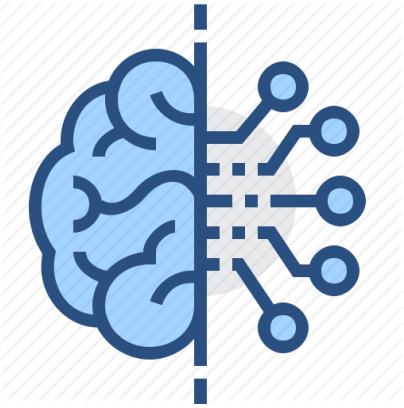
Thinking Machine - 1



How to Read This Book

This book is not about teaching, it's more about exploring. Author is noob trying to learn and know how computer works. Also, he wants to make projects to impress his crush on her birthday.

All the topics are divided into 3 major parts:

1. Why (Why is it important for author to read this new topics).
2. How (How will this topic help author in making effective projects).
3. What (What is in the topic to learn).

Complete this book, take free course at :

https://electrovertlabs.com/courses/thinking-machine-part-1/

Give quiz and download Completion certificate. Passing marks 75%

Special Thanks to:

Ron White: https://www.amazon.in/How-Computers-Work-Evolution-Technology-ebook/dp/B07D1B24HC

Tutorial Point Pvt Ltd: https://www.tutorialspoint.com/computer_fundamentals/index.htm

And Authors of blogs written on different topics.

By

(This book is compiled by* Electrovert Labs*)

Before starting, author wants to convey some message:

- It's totally okay if you don't know anything and completely new to Computers and Arduino.
  The fact that you are reading this makes you enter the group of people who wants to explore possibilities of the new world where everyone gets chance to do something meaningful and productive to change the world for good.

- Author is complete noob (new comer) in this world of computers. He don't want to teach anybody anything. He is just sharing whatever he is exploring and trying to make it simple.

- This book is not written for exam point of view, instead here author wants to explain how to think.

- Practice 5 Why concept. To dig deep in any topic, keep asking why, why 5 times and you'll know root cause.

- There is no point of memorizing anything, consider this book as a story of Computers.

- If you feel something is important, repeat it one more time.

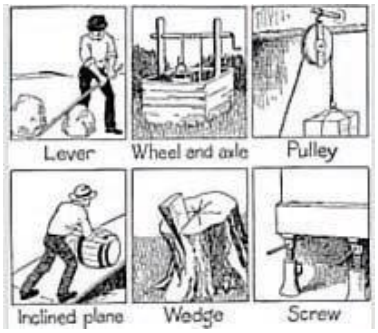
- **Learn the art of using google, it will help you a lot to clear your concepts and get answers of different questions going in your mind.**
- **Learn the art of using google, it will help you a lot to clear your concepts and get answers of different questions going in your mind.**

- Try not to skip theory part as knowing what you are using (computer) will give you visibility on what exactly is happening when you write codes or use Instagram.

- Try to maintain consistency throughout, learning how to write code is like playing video games. You'll easily get bored and feel weak in beginning and with time you'll be able to think logically on your own and deploy strategies to conquer.

Table of Contents

Introduction



THOSE OF US who studied physics in high school half a century ago had to learn about the six **simple machines**. They were the wheel and axle, level, pulley, inclined plane, wedge, and screw—all centuries-old tools that made physical work easier. The explanations of how they saved us labor were invariably accompanied by illustrations of people who looked like they were working awfully hard.

Just as invariably, someone would quote Archimedes saying, "Give me a long enough lever and a place to stand, and I will move the earth." Note that Archimedes gave somebody else the really hard job—finding a lever that would have to stretch from Earth to Jupiter. Still, in the days before any form of power other than the muscles of man and horse, the simple machines were ingenious.

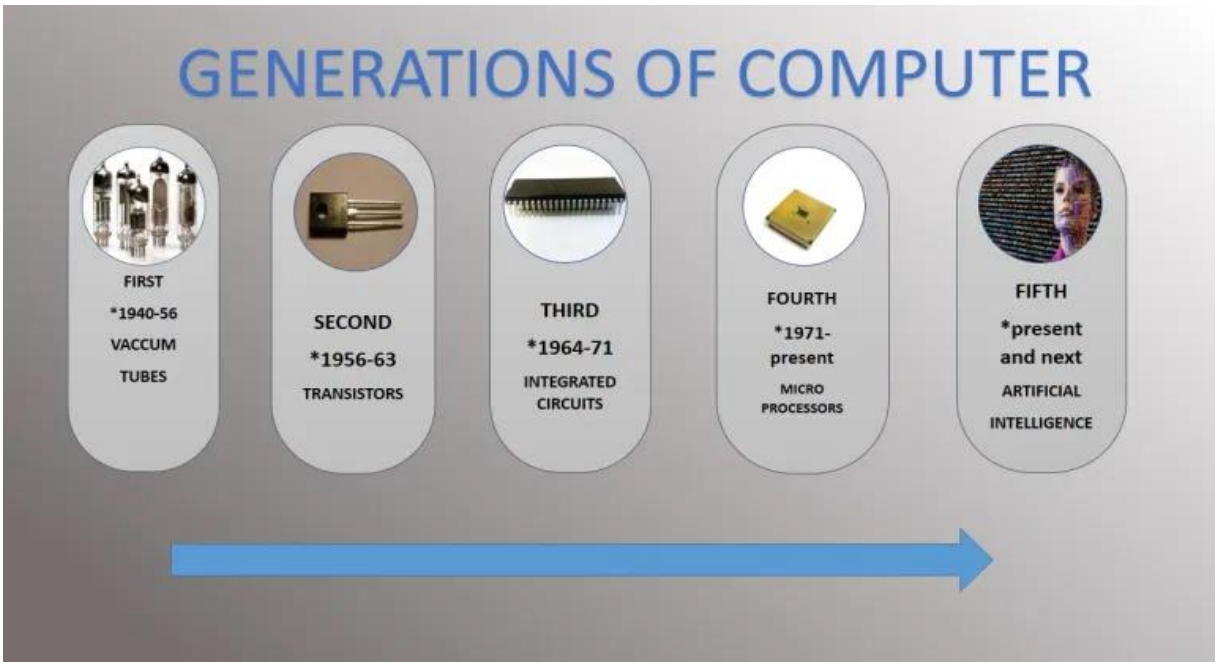
Those six machines were combined in various ways to create wagons, catapults, irrigation systems, roads, bridges, and, need I mention, the pyramids. Put some teeth on a wheel and you get a gear. Combine gears with levers and pulleys, and you have a clock. The ingenuity with which clever people combined simple machines into complex and powerful machines grew every century, abetted by engines running on water, steam, compressed air, and chemical fuels that magnified the power of these simple machines turned complex to a point that even Archimedes didn't imagine. Then came **electricity**.

Here was a mysterious force that would run through the right kind of metal wires. Those wires could be bent and wrapped around other wires and pieces of metal to produce an engine that, unlike water mills, was portable; one that, unlike steam and chemically fueled engines, didn't threaten to blow up. If electricity had produced no more than a safer, more efficient motor to run all those simple/complex machines, that in itself would have been a boon to civilization. But scientists and engineers discovered new properties in this new force despite the fact they didn't know what it was and couldn't even see it. You know the results: Everything from the electric light bulb to the computer, smartphone, or tablet on which you might be reading this very book. Electricity really began to pay off when engineers looked beyond its ability to make a filament glow or a motor turn.

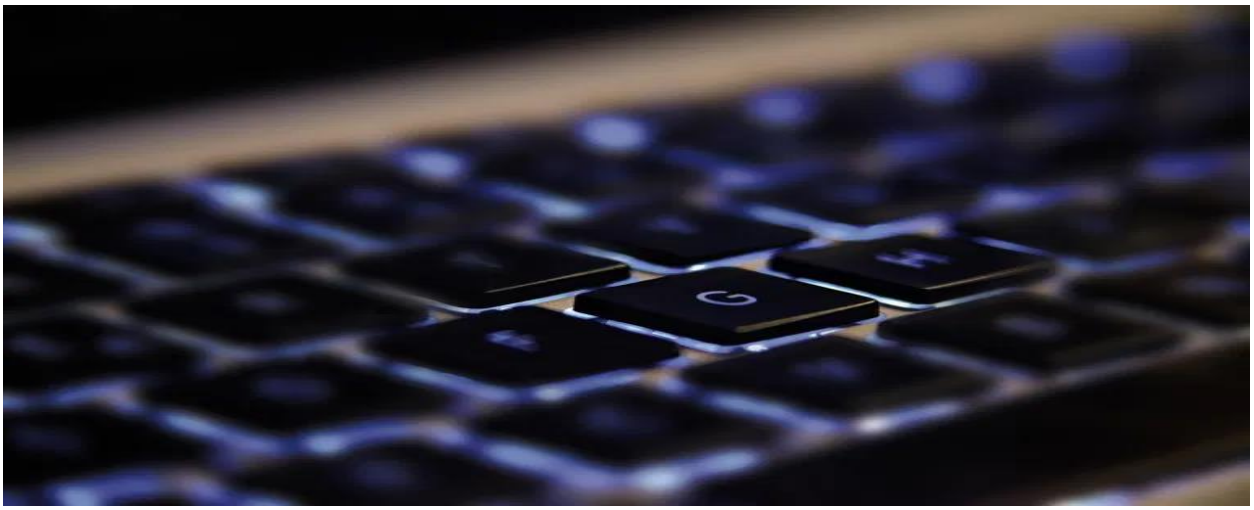
They began to discover more possibilities in electricity's partnership with magnetism, in the electromagnetic fields produced by electricity that could interact with other matter, even over great distances. They found that electricity and light were two facets of the same thing—an **electromagnetic spectrum** with mostly invisible fields made of waves vibrating at different frequencies and the sizes of which range from the microscopic to the Earth-encompassing.

Like the simple machines of medieval ages, these new elementary principles could be combined in endless ways to create new tools to accomplish jobs that were previously unimaginable.

The real power of electromagnetism, engineers found, was less in its ability to move trains, see inside a patient's body, and turn night into day than it was to store, manipulate, and distribute information. Behind all these uses are some very fundamental principles of physics that have led us to the brink of discovering the true nature of everything in the universe. It's these simple principles of a new age that you're going to look at in the first part of this book. I can already hear some of you saying, "Hey, when do we get to the stuff about computers?" You're there. The subjects of these first chapters *are* the stuff of computers. And it doesn't matter what type of computer you're talking about—mainframe, desktop, laptop, table, digital music player, camera, GPS, smartphone, smartwatch, or Glass. All are computers, and all work using some version of the principles and technology you'll find here.



WHY COMPUTERS?



Person1: Hey can you please multiply 2467 with 1231.
Person2: Yeah, give me 2 minutes.
Person1: Can you also calculate, (2467 – n) x 1231 for **n** between (1 - 500)?
Person2: Yeah let me find someone to do this.

And that's how necessity of Computers came.

We lazy humans don't want to do same thing over and over again, that's why we developed a system that will do repeated and boring tasks for us.

For further deep dive on this have a look at Wikipedia article:
https://en.wikipedia.org/wiki/Computer

HOW will it solve the purpose?

Depending on the hardware attached and with properly written software, they could basically do almost anything. On the highest levels, computers are already in control over a lot of things. Your phone, traffic lights, even your watch is all driven by computers with specialized hardware.

On the lowest level, computers basically deal with electricity. This has two states: either on or off. That's a bit. As this electricity goes through a bunch of transistors it will influence

other parts where electricity passes. Thus, turning a bit on might set all kinds of things in action.

When you extend it further, groups of bits will be capable of handling a bunch of instructions. Groups of bits will be sent to a processor which results in various transistors doing their work and some output is generated. And this output will determine the actions of other pieces of hardware.

A nice example of this are the Arduino hardware boards where you can actually write code that runs on a very small computer. And it has several ports where data either goes in or out. And even though the Arduino is extremely simple as a computer, it already shows how much you could do with a bit of computer hardware and additional programming.

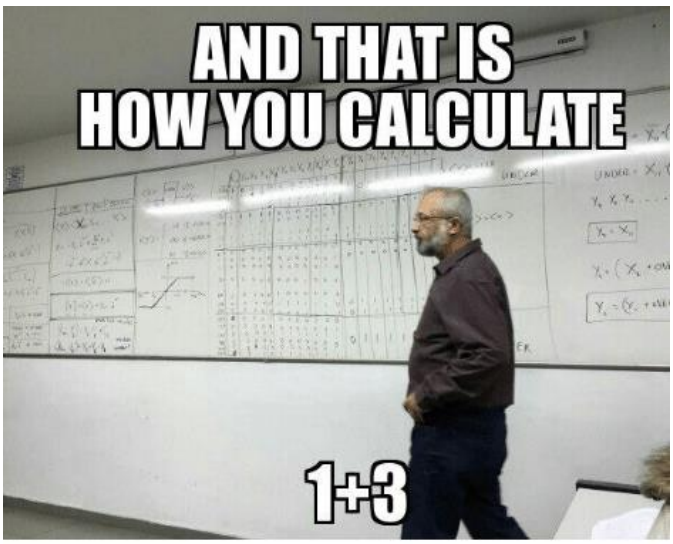
Although many users of the Arduino boards have never gone much further than making a LED blink, we will go much deeper.

We are going to explore Arduino in coming modules

Now, with proper hardware, you could provide a computer with enough information so it could drive a car through a large city from one end to the other without crashing while obeying all traffic laws! Google and some other companies have already made some very well-designed concepts of this and some of the techniques are already used in modern cars.

But the most common use for computers is generally when they're attached to a keyboard, mouse, monitor, Internet and printer. They are used for bookkeeping, math, all kinds of administrative tasks, viewing pictures of dogs, dresses and for watching lots and lots of porn. Plus, they are often used for games, for people to communicate with one another and ask questions like social networking sites and blogs.

So, computers can do quite a lot…

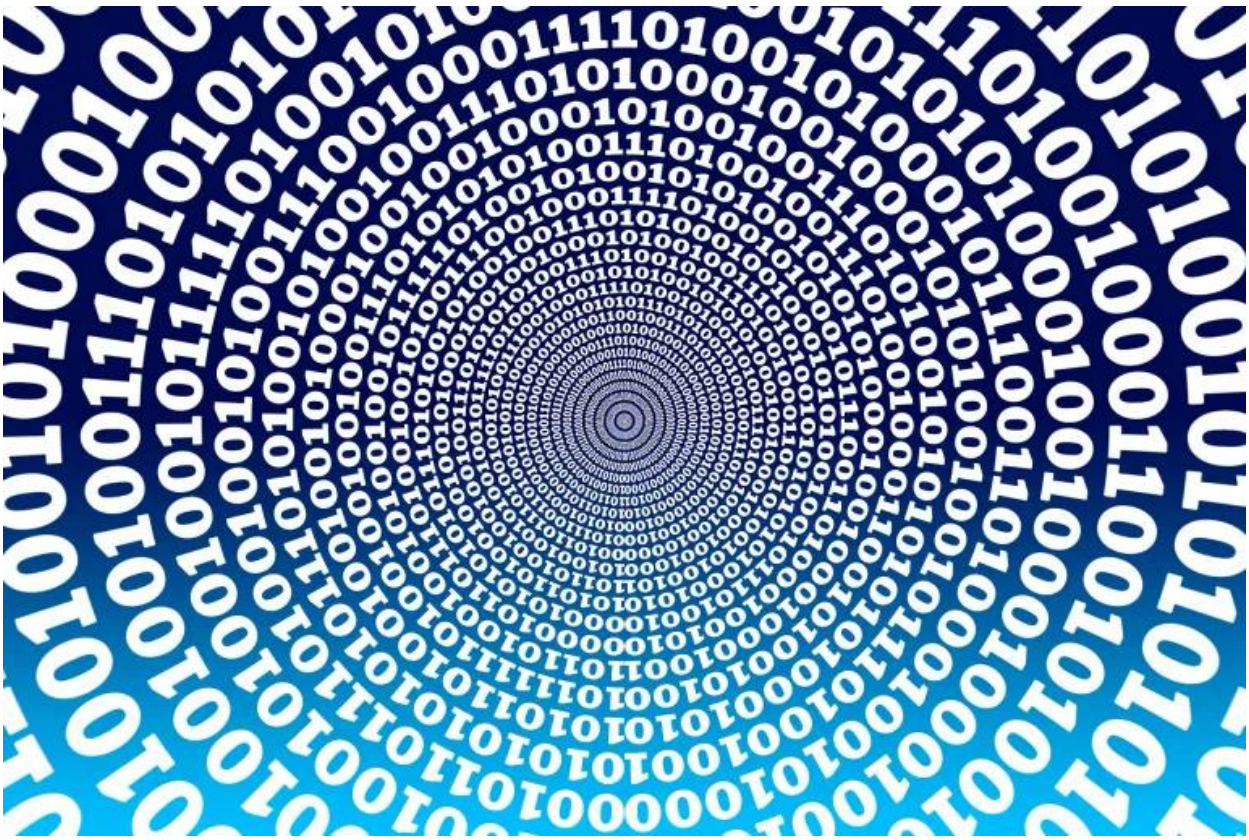


And good thing is you can calculate $12^{12} + 34^{34}$ too in no time.

You'll relate to this once you'll start coding ☺

Knowing how computer works (Hardware)

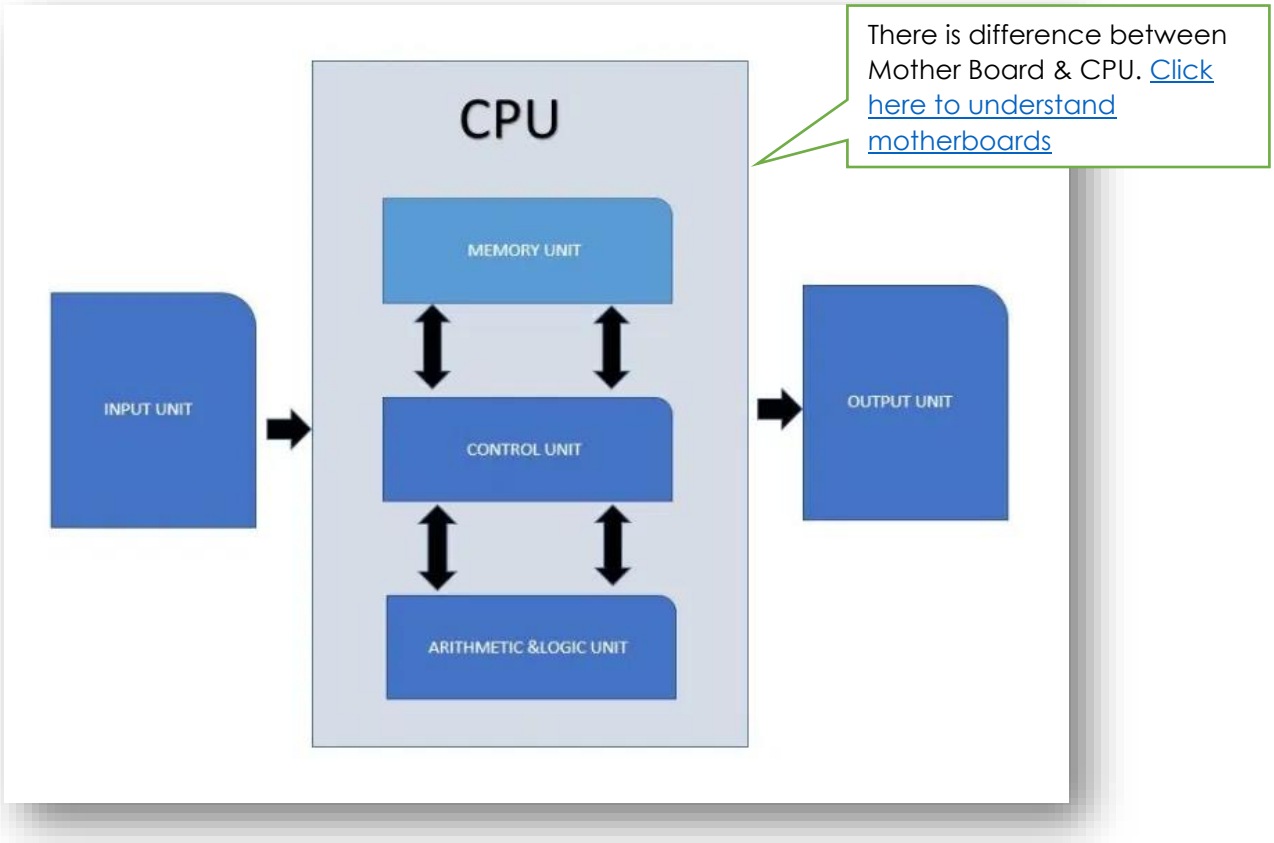
In this section we will know how a basic Computer architecture looks like from scratch.



All types of computers follow the same basic logical structure and perform the following five basic operations for converting raw input data into information useful to their users.

S.No.	Operation	Description
1	Take Input	The process of entering data and instructions into the computer system e.g. Keyboard, mouse, sensors, etc.
2	Store Data	Saving data and instructions so that they are available for processing as and when required.
3	Processing Data	Performing arithmetic, and logical operations on data in order to convert them into useful information.

4	Output Information	The process of producing useful information or results for the user, such as a printed report or visual display.
5	Control the workflow	Directs the manner and sequence in which all of the above operations are performed.



Input Unit

In computing, an input device is (a piece of computer hardware equipment) used to provide data and control signals to an information processing system such as a computer or information appliance. Examples of input devices include keyboards, mouse, scanners, sensors, etc.

CPU (Central Processing Unit)

CPU is considered as the brain of the computer. CPU performs all types of data processing operations. It stores data, intermediate results, and instructions (program). It controls the operation of all parts of the computer.

CPU itself has the following three components −

- ALU (Arithmetic Logic Unit)
- Memory Unit
- Control Unit

Output Unit

An output device is any device used to send data from a computer to another device or user. Most computer data output that is meant for humans is in the form of audio or video. Thus, most output devices used by humans are in these categories. Examples include monitors, projectors, speakers, headphones and printers.

Output devices allow computers to communicate with users and with other devices. This can include peripherals, which may be used for input/output (I/O) purposes, like network interface cards (NICs), modems, IR ports, RFID systems and wireless networking devices, as well as mechanical output devices, like solenoids, motors and other electromechanical devices.

Some of the most common output devices that people are familiar with include monitors, which produce video output; speakers, which produce audio output; and printers, which produce text or graphical output.

So we have very basic idea in our mind about architecture of a Computer.

Next step would be to understand how CPU works.

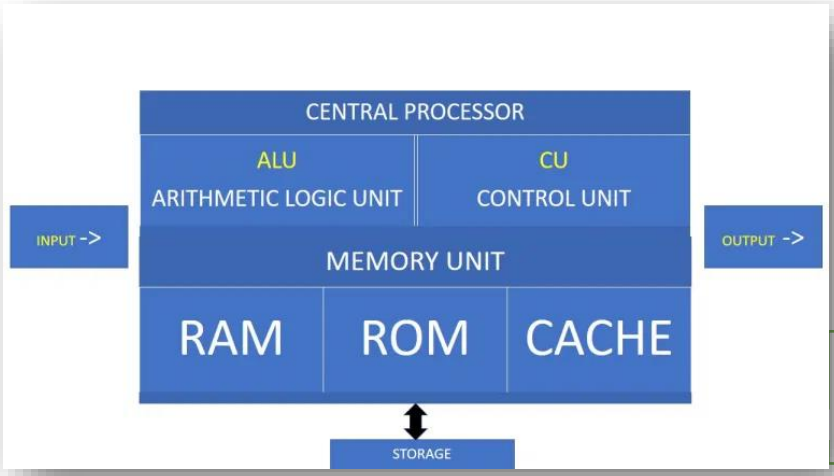
Computer - CPU (Central Processing Unit)

Central Processing Unit (CPU) consists of the following features −

- CPU is considered as the brain of the computer.
- CPU performs all types of data processing operations.
- It stores data, intermediate results, and instructions (program).
- It controls the operation of all parts of the computer.

CPU itself has following three components.

- Memory & Storage Unit
- Control Unit
- ALU(Arithmetic Logic Unit)



Memory Unit is RAM. Storage Unit is ROM and other storage devices.

Memory & Storage Unit:

This unit can store instructions, data, and intermediate results. This unit supplies information to other units of the computer when needed. It is also known as internal storage unit or the main memory or the primary storage or Random Access Memory (RAM). Its size affects speed, power, and capability. Primary memory and secondary memory are two types of memories in the computer.

Functions of the memory unit are −

- It stores all the data and the instructions required for processing.
- It stores intermediate results of processing.

- It stores the final results of processing before these results are released to an output device.
- All inputs and outputs are transmitted through the main memory.

Let's get little more into this topic:

Computer memory is storage area. It holds the data and instructions that the Central Processing Unit (CPU) needs. Before a program can run, the program is loaded from storage into the memory. This allows the CPU direct access to the computer program. Memory is needed in all computers.

A computer is usually a binary digital electronics device. Binary means it has only two states. On or Off. Zero or One. In a binary digital computer transistors are used to switch the electricity on and off. The computer's memory is made from lots of transistors.

Each on/off setting in the computer's memory is called a binary digit or bit. A group of eight bits is called a byte. Bit is the smallest unit of storage. It stores data in binary format such as 0's and 1's. But a single bit can't represent everything in the world that is to be stored. So a combination of bits i.e. 8 bits equal to 1 byte is used. Now the question arises why only 8 bits equals to 1 byte. So mathematically, n bits can form 2^n patterns. Example: 2^1 = 2 patterns, 2^2 = 4 patterns, 2^3 = 8 patterns and this goes like on 2^8 = 256 patterns. No. of patterns doubles every time, you increase the power by 1. So for 1 byte you have 256 patterns (0–255) and the total no. of characters in ASCII are 256. So for every character to be stored in memory it forms a unique pattern of 8 bits and then it is stored.

Characters in memory:

A byte of memory is used to store a code to represent a character such as a number, a letter or a symbol. Eight bits can store 256 different codes. This was thought enough and a byte became fixed at eight bits. This allows the ten decimal digits, 26 lower-case letters, 26 upper-case letters and many symbols. Early computers used six bits to a byte. This gave them 64 different codes. These computers did not have lower-case letters.

Computer scientists had to agree on which code would represent each character. Most modern computers use ASCII, the American Standard Code for Information Interchange. In ASCII each code is eight bits – any combination of 0s and 1s – and form one character. The letter A is denoted by the code 01000001.

To allow for all the different characters in all the worlds languages, modern computers need more than 256 different characters. Another code system called Unicode allows for 1,112,064 different characters by using from one to four bytes for each character.

Bytes are frequently used to hold individual characters in a text document. In the ASCII character set, each binary value between 0 and 127 is given a specific character. Most computers extend the ASCII character set to use the full range of 256 characters available in a byte. The upper 128 characters handle special things like accented characters from common foreign languages.
Computers store text documents, both on disk and in memory, using these codes. For example, if you use Notepad in Windows to create a text file containing the words, "Four score and seven years ago," Notepad would use 1 byte of memory per character (including 1 byte for each space character between the words -- ASCII character 32). When Notepad stores the sentence in a file on disk, the file will also contain 1 byte per character and per space.

Try this experiment: Open up a new file in Notepad and insert the sentence, "Four score and seven years ago" in it. Save the file to disk under the name test.txt. Then use the explorer and look at the size of the file.
You will find that the file has a size of 30 bytes on disk: 1 byte for each character. If you add another word to the end of the sentence and re-save it, the file size will jump to the appropriate number of bytes. Each character consumes a byte.

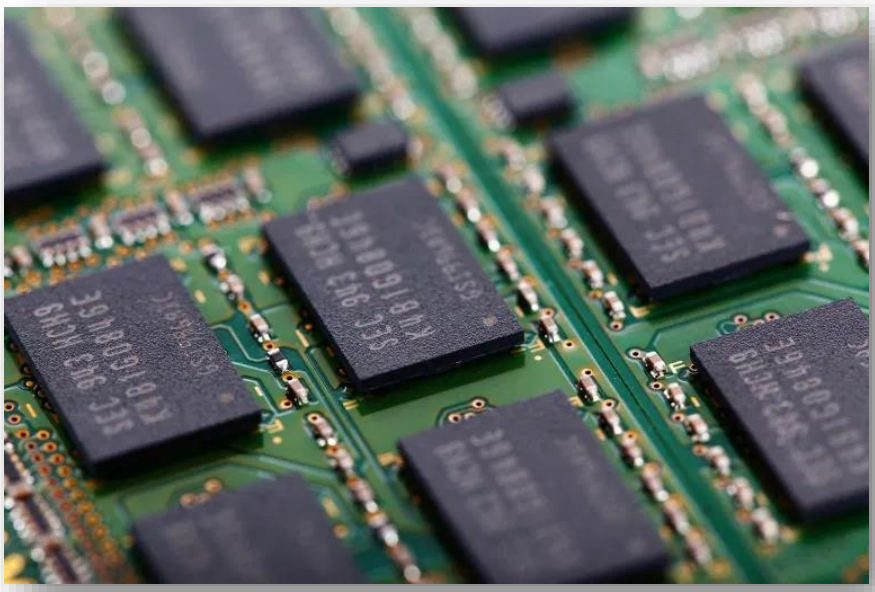
If you were to look at the file as a computer looks at it, you would find that each byte contains not a letter but a number -- the number is the ASCII code corresponding to the character (see below). So on disk, the numbers for the file look like this:

```
F   o   u   r     a   n   d       s   e   v   e   n
70 111 117 114 32 97 110 100 32 115 101 118 101 110
```

By looking in the ASCII table, you can see a one-to-one correspondence between each character and the ASCII code used. Note the use of 32 for a space -- 32 is the ASCII code for a space. We could expand these decimal numbers out to binary numbers (so 32 = 00100000) if we wanted to be technically correct -- that is how the computer really deals with things.
The first 32 values (0 through 31) are codes for things like carriage return and line feed. The space character is the 33rd value, followed by punctuation, digits, uppercase characters and lowercase characters. To see all 127 values, check out Unicode.org's chart.

Memory Address:

The computer's CPU can access each individual byte of memory. It uses an address for each byte. Computer memory addresses start at zero and go up to the biggest number the computer can use. Older computers were limited in how much memory they could address. 32-bit computers can address up to 4GB of memory. Modern computers use 64 bits and could address up to 18,446,744,073,709,551,616 bytes = 16 exabytes of memory.



The numbers that computers use can get very large. To make it easier, the unit K (for kilobyte) or Ki (for kibibyte) can be used. In computer memory numbers are powers of two. One Kibibyte is two to the power of 10, that is 2 x 2 x 2 x 2 x 2 x 2 x 2 x 2 x 2 x 2 and written as 2^{10} = 1024 bytes. For example, 64 Kibibytes, written as 64KiB or 64KB, of memory is the same as 65,536 bytes (1,024 × 64 = 65,536).

For larger memory capacities, the unit megabyte (MB) or mebibyte (MiB) and gigabyte (GB) or gibibyte (GiB) are used. One megabyte of computer memory means 2^{20} bytes or 1024KB, which is 1,048,576 bytes. One gibibyte means 2^{30} bytes or 1024MB.

The numbers are multiples of two. This is why a kilobyte of memory is 1024 bytes and not 1000 as would be the case for kilogram. To try to avoid this confusion the International Electrotechnical Commission (IEC) use the names kibibyte, mebibyte, and gibibyte for binary powers. They use kilobyte, megabyte, and gigabyte to mean powers of 10. The Joint Electron Device Engineering Council (JEDEC) have kept the older names. To make it worse the sizes of computer storage, such as hard disk drives (HDD), are measured in powers of ten. So a 500GB disk drive is 500 x 1000 x 1000 x 1000 bytes. This is a lot less that 500GB of memory which is 500 x 1024 x 1024 x1024. Most computer scientists still use the old names and have to remember that the units are different when talking about memory and storage devices.

What is the difference between a 32-bit and 64-bit system?

The terms "32-bit" and "64-bit" are commonly seen in system requirements and other technical literature, but few people actually know what these terms mean. Do they refer to hardware or software specifications? What makes a system 64-bit and what is the difference between a 64-bit computer and a 32-bit model? In most cases, you simply need to know if your operating system is 32-bit or 64-bit in order to run a certain program. However, when determining what software to install on your computer, it is helpful to understand what the terms actually mean.

Hardware:
32-bit and 64-bit are commonly used to describe processor architecture or design. A 32-bit processor includes a 32-bit register, which can store 2^{32} or 4,294,967,296 values. A 64-bit processor includes a 64-bit register, which can store 2^{64} or 18,446,744,073,709,551,616 values. Therefore, a 64-bit register is not twice as large as a 32-bit register, but is 4,294,967,296 times larger. That's a big difference, but how does it affect computing performance?

The CPU register stores memory addresses, which is how the processor accesses data from RAM. One bit in the register can reference an individual byte in memory, so a 32-bit system can address a maximum of 4 gigabytes (4,294,967,296 bytes) of RAM. The actual limit is often less – around 3.5 gigabytes – since part of the registry is used to store other temporary values besides memory addresses.

A 64-bit register can theoretically reference 18,446,744,073,709,551,616 bytes, or 17,179,869,184 gigabytes (16 exabytes) of memory. This is several million times more than an average workstation would need to access. What's important is that a 64-bit computer (which means it has a 64-bit processor) can access more than 4 GB of RAM. If a computer has 16 GB of RAM, it better have a 64-bit processor. Otherwise, at least 12 GB of the memory will be inaccessible by the CPU.

While 64 bits is far more storage than what modern computers require, it removes all bottlenecks associated with 32-bit systems. For example, 64-bit systems run more efficiently since memory blocks are more easily allocated. They also support 64-bit instructions and have 64-bit data paths, which enables them to process more data at once than 32-bit systems can.

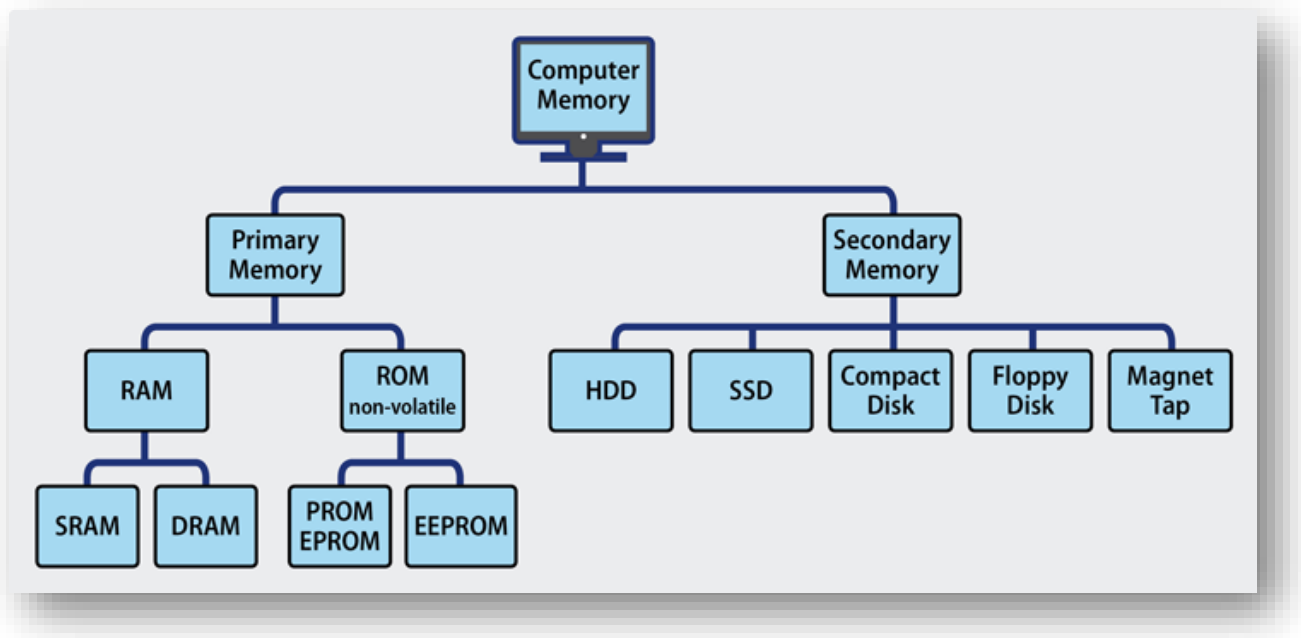
Software

So how does 32-bit or 64-bit hardware affect software? Generally speaking, 32-bit programs can run on a 64-bit system, but 64-bit programs will not run on a 32-bit system. This is because 64-bit applications include 64-bit instructions that will not be recognized by a 32-bit processor.

In order to run a 64-bit program, your operating system must be 64-bit. Around 2008, 64-bit versions of Windows and OS X became standard, though 32-bit versions were still available. Therefore, if you bought your computer in 2009 or later, there is a good chance you are running a 64-bit operating system. In Windows, you can check your OS version by right-clicking My Computer, selecting Properties, and clicking System to view the system type. If you have a Mac and you are running OS X 10.7 or later, your OS is 64-bit.

While it is possible to install a 32-bit operating system on a 64-bit system, it is best to install a 64-bit version if possible. The 64-bit OS will allow your computer to access more RAM, run applications more efficiently, and, in most cases, run both 32-bit and 64-bit programs.

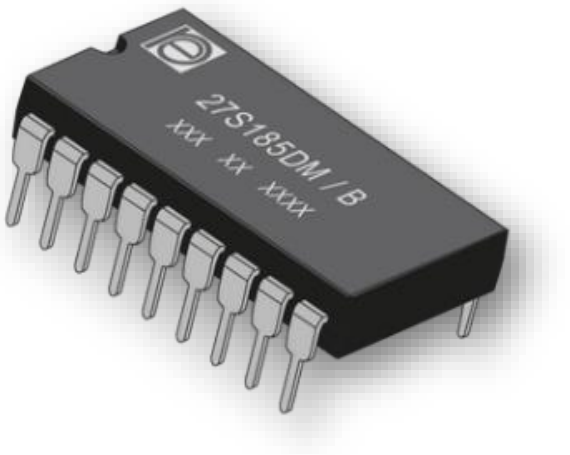
Exceptions to this rule include system utilities and antivirus programs that are written specifically for a 32 or 64-bit operating system. In these cases, you will need to install the version of the software that corresponds to your OS (which may be different than your hardware). Additionally, device drivers are often written for specific operating systems, so it is important to install 32-bit drivers if you are running a 32-bit OS and 64-bit drivers if your operating system is 64-bit.



Read only memory (ROM):

There are some programs and instructions which the computer will always need. Read only memory (ROM) is the permanent memory which is used to store these important control programs and systems software to perform functions such as booting up or starting up programs. ROM is non-volatile. That means the contents are not lost when the power is switched off. Its contents are written when the computer is built, but in modern computers, the user can change the contents using special software.

ROM or Read Only Memory is a part of any device containing a program, data, or information about the device, has been programmed during manufacturing. It is not easy to rewrite or modify any data on ROM.



Random access memory:

Random access memory (RAM) is used as the working memory of a computer system. It stores input data, intermediate results, programs, and other information temporarily. It can be read and/or written. It is usually volatile, which means that all data will be lost when the power is turned off. In most cases it is loaded again from the hard disk which is used as data storage.

RAM is of two types −

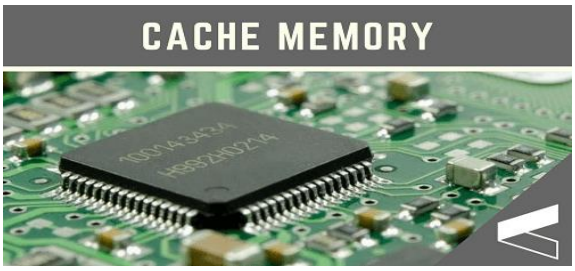
- Static RAM (SRAM)
- Dynamic RAM (DRAM)

SRAM	DRAM

• The word static indicates that the memory retains its contents as long as power is being supplied.
• However, data is lost when the power gets down due to volatile nature.
• SRAM chips use a matrix of 6-transistors and no capacitors. Transistors do not require power to prevent leakage, so SRAM need not be refreshed regularly.
• There is extra space in the matrix, hence SRAM uses more chips than DRAM for the same amount of storage space.
• SRAM is thus used as cache memory and has very fast access.
• Characteristic of Static RAM
  • Long life
  • No need to refresh
  • Faster
  • Used as cache memory
  • Large size
  • Expensive
  • High power consumption
	• DRAM, unlike SRAM, must be continually refreshed in order to maintain the data.
	• This is done by placing the memory on a refresh circuit that rewrites the data several hundred times per second.
	• DRAM is used for most system memory as it is cheap and small.
	• All DRAMs are made up of memory cells.
	• Each memory cell is made up of one capacitor and one transistor.
	• Characteristics of Dynamic RAM
	  • Short data lifetime
	  • Needs to be refreshed continuously
	  • Slower as compared to SRAM
	  • Used as RAM
	  • Smaller in size
	  • Less expensive
	  • Less power consumption

Cache Memory:

Cache memory is a very high speed semiconductor memory which can speed up the CPU. It acts as a buffer between the CPU and the main memory. It is used to hold those parts of data and program which are most frequently used by the CPU. The parts of data and programs are transferred from the disk to cache memory by the operating system, from where the CPU can access them.



Non-volatile memory:

Non-volatile memory is computer memory that keeps the stored information when not powered. Examples of non-volatile memory include:

- read-only memory
- flash memory

It can sometimes refer to computer storage. These are always non-volatile.
Examples include:

- Solid state devices that use flash memory, like Solid State Drives (SSD) and USB flash drives.
- Magnetic computer storage devices like hard disk drives (HDD), floppy disks and magnetic tape
- optical discs such as CD-ROM, DVD-ROM and Blu-ray
- paper storage such as paper tape and punched cards

Control Unit

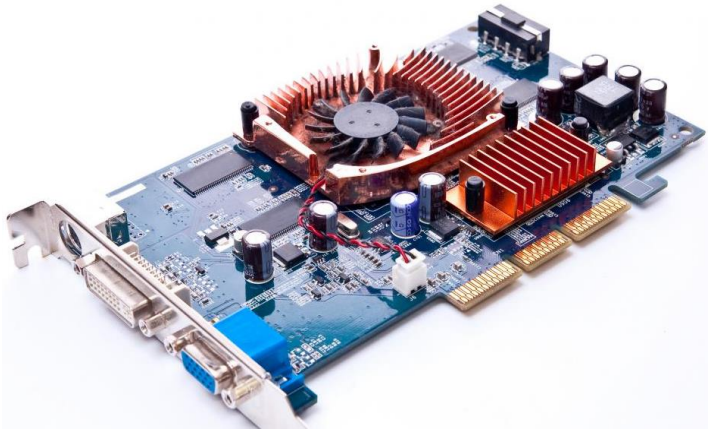
This unit controls the operations of all parts of the computer but does not carry out any actual data processing operations.

Functions of this unit are −

- It is responsible for controlling the transfer of data and instructions among other units of a computer.
- It manages and coordinates all the units of the computer.
- It obtains the instructions from the memory, interprets them, and directs the operation of the computer.
- It communicates with Input/output devices for transfer of data or results from storage.
- It does not process or store data.

A control unit is the subcomponent of the central processing unit (CPU) that manages all of the actions performed in this area in a computer. It is responsible for taking the various inputs from the computer, instructions and data and telling the processor what to do with them. Since the CPU is considered the brain of the computer, it is sometimes referred to as the brain within the brain. Depending on the CPUs architecture, the control unit may have varied tasks to perform.

The control unit is actually made up of several sub-components. During the hardwired days, all this wiring and circuitry formed what's known as a finite-state machine, a system having a singular purpose in directing the operations of the computer. Separate circuits were responsible for decoding and encoding instructions, while others handled logic or counting the instructions the CPU worked on. Everything happened in order, where the logic circuitry would be flipped one way or another to direct the instructions to storage.



An instruction is fetched and decoded, and then it needs to be executed in order, one after another until completion. In older CPUs, the instruction would have to go through the entire process and finish calculation before the next would begin. To speed up processing, modern CPUs use what are called pipelines, where each step is part of the pipeline. While one instruction is in the execution part of the pipeline, another is already in the decode phase, and another is being fetched. To handle all this, the control unit also needed to perform the role of a multiplexer, in that it takes multiple inputs or outputs and directs them into and out of the pipeline.

As computer CPUs continued to advance, much of this changed dramatically. The use of microcode, tiny programs that sit in special, high-speed read-only memory on the CPU, took the place of the old hardwired circuitry. These low-level programs took over the time-consuming job of physically rewiring a control unit and simplified changes to the CPU's architecture. The custom-written microprograms of the control unit, created during the CPU's design phase, are what enable the architecture of a particular type of CPU.

In general, much of the control unit's responsibilities depend on the CPU architecture. Some may simply fetch, decode, coordinate the execution, and direct the output of instructions. Others may have additional responsibilities that involve translation, which may slow down the CPU. In these cases, the control unit may be further split up into succinct components, such as a separate scheduling unit, or a retirement unit that takes care of organizing and storing the results from the arithmetic logic unit (ALU).

ALU (Arithmetic Logic Unit)

This unit consists of two subsections namely,

- Arithmetic Section
- Logic Section

Arithmetic Section:

Function of arithmetic section is to perform arithmetic operations like addition, subtraction, multiplication, and division. All complex operations are done by making repetitive use of the above operations.

Logic Section:

Function of logic section is to perform logic operations such as comparing, selecting, matching, and merging of data.

Arithmetic logic unit is a combinational digital electronic circuit that performs arithmetic and bitwise operations on integer binary numbers. This is in contrast to a floating-point unit (FPU), which operates on floating point numbers. An ALU is a fundamental building block of many types of computing circuits, including the central processing unit (CPU) of computers, FPUs, and graphics processing units (GPUs). A single CPU, FPU or GPU may contain multiple ALUs.

The inputs to an ALU are the data to be operated on, called operands, and a code indicating the operation to be performed; the ALU's output is the result of the performed operation. In many designs, the ALU also has status inputs or outputs, or both, which convey information about a previous operation or the current operation, respectively, between the ALU and external status registers.

For further details on ALU click on the link below:
https://en.wikipedia.org/wiki/Arithmetic_logic_unit

Computer – Motherboard

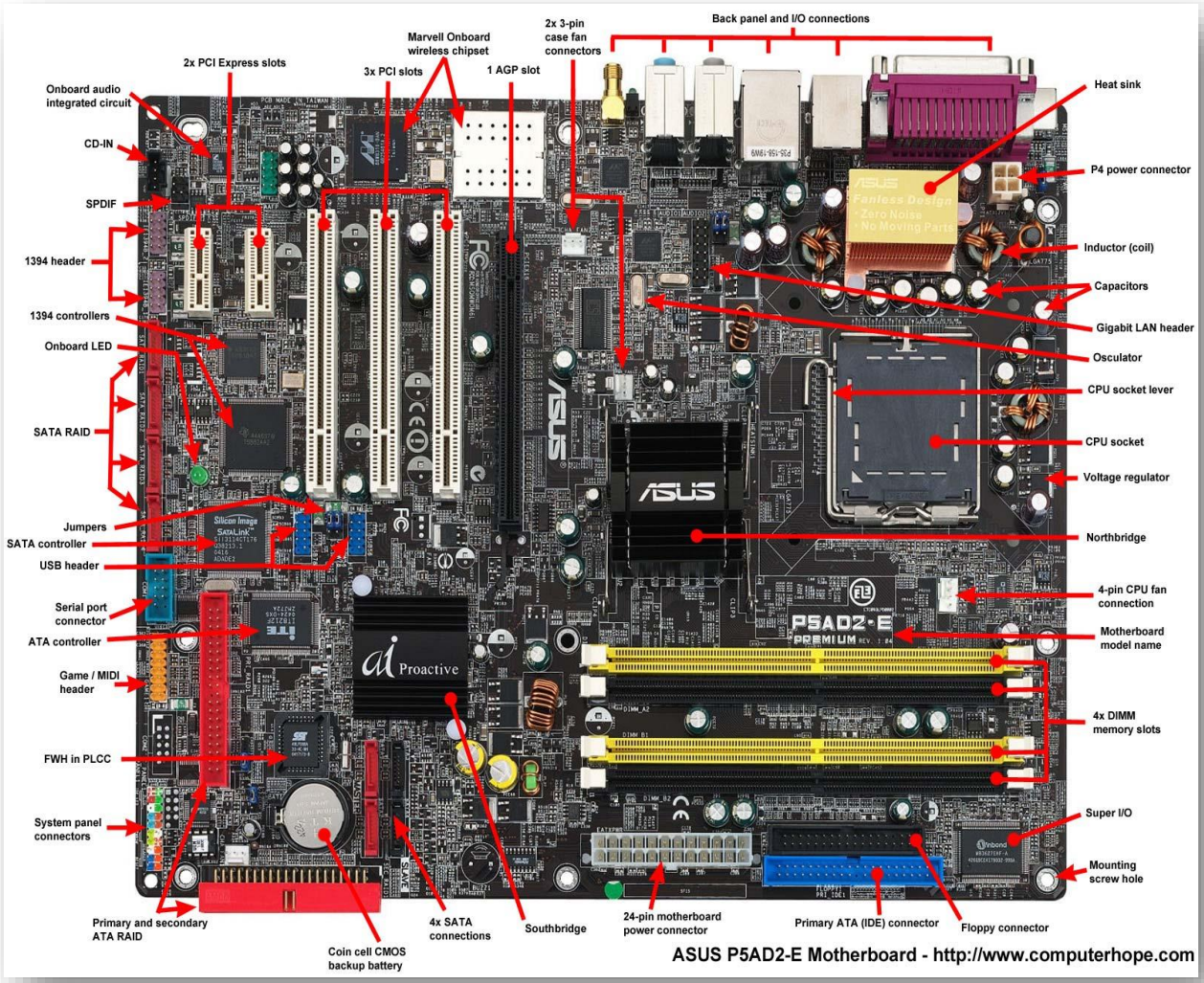
The motherboard is a printed circuit board and foundation of a computer that is the biggest board in a computer chassis. It allocates power and allows communication to and between the CPU, RAM, and all other computer hardware components.

Motherboard overview:

A motherboard provides connectivity between the hardware components of a computer, like the processor (CPU), memory (RAM), hard drive, and video card. There are multiple types of motherboards, designed to fit different types and sizes of computers.

Each type of motherboard is designed to work with specific types of processors and memory, so they are not capable of working with every processor and type of memory. However, hard drives are mostly universal and work with the majority of motherboards, regardless of the type or brand.

Below is a picture of the ASUS P5AD2-E motherboard with labels next to each of its major components.



For more details about Mother Board refer to the link below:
https://www.computerhope.com/jargon/m/mothboar.htm

How many connections, ports, or slots are on a motherboard?
There is no set standard to how many connections, ports, or expansion slots are on a motherboard. The best method of determining how many connections, ports, or slots are available for your motherboard is to look up the specifications contained in its documentation. If you've lost or discarded your motherboard's documentation, you can often download a free PDF version from the manufacturer's website.

How does a motherboard connect to a computer case?
A computer motherboard connects to a desktop computer case using standouts. Once the motherboard is attached to the case, all the other devices connect either to the motherboard itself or an installed expansion card.

What was the first motherboard?
The first motherboard is considered to be one used in the IBM Personal Computer, released in 1981. At the time, IBM referred to it as a "planar" instead of a motherboard. The IBM Personal Computer and the motherboard inside it would set the standard for IBM-compatible computer hardware going forward.

Is there a motherboard in a laptop, smartphone, and tablet?
Yes, although the board is often referred to as a "logic board" and not a motherboard. The logic board is very similar to a motherboard and operates the same way. However, because of size requirements with most logic boards, the components like the processor and RAM (in tablets and smartphones) are soldered onto the board. Also, because many of these devices have no upgrade options, there are no slots or sockets like a traditional computer motherboard.

What is Port in Computer/Computer Port?

A computer port or a communication port is a connection point used as an interface between the computer & the peripherals like keyboard, mouse, printer, display unit, monitor, flash drive and speaker. The computer port transmits the data from any peripheral to the computer. In general, the communication ports are available in two types and the classification of this can be done based on the protocol used & type for communication like Serial Ports as well as Parallel Ports.

Characteristics of Ports

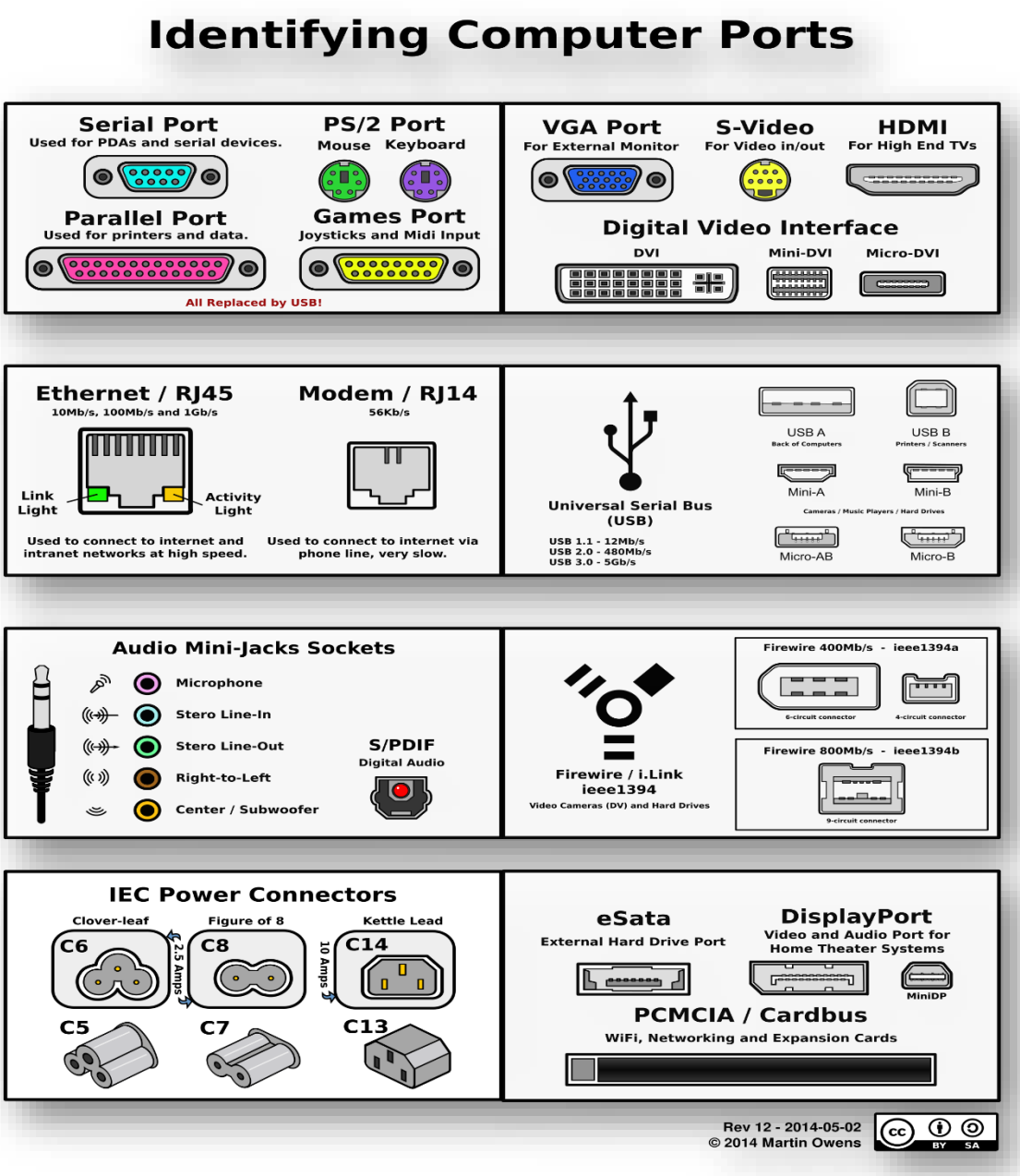
A port has the following characteristics −

- External devices are connected to a computer using cables and ports.
- Ports are slots on the motherboard into which a cable of external device is plugged in.
- Examples of external devices attached via ports are the mouse, keyboard, monitor, microphone, speakers, etc.

Let us now discuss a few important types of ports –

Serial Port
- Used for external modems and older computer mouse
- Two versions: 9 pin, 25 pin model
- Data travels at 115 kilobits per second

Parallel Port
- Used for scanners and printers
- Also called printer port
- 25 pin model
- IEEE 1284-compliant Centronics port



PS/2 Port

- Used for old computer keyboard and mouse
- Also called mouse port
- Most of the old computers provide two PS/2 port, each for the mouse and keyboard
- IEEE 1284-compliant Centronics port

Universal Serial Bus (or USB) Port

- It can connect all kinds of external USB devices such as external hard disk, printer, scanner, mouse, keyboard, etc.
- It was introduced in 1997.
- Most of the computers provide two USB ports as minimum.
- Data travels at 12 megabits per seconds.
- USB compliant devices can get power from a USB port.

VGA Port

- Connects monitor to a computer's video card.
- It has 15 holes.
- Similar to the serial port connector. However, serial port connector has pins, VGA port has holes.

Power Connector

- Three-pronged plug.
- Connects to the computer's power cable that plugs into a power bar or wall socket.

Firewire Port

- Transfers large amount of data at very fast speed.
- Connects camcorders and video equipment to the computer.
- Data travels at 400 to 800 megabits per seconds.
- Invented by Apple.
- It has three variants: 4-Pin FireWire 400 connector, 6-Pin FireWire 400 connector, and 9-Pin FireWire 800 connector.

Modem Port

- Connects a PC's modem to the telephone network.

Ethernet Port

- Connects to a network and high speed Internet.
- Connects the network cable to a computer.
- This port resides on an Ethernet Card.
- Data travels at 10 megabits to 1000 megabits per seconds depending upon the network bandwidth.

Game Port

- Connect a joystick to a PC
- Now replaced by USB

Digital Video Interface, DVI port

- Connects Flat panel LCD monitor to the computer's high-end video graphic cards.
- Very popular among video card manufacturers.

Sockets

- Sockets connect the microphone and speakers to the sound card of the computer.

Additional: How Computer Remembers (ONLY FOR GEEKS)

IT DOESN'T MAKE ANY DIFFERENCE if your electronic device of choice is a desktop computer, laptop, camera, smartphone, tablet, or glasses. Even if their functions wildly differ, all devices work using the same fundamental foundation of computing, the **transistor**. The transistor—the basic building block from which all microchips are built—has an unsubtle way of handling information: binary. It has two choices; nothing in between. If current passes through a transistor, the transistor stands for a 1, or if current doesn't pass through it, it becomes a 0. From these 1s and 0s, called **bits**, a computer can create any number as long as it has enough transistors grouped together to hold them all.

Binary notation starts off simply enough:

Decimal Number	Binary Number	Decimal Number	Binary Number
0	0	6	110
1	1	7	111
2	10	8	1000
3	11	9	1001
4	100	10	1010
5	101		

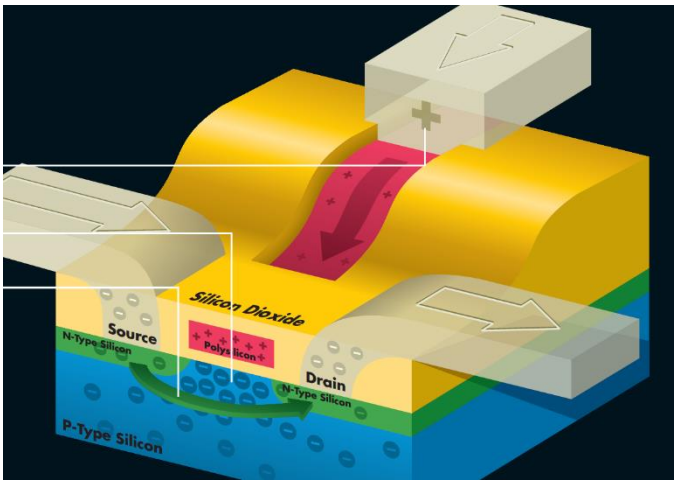
The processors in the first personal computers could work with 16 bits at a time, which limits them to numbers no bigger than the decimal number 65,535. Why, that's not big enough to count all the people in India! Luckily processor upgrades gave PCs the ability to work with 32 bits at a time, taking them up to the decimal notation of 4,294,967,295. Not bad, but the processors in common use in 2014 and (spoiler alert!) for a long time into the future can juggle 64 bits with the ease of The Flying Karamazov Brothers throwing swords back and forth—and a lot more easily than the boys juggling 281,474,976,710,655 swords, the decimal equivalent of 64 bits.

A processor being able to tackle more bits simultaneously doesn't result only in bigger numbers. Higher bit counts mean more data and instructions can be moved from memory into the processor in a single trip. That means faster computing. It's a mistake to think of transistors in terms of number alone, however. The same on/off bits can stand for true and false, enabling devices to work with **Boolean logic**. ("Select this AND this but NOT this.") Combinations of transistors in various configurations are called **logic gates**, which are combined into arrays called **half adders**, which in turn are combined into **full adders**, allowing the processor to calculate with numbers instead of merely counting. Transistors make it possible for a small amount of electrical current to control a second, much stronger current— just as the small amount or energy needed to throw a wall switch controls the more powerful energy surging through the wires to give life to a spotlight.

How a Little Transistor Does Big Jobs

THE MOST IMPORTANT INVENTION of the 20th century and the thing that will most influence technical and social progress in the 21stcentury is simply a switch. Well, not simply a switch. It's a transistor switch, a cousin of the wall switch that closes to let electricity flow through it or opens to block the current. The difference between the kin is that the transistor can be made exceedingly small. Its size gives it the ability turn off and on so rapidly that the number of times it closes or opens is measured in millionths of a second. And its size lets us join transistors into mighty armies that we call microprocessors, capable of conquering the prickly, unexplored wildernesses of science, music, and thought.

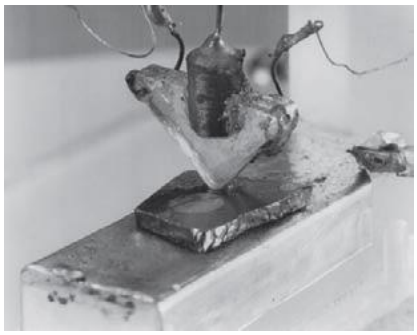
1. A small, positive electrical charge is sent down one aluminum lead that runs into the transistor. The positive charge spreads to a layer of electrically conductive polysilicon buried in the middle of nonconductive silicon dioxide. Silicon dioxide is the main component of sand and the material that gave Silicon Valley its name.



2. The positive charge attracts negatively charged electrons from the base made of P-type (positive) silicon that separates two layers of N-type (negative) silicon.

3. The rush of electrons out of the P-type silicon creates an electronic vacuum that is filled by electrons rushing from another conductive lead called the source. In addition to filling the vacuum in the P-type silicon, the electrons from the source also flow to a similar conductive lead called the drain. The rush of electrons completes the circuit, turning on the transistor so that it represents 1 bit. If a negative charge is applied to the polysilicon, electrons from the source are repelled and the transistor is turned off.

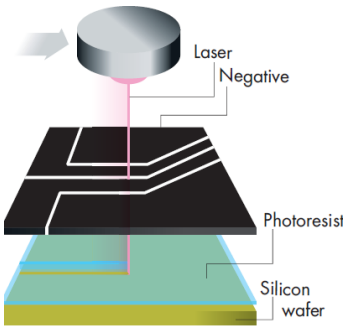
The Birth of a Microchip



The first silicon transistor, shown in the photo, was invented at Bell Labs in 1947. The device, made of two gold electrical contacts and a germanium crystal, was built on a human scale, which make its workings easily visible. But to be really useful, a transistor needs other transistors—so many that creating them on Bell Labs' human scale would guarantee only gargantuan gadgets. Scientists quickly developed ways that reduced the size and increased the efficiency of transistors. Today, a microchip is conceived on a computer that converts the engineers' logical concepts into physical designs used to fabricate the soul of a chip. The body of the chip is born of a thin, highly polished slice of silicon crystal grown in the lab. Although silicon is the second-most plentiful element on earth, it does not appear naturally in its pure form.

Building a Microprocessor

In a photoetching process, a chemical called photoresist covers the silicon wafer's surface. Intense thin beams of light are shown through a negative produced from the engineer's CAD designs. Or, the computer might guide a beam of electrons to trace the chip design. Photoresist that has been touched by beams of light or electrons is washed away, and where the silicon is no longer protected by photoresist, an acid bath etches channels into the silicon's surface. These channels are filled with conductive materials to create the circuits along which the computer's digital signals travel.



Writing Data to RAM

When you read data from a given memory address, the memory manager checks if there is a copy in the cache. If yes (cache hit), the copy is read into the register. If no (cache miss), the cache is first updated with a row of memory from RAM; then the data can be read into the register.

Conversely, when you write to a memory address, the data is just written to the cache, making room if required. Later, when the cache line must be freed or is explicitly flushed, the row of memory is copied to RAM.

If you perform successive reads and writes to the same address, it can very well arise that all changes are made in the cache and the RAM left untouched. From the processor's point of view, the presence of the cache is transparent. It just reads from and writes to memory addresses, and the memory manager optimizes the transfers.

Note that you don't "create" data in memory. When a program is loaded, some RAM space is assigned to it and reserved (possibly also initialized), and the addresses of the variables are "mapped" onto that space. Then the program is allowed to write to or read from these addresses.

In modern, multicore CPUs, cache management is much more complicated because several concurrent reads/writes to the same memory addresses can occur. The MMU must ensure that all caches and RAM remain coherent and "synchronize" their content.

The stack and the heap have a range of memory addresses allocated, and they are handled through the cache like ordinary program space. The stack is an essential resource during program execution and it is extremely frequently accessed. It is important that it be cached as well.

How RAM works?

Random access memory (RAM) is the best known form of computer memory. RAM is considered "random access" because you can access any memory cell directly if you know the row and column that intersect at that cell.

The opposite of RAM is serial access memory (SAM). SAM stores data as a series of memory cells that can only be accessed sequentially (like a cassette tape). If the data is not in the current location, each memory cell is checked until the needed data is found. SAM works very well for

memory buffers, where the data is normally stored in the order in which it will be used (a good example is the texture buffer memory on a video card). RAM data, on the other hand, can be accessed in any order.

Similar to a microprocessor, a memory chip is an integrated circuit (IC) made of millions of transistors and capacitors. In the most common form of computer memory, dynamic random access memory (DRAM), a transistor and a capacitor are paired to create a memory cell, which represents a single bit of data. The capacitor holds the bit of information -- a 0 or a 1 (see How Bits and Bytes Work for information on bits). The transistor acts as a switch that lets the control circuitry on the memory chip read the capacitor or change its state.

A capacitor is like a small bucket- that is able to store electrons. To store a 1 in the memory cell, the bucket is filled with electrons. To store a 0, it is emptied. The problem with the capacitor's bucket is that it has a leak. In a matter of a few milliseconds a full bucket becomes empty. Therefore, for dynamic memory to work, either the CPU or the memory controller has to come along and recharge all of the capacitors holding a 1 before they discharge. To do this, the mem-ory controller reads the memory and then writes it right back. This refresh operation happens automatically thousands of times per second.

The capacitor in a dynamic RAM memory cell is like a leaky bucket. It needs to be refreshed periodically or it will discharge to 0. This refresh operation is where dynamic RAM gets its name. Dynamic RAM has to be dynamically refreshed all of the time or it forgets what it is holding. The downside of all of this refreshing is that it takes time and slows down the memory.

How ROM works?

Similar to RAM, ROM chips contain a grid of columns and rows. But where the columns and rows intersect, ROM chips are fundamentally different from RAM chips. While RAM uses transistors to turn on or off access to a capacitor at each intersection, ROM uses a diode to connect the lines if the value is 1. If the value is 0, then the lines are not connected at all.

A diode normally allows current to flow in only one direction and has a certain threshold, known as the forward break over, that determines how much current is required before the diode will pass it on. In silicon-based items such as processors and memory chips, the forward break over voltage is approximately 0.6 volts. By taking advantage of the unique properties of a diode, a ROM chip can send a charge that is above the forward break over down the appropriate column with the selected row grounded to connect at a specific cell. If a diode is present at that cell, the charge will be conducted through to the ground, and, under the binary system, the cell will be read as being "on" (a value of 1). The neat part of ROM is that if the cell's value is 0, there is no diode at that intersection to connect the column and row. So the charge on the column does not get transferred to the row.

As you can see, the way a ROM chip works necessitates the programming of perfect and complete data when the chip is created. You cannot reprogram or rewrite a standard ROM chip. If it is incorrect, or the data needs to be updated, you have to throw it away and start over. Creating the original template for a ROM chip is often a laborious process full of trial and error. But the benefits of ROM chips outweigh the drawbacks. Once the template is completed, the actual chips can cost as little as a few cents each. They use very little power, are extremely reliable and, in the case of most small electronic devices, contain all the necessary programming to control the device. A great example is the small chip in the singing fish toy. This chip, about the size of

your fingernail, contains the 30-second song clips in ROM and the control codes to synchronize the motors to the music.

EPROM

Working with ROMs can be a wasteful business. Even though they are inexpensive per chip, the cost can add up over time. Erasable programmable read-only memory (EPROM) addresses this issue. EPROM chips can be rewritten many times. Erasing an EPROM requires a special tool that emits a certain frequency of ultraviolet (UV) light. EPROMs are configured using an EPROM programmer that provides voltage at specified levels depending on the type of EPROM used.

Once again we have a grid of columns and rows. In an EPROM, the cell at each intersection has two transistors. The two transistors are separated from each other by a thin oxide layer. One of the transistors is known as the floating gate and the other as the control gate. The floating gate's only link to the row (wordline) is through the control gate. As long as this link is in place, the cell has a value of 1. To change the value to 0 requires a curious process called Fowler-Nordheim tunneling. Tunneling is used to alter the placement of electrons in the floating gate. An electrical charge, usually 10 to 13 volts, is applied to the floating gate. The charge comes from the column (bitline), enters the floating gate and drains to a ground.

This charge causes the floating-gate transistor to act like an electron gun. The excited electrons are pushed through and trapped on the other side of the thin oxide layer, giving it a negative charge. These negatively charged electrons act as a barrier between the control gate and the floating gate. A device called a cell sensor monitors the level of the charge passing through the floating gate. If the flow through the gate is greater than 50 percent of the charge, it has a value of 1. When the charge passing through drops below the 50-percent threshold, the value changes to 0. A blank EPROM has all of the gates fully open, giving each cell a value of 1.

To rewrite an EPROM, you must erase it first. To erase it, you must supply a level of energy strong enough to break through the negative electrons blocking the floating gate. In a standard EPROM, this is best accomplished with UV light at a frequency of 253.7. Because this particular frequency will not penetrate most plastics or glasses, each EPROM chip has a quartz window on top of it. The EPROM must be very close to the eraser's light source, within an inch or two, to work properly.

An EPROM eraser is not selective, it will erase the entire EPROM. The EPROM must be removed from the device it is in and placed under the UV light of the EPROM eraser for several minutes. An EPROM that is left under too long can become over-erased. In such a case, the EPROM's floating gates are charged to the point that they are unable to hold the electrons at all.

EEPROMs and Flash Memory

Though EPROMs are a big step up from PROMs in terms of reusability, they still require dedicated equipment and a labor-intensive process to remove and reinstall them each time a change is necessary. Also, changes cannot be made incrementally to an EPROM; the whole chip must be erased. Electrically erasable programmable read-only memory (EEPROM) chips remove the biggest drawbacks of EPROMs.

In EEPROMs:

The chip does not have to removed to be rewritten.

The entire chip does not have to be completely erased to change a specific portion of it.

Changing the contents does not require additional dedicated equipment.

Instead of using UV light, you can return the electrons in the cells of an EEPROM to normal with the localized application of an electric field to each cell. This erases the targeted cells of the EEPROM, which can then be rewritten. EEPROMs are changed 1 byte at a time, which makes them versatile but slow. In fact, EEPROM chips are too slow to use in many products that make quick changes to the data stored on the chip.

Manufacturers responded to this limitation with Flash memory, a type of EEPROM that uses in-circuit wiring to erase by applying an electrical field to the entire chip or to predetermined sections of the chip called blocks. Flash memory works much faster than traditional EEPROMs because it writes data in chunks, usually 512 bytes in size, instead of 1 byte at a time.

How Hard Disk works?

How SSD works?

Click on the text and you'll be redirected to respective pages where it is well explained.

Knowing how computer works (Software)

Software — The Computer's Own Poetry (Origin & History)

The Language programmers use to create software sounds a lot like the fairy godmother's incantation: grep, mov, endif, cur_x, and selField. And software really is magical. You invoke your computer— maybe with a touch, at the command of your voice, with the wink of an eye, or by pointing at a little picture with a mouse and clicking—and suddenly all these things begin to happen that could only be witchery. Beautiful color images and voices and sounds emanate from your PC. The software looks at a few numbers and predicts banana futures in three months. Ask your smartphone for information on a person, a country, or a date, and the phone responds like a crystal ball. You ask the software in a tablet to take you high in the clouds over someone's home on the other side of the world, and you're there in seconds—seconds! Magic carpets just can't compete.

Software's abracadabra needn't be a mystery to those of us who don't speak VB, Python, Java, C, C++, C#, PHP, and other curiously named software languages. It's its own form of poetry. Within the words are hidden meanings, plots and subplots, forms more rigid than an iambic pentameter. And like a lot of ordinary poetry, it's difficult for the layman to understand it all. We have to turn to the experts—the machines that read software.

You've done just that. Before you bought your desktop PC, your iPad, or your Galaxy 5S, you had used software. Software doesn't include only things like Word, Photoshop, and Angry Birds. Software is also music recordings and Blu-ray discs. Blueprints for building a dog house, a dress pattern, and a telephone number are programs. A program is simply any set of instructions for an ordered series of actions. A program can exist as a printout of computer code or as a recipe in a cookbook. And you're already a programmer, even if you've never touched a computer.

Have you created a playlist of the best of "Kai Po Che!"? Have you set your DVR to record every episode of The Daily Show, excepting the reruns? Have you entered the phone numbers of your contacts into your phone? Each time, you were programming. A program is simply a recipe, whether for Aunt Hattie's chocolate pie or for how to calculate the paychecks for a thousand employees.

There are, inevitably, layers of nuance. Think of what you do when you use a microwave. You press buttons in a certain order to make the microwave work at a specific power level for a certain length of time, then change to a different power level, and so on until your program produces a steaming hot chicken curry. That's programming—a set of instructions in a particular order—created on the fly. But if you press a button that's preset for microwave popcorn, you're using software—a preconfigured set of programming instructions that, in this case, are recorded permanently in a microchip inside the microwave.

Hardware is capable of doing a variety of tasks, but without programming and software to control it, hardware can't do much of anything. By itself, for example, a claw hammer is useful only as a paperweight, but used by a carpenter, it can drive nails, pull out old nails, and even crack walnuts. The carpenter swinging the hammer is programming it on the fly. A home entertainment center is a complex system of hardware capable of generating a variety of sights and sounds, but it can't whisper a peep on its own. It needs software in the form of a disc, or a Roku video streamer, or at

least a basic cable connection. The signals from the disc, the network connection, and cable tell the hardware what electrical pulses to use to re-create the sounds of Mahler's Tenth Symphony or the sights of The Avengers.

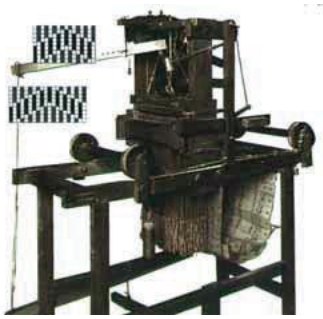
Software of the Second Millennium

If you think of software as a recorded set of instructions that control the actions of a machine, then software's really been around for quite some time. Music notation is written instructions for programming a piano on the fly. But if you have a player piano, the rolls of punched paper are its software. In the 18th century, weaving was done by someone manually slipping a bobbin wrapped with thread over and under other threads on a loom. The process was slow and rife with opportunities for slip-ups. But in 1804, Joseph-Marie Jacquard programmed a weaving loom using a series of punched cards that controlled what patterns were woven into a fabric. Different cards resulted in different patterns. Jacquard's invention is considered the birth of modern computer programming, and punch cards were used with computers well into the 20th century. The invention was also the occasion for the first wave of being scared by technology. In 1811, Ned Ludd, a Nottingham weaver who feared the new looms would replace people, led his co-workers in a frenzied attack on the machinery. Luddite has come to mean any person who resists technology advancements.

Software made little progress for the next century. In 1889, Thomas A. Edison invented the kinetoscope, a hardware device that let people view moving pictures on film. Film, sound recordings, and radio broadcasts—things we don't normally think of as software—are, in fact, just that, and the most prevalent form of software in the first half of the 20th century.

The first computers had neither a keyboard nor a monitor, and they had no software. Their creators programmed instructions by tediously flipping a series of switches in a precise arrangement. This set up a pattern of on and off electrical currents, and the computer in turn activated more electrical switches in the form of vacuum tubes. Finally, the result was displayed in the form of a panel of lights turned off or on to represent the zeroes and ones in the binary number system.

John von Neumann in 1945 first proposed the idea of a general-purpose electronic digital computer with a stored program. But the computer, the ENIAC, was not built until 1952. Meanwhile, scientists in England in 1948 created the Manchester Mark I, the first computer that could store a program electronically instead of making programmers set switches manually.



The Jacquard Loom

In 1804, Joseph-Marie Jacquard Programmed a weaving loom using a series of punched cards to form patterns in the fabric. Punch cards to enter data and instructions into computers were used well into the 20th century

During the next two decades, computers gained such accouterments as keyboards and displays. They used software in the form of punched cards, punched paper tape, and magnetic tape. All

those forms of software were awkward and slow to use compared to modern software, but they made life much easier for early programmers.

Not that there were many programmers back then to have easier lives. Some of the earliest writers of software sprang from a model railroad club at MIT. The railroaders used telephone switches to control their complex system of tracks, crossings, and rail switches. In the railroad, they had created a simple and very specialized form of computer. They already thought in terms of switches, the perfect training for writing software.

Most of the software writing then was going on in the universities, military, and businesses that were big enough to afford the then room-filling computers, called mainframes. But regardless of where different programs originated, in the 1960s they had one thing in common: They would work only on a specific computer for a specialized purpose. A program that was written, for example, to handle Gizmo Corp.'s payroll was customized specifically to match that company's accounting practices and record keeping, and it would run only on a computer configured like the one at Gizmo Corp. If another company wanted a payroll program, it was written from the ground up again. And if a manager at Gizmo wanted to look at payroll information from a different angle, the entire program had to be altered.

Programs back then had one other thing in common: They were either very expensive or free. Computer companies, such as IBM, saw software as a tool to sell hardware—"big irony." Because each program was a custom job—in a discipline that was still finding itself—companies expected to spend thousands of dollars. And compared to a computer that cost millions, a few thousand for software didn't seem so bad.

On the other hand, computer companies didn't charge for an operating system—the crucial software that lets a computer run a program that performs specific applications. Think of an operating system as the producer of a movie. The producer doesn't appear on screen. But that person works behind the scenes, hiring writers, directors and actors, finding and paying the hundreds of people responsible for building sets, lighting the scenes, driving stars to locations, and serving lunch—all so that the actors can do their best without having to worry about the hundreds of details that are essential to creating a blockbuster—or a flop.

Early on, programmers routinely swapped their software code with other programmers. Many saw computers as more than business tools. Computers, somehow, were going to be the great equalizer. Information was going to be the next industrial revolution, and computers were going to give Joe Blow access to the same information available to the board of GM. Such idealism, joined with the fact that all programmers were really just in the learning stage, led to a philosophy that all software should be distributed for free. That ideal is still alive today in the open source movement, where powerful operating systems such as Linux are free and users are encouraged to make improvements for all to share. Gradually, the nature of software changed. Computer companies began unbundling software from hardware. A new working class—the "cowboy," programmer for hire— traveled from company to company customizing programs and then moving on. In the mid-'60s, it occurred to some of these cowboys that they could write one program and sell it to several companies. A new industry was born.

Software — The Computer's Own Poetry

Every day we interact with software that helps us perform tasks and increase our efficiency. From the Microsoft Windows that greet us when we turn on the computer to the browser we use to surf the web, and the application on our smartphone that guides us on how many calories did we burn today! Each one of these different types of software helps us perform our day to day tasks either directly or indirectly.

Different Types of Software and How to Classify Them

In the first section, we will begin by classifying different types of software. But before that, let's first answer an essential question – What is software?

What Is Software – Definition and Examples of Software?

By definition, Software is a computer program that provides instructions and data to execute user's commands. It is an indispensable part of the machine you cannot see, but it allows you to use the computer … just like how a mouse, monitor, hard drive and keyboard help you use the computer.

Some common examples of software include Microsoft Word, Photoshop, Adobe Reader, Google Chrome, Gmail, PowerPoint, VLC, and many other similar computer programs that we often use in our daily life. If we sat down to list all the examples of software the list would never end, but what's more important than that is to understand how they differ from each other.

Types of software can be broadly classified into two categories.

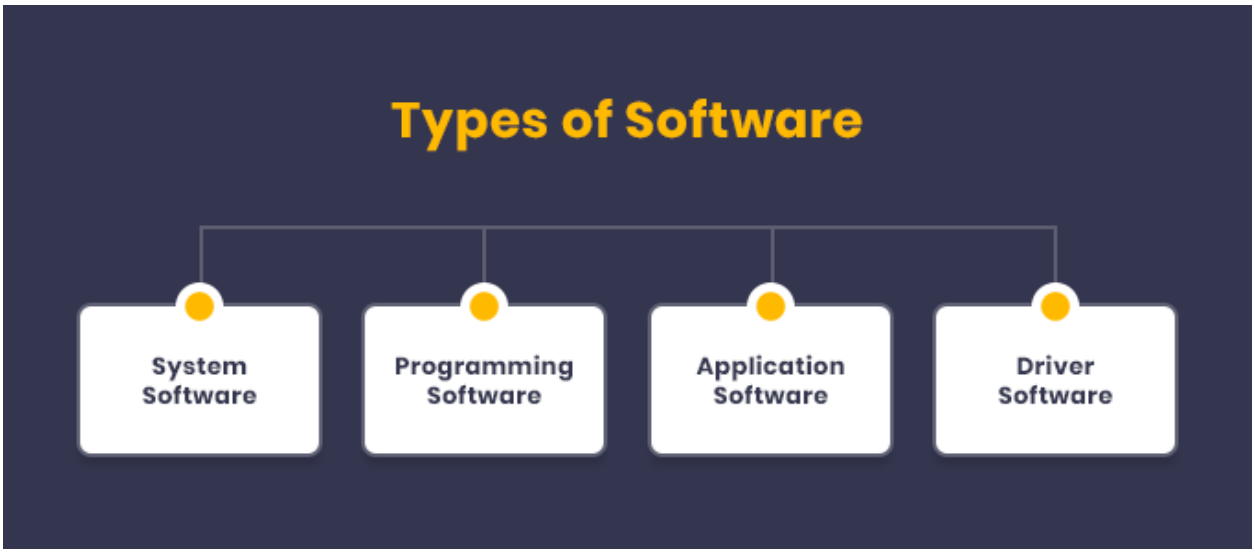
What Are The Two Major Software Types?

The two major types of computer software are:

- Application Software
- System Software

Whereas, two other types of computer software are:

- Programming Software
- Driver Software



Often programming and driver software are considered as types of system software.

Application Software

As a user of technology, Application Software or 'Apps' are what you engage with the most. These types of computer software are productive end-user programs that help you perform tasks. Following are some examples of application software that allow you to do specific work:

- MS Excel: It is spreadsheet software that you can use for presenting and analyzing data.
- Photoshop: It is a photo editing application software by Adobe. You can use it to visually enhance, catalog and share your pictures.
- Skype: It is an online communication app that you can use for video chat, voice calling and instant messaging.

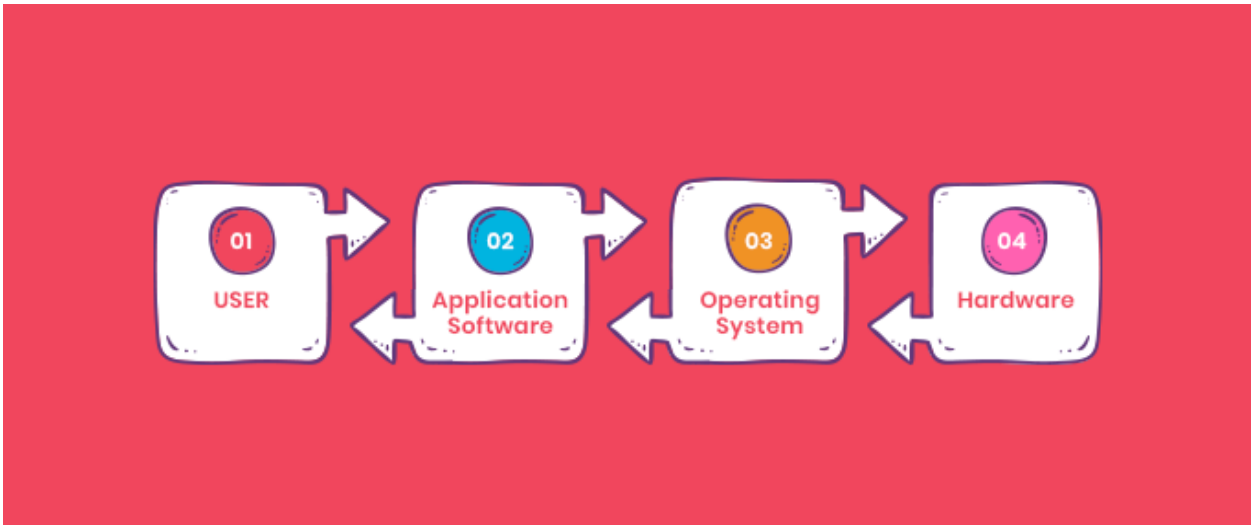
Software applications are also referred to as non-essential software. They are installed and operated on a computer-based on the user's requirement. There are plenty of application software that you can use to perform different tasks. The number of such apps keeps increasing with technological advances and the evolving needs of the users. You can categorize these software types into different groups, as shown in the following table:

Application Software Type	Examples
Word processing software: Tools that are used to create word sheets and type documents etc.	Microsoft Word, WordPad, AppleWorks and Notepad
Spreadsheet software: Software used to compute quantitative data.	Apple Numbers, Microsoft Excel and Quattro Pro
Database software: Used to store data and sort information.	Oracle, MS Access and FileMaker Pro
Application Suites: A collection of related programs sold as a package.	OpenOffice, Microsoft Office
Multimedia software: Tools used for a mixture of audio, video, image and text content.	Real Player, Media Player
Communication Software: Tools that connect systems and allow text, audio, and video-based communication.	MS NetMeeting, IRC, ICQ
Internet Browsers: Used to access and view websites.	Netscape Navigator, MS Internet Explorer, and Google Chrome
Email Programs: Software used for emailing.	Microsoft Outlook, Gmail, Apple Mail

System Software

System Software helps the user, hardware, and application software to interact and function together. These types of computer software allow an environment or platform for other software and applications to work in. This is why system software is essential in managing the whole computer system.

When you first power up your computer, it is the system software that is initially loaded into memory. Unlike application software, the System software is not used by end-users like you. It only



runs in the background of your device, at the most basic level while you use other application software. This is why system software is also called "low-level software".

Operating systems are an example of system software. All of your computer-like devices run on an operating system, including your desktop, laptop, smartphone, and tablet, etc. Here is a list of examples of an operating system. Let's take a look and you might spot some familiar names of system software:

For desktop computers, laptops and tablets:

- Microsoft Windows
- Mac (for Apple devices)
- Linux

For smartphones:

- Apple's iOS
- Google's Android
- Windows Phone OS

Operating System: https://www.youtube.com/watch?v=v91sNM6BgK8

Additional: What Exactly Happens When You Turn On Your Computer? (Detailed, for short description Google it)

YouTube video: https://www.youtube.com/watch?v=v91sNM6BgK8

When you power on a computer, it goes through a "boot up" process– a term that comes from the word "bootstrap." Here's what's happening in the background—whether you're using a Windows PC, Mac, or Linux system.

Booting is a process of starting a computer.

The Hardware Powers On

When you press the power button, the computer supplies power to its components—the motherboard, CPU, hard disks, solid state drives, graphics processors, and everything else in the computer.

The piece of hardware that supplies power is known as the "power supply." Inside a typical desktop PC, it looks like a box at the corner of the case (the yellow thing in the picture below), and it's where you connect the AC power cord.

The CPU Loads the UEFI or BIOS

Now that it has electricity, the CPU initializes itself and looks for a small program that is typically stored in a chip on the motherboard.

In the past, the PC loaded something called a BIOS (Basic Input/Output System.) On modern PCs, the CPU loads UEFI (Unified Extensible Firmware Interface) firmware instead. This is a modern replacement for the old-style BIOS. But, to make it extra confusing, some PC manufacturers still call their UEFI software "BIOS" anyway.

The UEFI or BIOS Tests and Initializes Hardware

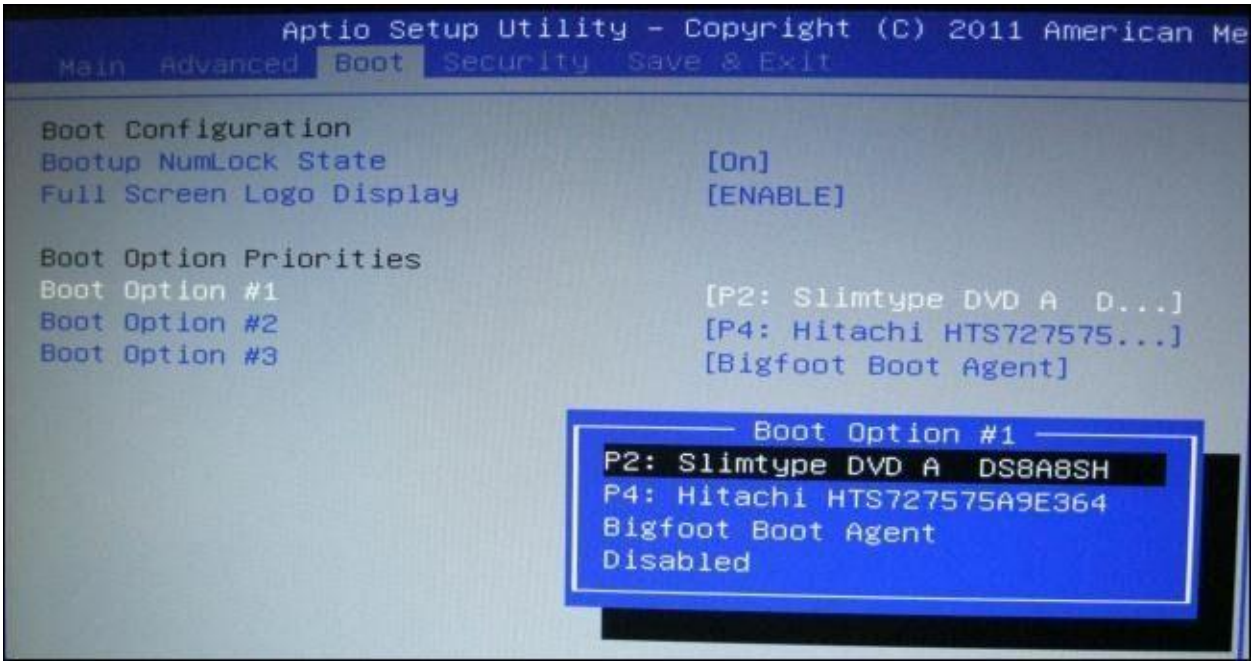
The BIOS or UEFI firmware loads configuration settings from a special place on the motherboard—traditionally, this was in memory backed up by a CMOS battery. If you change some low-level settings in your BIOS or UEFI settings screen, this is where your custom settings are stored.

The CPU runs the UEFI or BIOS, which tests and initializes your system's hardware—including the CPU itself. For example, if your computer doesn't have any RAM, it will beep and show you an error, stopping the boot process. This is known as the POST (Power On Self Test) process.

You may see the PC manufacturer's logo appear on your screen during this process, and you can often press a button to access your BIOS or UEFI settings screen from here. However, many modern

PCs fly through this process so fast that they don't bother displaying a logo and require accessing their UEFI setting screen from the Windows Boot Options menu.

The UEFI or BIOS Hands Off to a Boot Device



After its done testing and initializing your hardware, the UEFI or BIOS will hand off responsibility for booting your PC to your operating system's boot loader.

The UEFI or BIOS looks for a "boot device" to boot your operating system from. This is usually your computer's hard disk or solid-state drive, but may also be a CD, DVD, USB drive, or network location. The boot device is configurable from within the UEFI or BIOS setup screen. If you have multiple boot devices, the UEFI or BIOS attempts to hand off the startup process to them in the order they're listed. So, for example, if you have a bootable DVD in your optical drive, the system might try starting from that before it tries starting from your hard drive.

Traditionally, a BIOS looked at the MBR (master boot record), a special boot sector at the beginning of a disk. The MBR contains code that loads the rest of the operating system, known as a "bootloader." The BIOS executes the bootloader, which takes it from there and begins booting the actual operating system—Windows or Linux, for example.

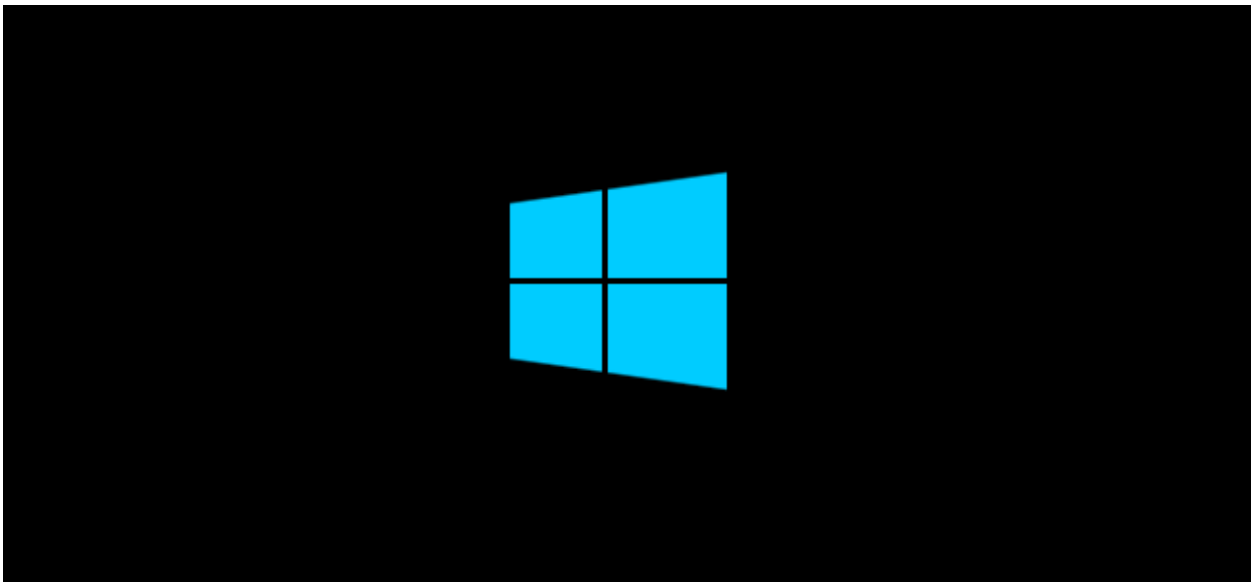
Computers with UEFI can still use this old-style MBR boot method to boot an operating system, but they normally use something called an EFI executable instead. These don't have to be stored at the beginning of a disk. Instead, they're stored on something called an "EFI system partition."

Either way, the principle is the same—the BIOS or UEFI examines a storage device on your system to look for a small program, either in the MBR or on an EFI system partition, and runs it. If there's no bootable boot device, the boot up process fails, and you'll see an error message saying so on your display.

On modern PCs, the UEFI firmware is generally configured for "Secure Boot." This ensures the operating system that it starts hasn't been tampered with and won't load low-level malware. If

Secure Boot is enabled, the UEFI checks whether the bootloader is properly signed before starting it.

The Bootloader Loads the Full OS



The bootloader is a small program that has the large task of booting the rest of the operating system. Windows uses a bootloader named Windows Boot Manager (Bootmgr.exe), most Linux systems use GRUB, and Macs use something called boot.efi.

If there's a problem with the bootloader—for example, if its files are corrupted on disk—you'll see a bootloader error message, and the boot process will stop.

The bootloader is just one small program, and it doesn't handle the boot process on its own. On Windows, the Windows Boot Manager finds and starts the Windows OS Loader. The OS loader loads essential hardware drivers that are required to run the kernel—the core part of the Windows operating system—and then launches the kernel. The kernel then loads the system Registry into memory and also loads any additional hardware drivers that are marked with "BOOT_START," which means they should be loaded at boot. The Windows kernel then launches the session manager process (Smss.exe), which starts the system session and loads additional drivers. This process continues, and Windows loads background services as well as the welcome screen, which lets you sign in.

On Linux, the GRUB boot loader loads the Linux kernel. The kernel also starts the init system—that's system on most modern Linux distributions. The init system handles starting services and other user processes that lead all the way to a login prompt.

This involved process is just a way of making everything load correctly by doing things in the correct order.

By the way, so-called "startup programs" actually load when you sign into your user account, not when the system boots. But some background services (on Windows) or daemons (on Linux and macOS) are started in the background when your system boots.

Programming Software

Coding or Programming software is the type of software that is not used by end-users. It is not for you unless of course, you are a programmer. Programming software are programs that are used to write, develop, test, and debug other software, including apps and system software. For someone who works at a software development company, for example, this type of software would make their life easier and efficient.

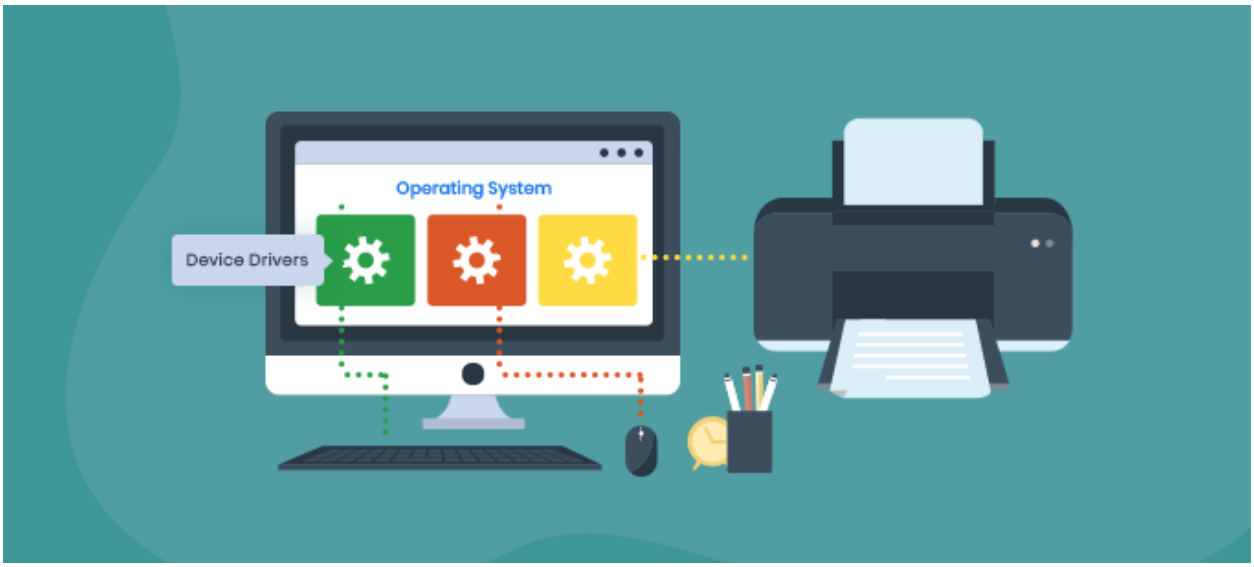
Programming software is used by software programmers as translator programs. They are facilitator software used to translate programming languages (i.e., Java, C++, Python, PHP, BASIC, etc) into machine language code. Translators can be compilers, interpreters and assemblers. You can understand compilers as programs that translate the whole source code into machine code and execute it. Interpreters run the source code as the program is run line by line. And assemblers translate the basic computer instructions – assembly code – into machine code.

Different programming language editors, debuggers, compilers and Integrated Development Environment (IDE) are an example of programming software. For example:

- Eclipse – a Java language editor
- VS code editor – programming language editor for different Operating Systems (OS).
- Notepad++ – an open-source editor for windows
- Sublime Text – A cross-platform code editor for Mac, Windows, and Linux

Driver Software

Driver software is often classified as one of the types of system software. They operate and control devices and peripherals plugged into a computer. Drivers are important because they enable the devices to perform their designated tasks. They do this by translating commands of an Operating System for the Hardware or devices, assigning duties. Therefore, each device connected with your computer requires at least one device driver to function.



Since there are thousands of types of devices, drivers make the job of your system software easier by allowing it to communicate through a standardized language. Some examples of driver software that you may be familiar with are:

- Printer Driver
- Mouse Driver
- Network Card

Usually, the operating system comes built-in with drivers for mouse, keyboard, and printers by default. They often do not require third-party installations. But for some advanced devices, you may need to install the driver externally. Moreover, if you use multiple operating systems like Linux, Windows, and Mac, then each of these supports different variants of drivers. For them, separate drivers need to be maintained for each.

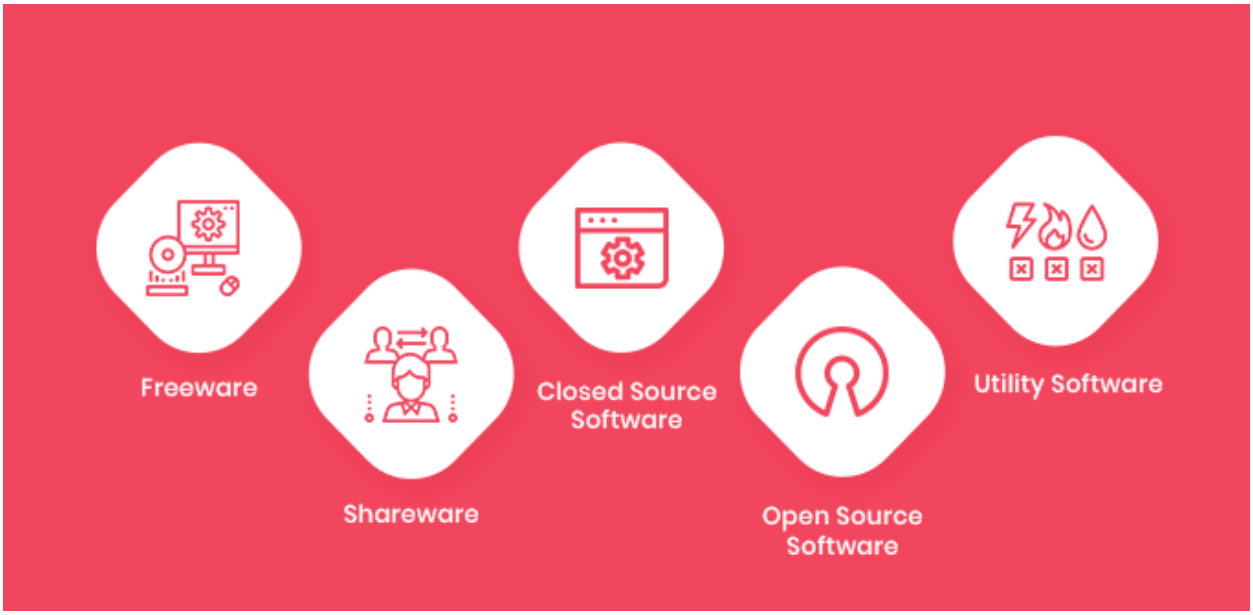
Five Additional Different Types of Software That You Might Be Familiar With

Now that we have discussed the major types of software and now you must be wondering about the software you use most frequently. For example, trendy social media software applications like Snapchat and Instagram or photo editing apps like PhotoShop and Snapseed. You must be thinking about what category they fall under. Can they only be classified as a broad category of Application software or are there better ways to describe them? The answer – yes, other than the major types of computer software there are various subcategories of software.

Let's discuss five additional subcategories of software and understand those using examples of trendy software.

These are:

- Freeware
- Shareware
- Open Source Software
- Closed Source Software
- Utility Software



Freeware

Freeware software is any software that is available to use for free. They can be downloaded and installed over the internet without any cost. Some well-known examples of freeware are:

- Google Chrome
- Skype
- Instagram
- Snapchat
- Adobe reader

Although they all fall under the category of Application or end-user software, they can further be categorized as freeware because they are free for you to use.

Shareware

Shareware, on the other hand, are software applications that are paid programs, but are made available for free for a limited period of time known as 'trial period'. You can use the software

without any charges for the trial period but you will be asked to purchase it for use after the trial ends. Shareware allows you to test drive the software before you actually invest in purchasing it. Some examples of Shareware that you must be familiar with are:

- Adobe PhotoShop
- Adobe Illustrator
- Netflix App
- MatLab
- McAfee Antivirus

Open Source Software

This is a type of software that has an open-source code that is available to use for all users. It can be modified and shared to anyone for any purpose. Common examples of open source software used by programmers are:

- LibreOffice
- Arduino IDE
- PHP
- GNU Image Manipulation Program (GIMP)

Closed Source Software

These are the types of software that are non-free for the programmers. For this software, the source code is the intellectual property of software publishers. It is also called 'proprietary software' since only the original authors can copy, modify and share the software. Following are some of the most common examples of closed-source software:

- .Net
- Java
- Android
- Microsoft Office
- Adobe PhotoShop

Utility Software

Utility software is considered a subgroup of System software. They manage the performance of your hardware and application software installed on your computer, to ensure they work optimally. Some features of utility software include:

- Antivirus and security software
- File compressor
- Disk cleaner
- Disk defragmentation software
- Data backup software

INTERNET

Brief History

It was 1957 and the Cold War was sub-zero. The Soviet Union launched the first satellite, Sputnik, shaking the confidence of the United States in its scientific and technology leadership. In response, President Eisenhower created the Advanced Research Projects Agency (ARPA) within the Department of Defense (DoD) to get the U.S. into space (and to boost military missile development). That role was replaced by NASA, and ARPA redefined itself as a sponsor of advanced research projects at various universities and contractors.

In late 1960, Paul Baran of the RAND Corporation wrote a series of technical papers for the Pentagon analyzing the vulnerability of communications in case of a Soviet nuclear attack. A part of the papers were two ideas that would have far greater impact than anyone imagined. Baran said that military command messages and control signals should be carried over a distributed network that would have redundant connections in case a missile took out part of the system. The best way to do this would be to break each message into blocks and send each block separately over that network, avoiding any parts that aren't working.

A specific distributed military network was never built, but a few years later ARPA began looking for a way its members could distribute messages and data among themselves so they could take advantage of each other's research. They came up with the idea of a distributed network called ARPAnet, built around something called interface message processor (IMP), which connected computers at university research centers. IMP also incorporated a technology called TCP/ IP, developed at the National Science Foundation. Standing for transmission control protocol/Internet protocol, TCP/IP calls for breaking up messages and data into packets to which addressing information, error correction code, and identification are added. The packets can all travel to their destinations over the distributed network, and a computer on the other end checks for mistakes and pieces them together in the right order. In 1969, computers at universities all over the country were linked to ARPAnet. The first letter sent over the new network was an "L" sent from UCLA to the Stanford Research Institute.

The ARPAnet continued to expand until, in 1972, it connected 23 host sites. By 1975, one new installation was being added each month. Meanwhile, other types of networks, such as the Computer Science Network (CSNET), designed to be a less-expensive version of ARPAnet, sprang up across the country. In 1974, researcher Bob Kahn and Vint Cerf came up with the idea of a "network of networks" that would let dissimilar networks communicate with one another. By 1982, different networks were adopting TCP/IP as their communications standard, and the term "Internet" was used for the first time. A year later, a gateway was set up between CSNET and ARPAnet using TCP/IP as a common standard so people on the two networks could communicate with each other.

A high-speed (56Kbps) backbone was built by the NSF to connect five supercomputing centers. Because all the transmission time wasn't used, the NSF agreed to let local networks connect to each other through the backbone. The Internet was born, even if people didn't realize it yet. For years, the Internet was the territory of colleges and defense contractors. As the Internet grew,

many of the people who nursed it through its infancy were dismayed when, in 1991, the NSF lifted restrictions on the commercial use of the Net.
The first time someone sent out advertising over the Internet—a practice destined to earn the name spam—elicited reactions among Internet purists that were angry and vocal. And equally futile. The Internet and the World Wide Web, a section of the Internet developed to lift it out of its text-only origins and into the world of graphics, sound, and video, had lives of their own. The imp was out of the bottle, and today what started as a modest experiment in communicating is growing at a rate of 100% to 200% a year.

Looking back over the last half century, you find that in no other part of computer domain, is it easier to see evolution in technology than in the Internet. The evolution of other parts of computing are hidden in distant origins, in the microscopic scale at which some mutations take place, and in the mind of engineers better at creating technology than explaining it. But the Internet is something we've experienced directly, even when it was still a gleam in the eye of the technocrats who created it.

In 1964, the Internet was still gestating, but we already had networks linking computers at a single location. A year later, networks spread to connect different locations. For several years, the infant Internet was the province of scientists and universities. In 1969, the first commercial online service, CompuServe, started up, but it wouldn't be until 1981, when the first modem was marketed, that the concept of communicating with anyone anywhere through computers became a phenomenon. That first 300 BAUD modem was so painfully slow that you could speak out loud each of the letters of a message as it popped onscreen and never fall behind. But that didn't matter. You could converse, download programs, play games, and send email—all in real time. We had seen the Internet, and there was no holding it back. The world has shrunk several times since then.

Now the Internet is part of our everyday life. It's how we stay in touch, entertain and educate ourselves, get instant information, shop, and solve problems no one person could solve alone. We've seen it grow, but that's been a lot like seeing one of those animations that illustrate millions of years of evolution by showing an amoeba morphing from a one-celled organism through fish and amphibians, through dinosaurs and birds, through mammals and apes to produce humans.

Radio-Newspaper Receiver for Home Use

DESIGNED to fit the top of a commercial table receiver which it matches in cabinet style, a complete radio-newspaper receiver for home use has just been placed on the market. All necessary apparatus for receiving and printing news bulletins and pictures transmitted over the air are contained in the unit. The news is automatically printed on a continuous sheet of paper that unwinds from a roll as it is received. The instrument can be used in conjunction with any radio receiver, the manufacturer declares, provided it has an output of at least five watts.

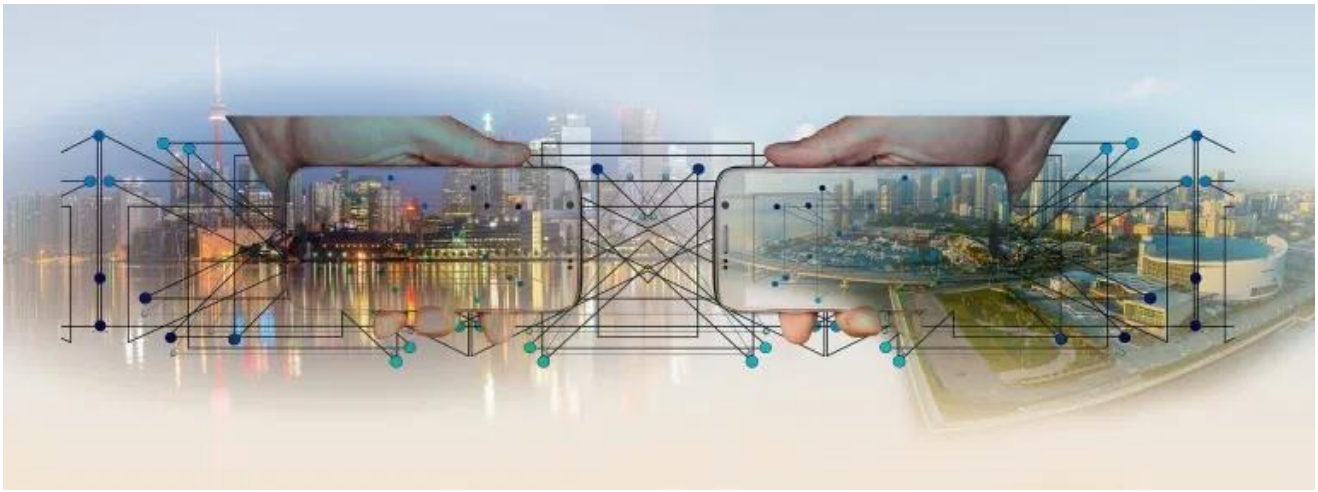
Anyone can now own one of these home-model, radio-newspaper receivers

The Internet's missing link.

A story from the buried history of the Internet describes a device that, when connected to a radio, prints news bulletin and pictures.

Now we will try to understand what is INTERNET and how it works!

Definition: The internet is a world-wide network of computers linked together by telephone wires, satellite links and other means. For simplicity's sake we will say that all computers on the internet can be divided into two categories: servers and browsers.



Servers are where most of the information on the internet "lives". These are specialized computers which store information, share information with other servers, and make this information available to the general public.

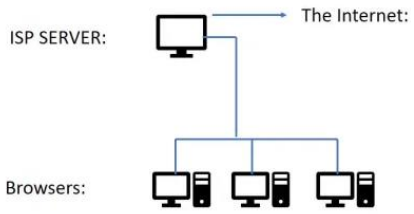
Browsers are what people use to access the World Wide Web from any standard computer.

"When you connect your computer to the internet, you are connecting to a special type of server which is provided and operated by your Internet Service Provider (ISP). The job of this "ISP Server" is to provide the link between your browser and the rest of the internet. A single ISP server handles the internet connections of many individual browsers - there may be thousands of other people connected to the same server that you are connected to right now."

Servers are place where relevant data is stored.

ISP is a company that provides individuals and other companies' access to the Internet and other related services.

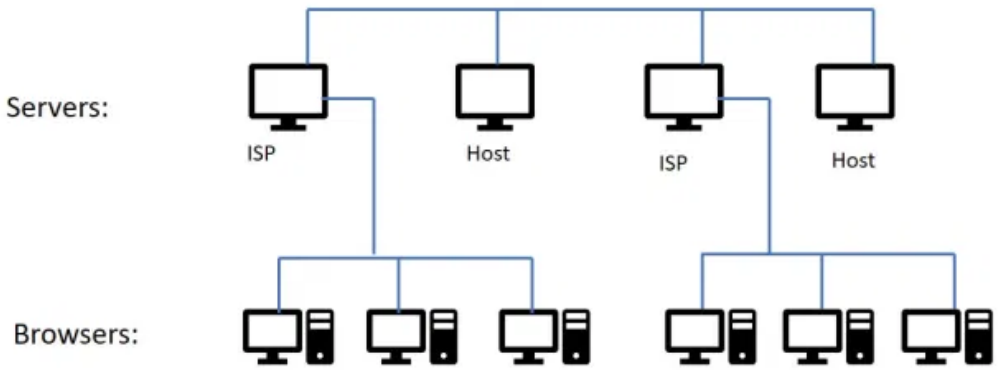
The following picture shows a small "slice" of the internet with several home computers connected to a server:



ISP servers receive requests from browsers to view webpages, check email, etc. Of course each server can't hold all the information from the entire internet, so in order to provide browsers with the pages and files they ask for, ISP servers must connect to other internet servers. This brings us to the next common type of server: the "Host Server".

Host servers are where websites "live". Every website in the world is located on a host server somewhere (Sites like Godaddy, Hostinger etc provides us platform to host our website i.e. our data is stored in their server). The host server's job is to store information and make it available to other servers.

The picture below show a slightly larger slice of the internet:



To view a web page from your browser, the following sequence happens:

1. You either type an address (URL) into your "Address Bar" or click on a hyperlink.
2. Your browser sends a request to your ISP server asking for the page.
3. Your ISP server looks in a huge database of internet addresses and finds the exact host server which houses the website in question, then sends that host server a request for the page.
4. The host server sends the requested page to your ISP server.
5. Your ISP sends the page to your browser and you see it displayed on your screen.

Dr. Cecilia Aragon, director of the Human-Centered Data Science Lab at the University of Washington in Seattle, one of the top-ranked computer science programs in the country, explains the steps.

"THE BROWSER SETS UP A CONNECTION BETWEEN YOU AND SOMEBODY ELSE, SOME OTHER SERVER SOMEWHERE ELSE IN THE WORLD."

"Some may get lost and be resent again, but that goes through a number of routers back and forth until it gets back to your ISP and then your ISP sends the packets back to the router through WIFi to a port on your computer and then your browser, which is listening for those packets, takes the data and displays them in a form you as a human can understand," Aragon says. "And that is incredibly oversimplified."

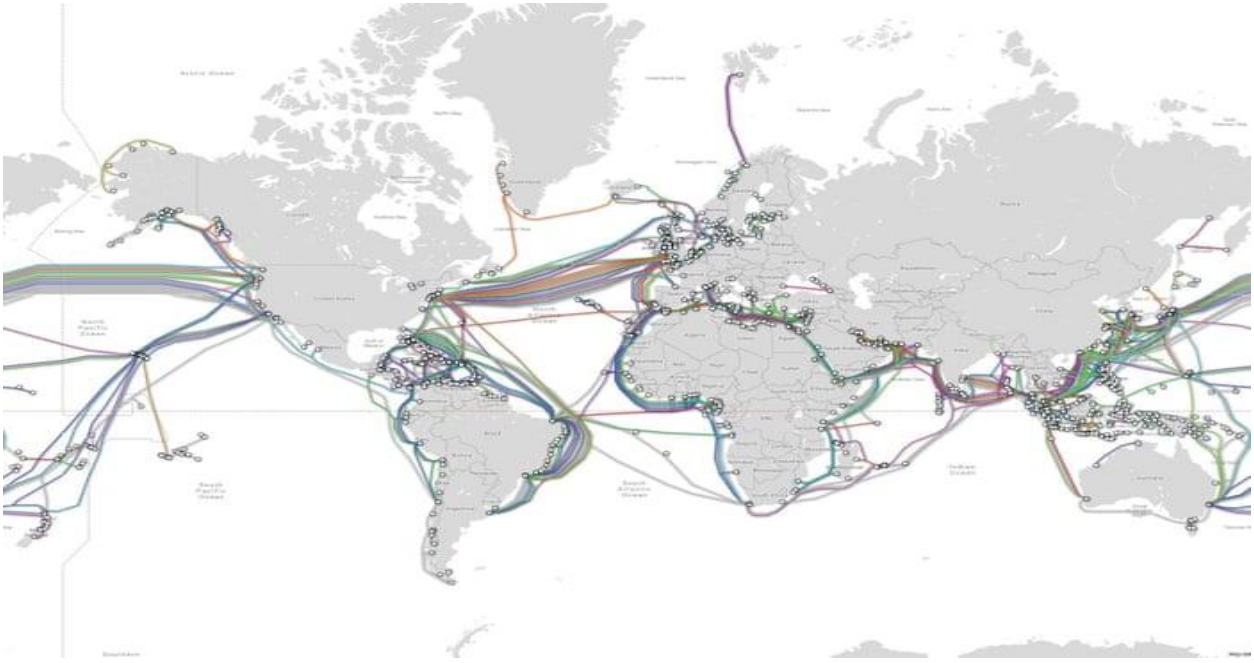
How big is the internet?

One measure is the amount of information that courses through it: about five Exabyte a day. That's equivalent to 40,000 two-hour standard definition movies per second.

It takes some wiring up. Hundreds of thousands of miles of cables crisscross countries, and more are laid along sea floors to connect islands and continents. About 300 submarine cables, the deep-sea variant only as thick as a garden hose, underpin the modern internet. Most are bundles of hair-thin fiber optics that carry data at the speed of light.

The cables range from the 80-mile Dublin to Anglesey connection to the 12,000-mile Asia-America Gateway, which links California to Singapore, Hong Kong and other places in Asia. Major cables serve a staggering number of people. In 2008, damage to two marine cables near the Egyptian port of Alexandria affected tens of millions of internet users in Africa, India, Pakistan and the Middle East.

Last year, the chief of the British defense staff, Sir Stuart Peach, warned that Russia could pose a threat to international commerce and the internet if it chose to destroy marine cables.



What is the World Wide Web?

The web is a way to view and share information over the internet. That information, be it text, music, photos or videos or whatever, is written on web pages served up by a web browser.

Google handles more than 40,000 searches per second, and has 60% of the global browser market through Chrome. There are nearly 2bn websites in existence but most are hardly visited. The top 0.1% of websites (roughly 5m) attract more than half of the world's web traffic.

Among them are Google, YouTube, Facebook, the Chinese site Baidu, Instagram, Yahoo, Twitter, the Russian social network VK.com, Wikipedia, Amazon and a smattering of porn sites. The rise of apps means that for many people, being on the internet today is less about browsing the open web than getting more focused information: news, messages, weather forecasts, videos and the like.

Complete this book, take free course at:

https://electrovertlabs.com/courses/thinking-machine-part-1/

Give quiz and download Completion certificate. Passing marks 75%

Give us feedback at: contact@electrovertlabs.com

Feedbacks and suggestions are highly appreciated.