program until the first error is met, in which case it stops. Debugging is typically easier.</td>
<td>Generates error messages only after scanning the entire program. Debugging may be harder.</td>
</tr>
</tbody>
</table>
An *operating system* (OS) is the software that manages a computer’s basic functions, such as scheduling, running applications, and controlling hardware.

- **Microsoft Windows** is currently the most widely used operating system.
- Apple laptops run **MacOS**.
- Many powerful mainframes and workstations run a version of **UNIX** or **Linux**.
- iPhones run **iOS**; other phones (such as Samsung) run **Android**.

Users interact with the OS to run programs called *applications*.

- Browsers
- E-mail, messaging, and social networking
- Productivity tools (word processors, spreadsheets, calendars)
- Games