Unit 3. Computers

A Lego RCX Computer is an example of an embedded computer used to control mechanical devices. It is fully programmable.

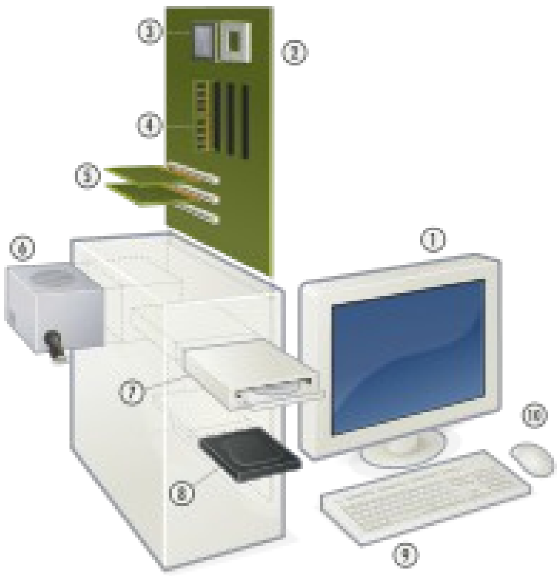
A **computer** is a machine for manipulating data according to a list of instructions known as a **program**.

Computers are extremely **versatile**. In fact, they are *universal* information-processing machines. According to the Church–Turing thesis, a computer with a certain minimum **threshold capability** is in principle capable of performing the tasks of any other computer. Therefore, computers with capabilities ranging from those of a personal digital assistant to a supercomputer may all perform the same tasks, as long as time and memory capacity are not considerations. Therefore, the same computer designs may be adapted for tasks ranging from processing company payrolls to controlling unmanned spaceflights. Due to technological advancement, modern electronic computers are exponentially more capable than those of preceding generations .

Computers take numerous physical forms. Early electronic computers were the size of a large room, while entire modern **embedded** computers may be smaller than a deck of playing cards. Even today, enormous computing **facilities** still exist for specialized **scientific computation** and for the **transaction processing requirements** of large organizations. Smaller computers designed for individual use are called **personal computers (PCs)**. Along with its **portable equivalent**, the **laptop** computer, the personal computer is the **ubiquitous** information processing and communication tool, and is usually what is meant by "a computer". However, the most common form of computer in use today is the embedded computer. Embedded computers are usually relatively simple and physically small computers used to control another device. They may control machines from fighter aircraft to industrial robots to digital cameras.

During the first half of the 20th century, many scientific computing needs were met by increasingly **sophisticated special-purpose analog computers**, which used a direct mechanical or electrical model of the problem as a basis for computation. These became increasingly rare after the development of the **programmable digital computer**.

How computers work: the stored program architecture



An exploded view of a modern personal computer:

1.Display **2**. Motherboard **3.** CPU (Microprocessor**)** **4.** Primary storage (RAM **5.** Expansion cards **6.** Power supply **7.** Optical disc drive **8.** Secondary storage (HD) **9.** Keyboard **10**. Mouse

While the technologies used in computers have changed dramatically since the first electronic, general-purpose computers of the 1940s, most still use the **stored program architecture** (sometimes called the von Neumann architecture). The design made the universal computer a practical reality.

The architecture describes a computer with *four main sections*: the **arithmetic** and **logic unit** (**ALU**), the **control circuitry**, the **memory**, and the **input** and **output devices** (collectively termed **I/O**). These parts are **interconnected** by **bundles of wires** (called **"buses"** when the same bundle supports more than one **data path**) and are usually driven by a **timer** or **clock** (although other events could drive the control circuitry).

Conceptually, a computer's memory can be viewed as a **list of cells**. Each cell has a **numbered** "**address"** and can **store** a small, fixed amount of information. This information can either be an **instruction**, telling the computer what to do, or data, the information which the computer is to **process** using the instructions that have been placed in the memory. In principle, any cell can be used to store either instructions or data.

The ALU is in many senses the *heart of the computer*. It is capable of *performing two classes of basic operations*. The first is arithmetic operations; for instance, **adding** or **subtracting** two numbers together. The **set of arithmetic operations** may be very limited; indeed, some designs do not directly support **multiplication** and **division** operations (instead, users support multiplication and division through programs that perform **multiple** additions, subtractions, and other **digit manipulations**). The second class of ALU operations involves *comparison* operations: given two numbers, determining if they are equal, or if not equal which is larger.

The **I/O systems** are the means by which the computer receives information from the outside world, and reports its results back to that world. On a typical personal computer, input devices include objects like the **keyboard** and **mouse**, and output devices include computer **monitors**, **printers** and the like, but as will be discussed later a huge variety of devices can be connected to a computer and serve as I/O devices.

The **control system** ties this all together. Its job is to read instructions and data from memory or the I/O devices, decode the instructions, providing the ALU with the correct inputs according to the instructions, "tell" the ALU what operation to perform on those inputs, and send the results back to the memory or to the I/O devices. One key component of the control system is a **counter** that keeps **track** of what the address of the current instruction is; typically, this is incremented each time **an instruction is executed**, unless the instruction itself indicates that the next instruction should be at some other location (allowing the computer to repeatedly execute the same instructions).

Since the 1980s the ALU and **control unit** (collectively called a **central processing unit** or **CPU**) have typically been located on a single integrated circuit called a **microprocessor**.

The functioning of such a computer is in principle quite straightforward. Typically, on each clock cycle, the computer fetches instructions and data from its memory. The instructions are executed, the results are stored, and the next instruction is fetched. This procedure repeats until a ***halt*** **instruction** is encountered.

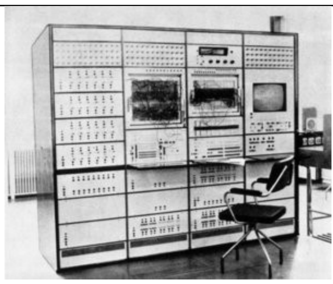
The set of instructions interpreted by the control unit, and executed by the ALU, are limited in number, precisely defined, and very simple operations. Broadly, they fit into one or more of four categories: **1)** moving data from one location to another (an example might be an instruction that "tells" the CPU to "copy the contents of memory cell 5 and place the copy in cell 10"). **2)** executing arithmetic and logical processes on data (for instance, "add the contents of cell 7 to the contents of cell 13 and place the result in cell 20"). **3)** testing the condition of data ("if the contents of cell 999 are 0, the next instruction is at cell 30"). **4)** altering the sequence of operations (the previous example **alters the sequence of operations**, but instructions such as "the next instruction is at cell 100" are also standard).

Instructions, like data, are represented within the computer as **binary code** — a base two system of **counting**. For example, the code for one kind of "**copy" operation** in the Intel x86 line of microprocessors is 10110000 [4]. The **particular instruction set** that a specific computer supports is known as that computer's **machine language**. Using an already-popular machine language makes it much easier to run existing **software** on a new machine; **consequently**, in markets where commercial software availability is important suppliers have **converged** on one or a very small number of **distinct machine languages**.

More **powerful computers** such as **minicomputers**, **mainframe** computers and **servers** may differ from the model above by dividing their work between more than one main CPU. **Multiprocessor** and **multicore** personal and laptop computers are also beginning to become available.[5][6]

Supercomputers often have highly unusual architectures significantly different from the basic stored-program architecture, sometimes featuring thousands of CPUs, but such designs tend to be useful only for specialized tasks. At the other end of the size scale, some microcontrollers use the Harvard architecture that ensures that program and data memory are logically separate.

Computer types:

 **analogue computer**	 **hybrid computer**
 **supercomputer**	 **mainframe**
 **workstation**	 **laptop**
 **roll-away**	 **embedded**

 **cart**	 **tablet PC**
 **handheld**	 **subnotebook**
  **thin client**	 **minicomputer**
 **computer terminal**	 **server**