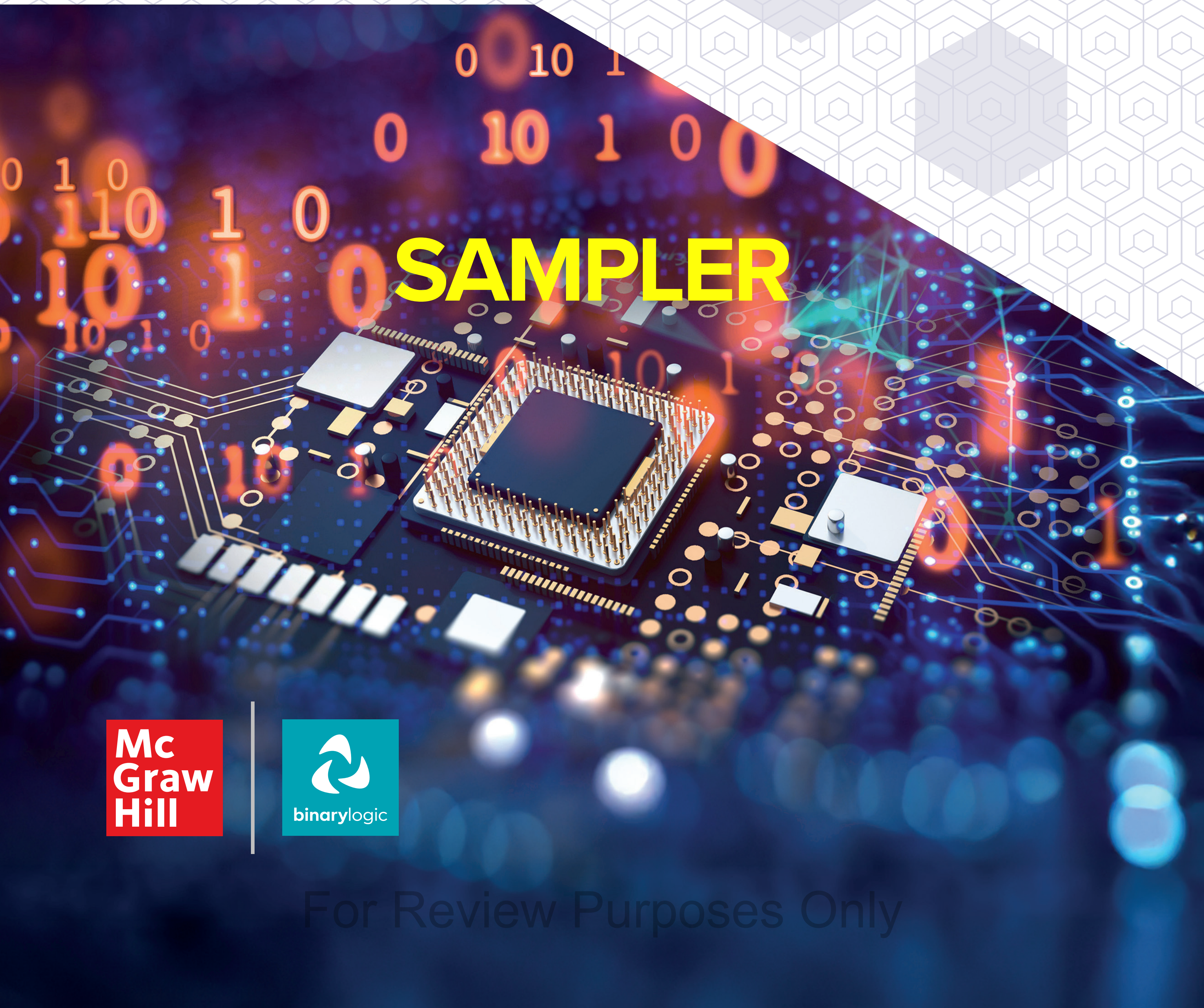
Foundations of Technology

Computer Science



SAMPLER

McGraw Hill | binarylogic

Foundations of Technology

Computer Science

Foundations of Technology
Computer Science

Printed and distributed by McGraw Hill in association with Binary Logic SA.

MHID: 1264929242 ISBN: 9781264929245

mheducation.com binarylogic.net

Contents

Key Features

> An innovative approach to building digital competencies, developed by expert educators.

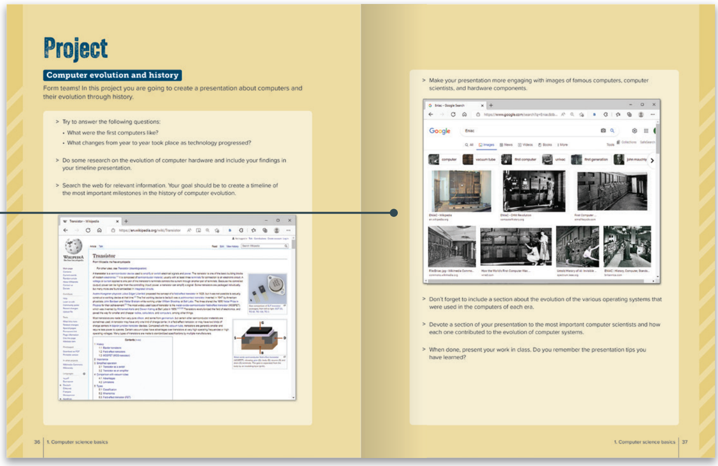
> Curriculum aligns with the latest industry standards, preparing students for certifications and future careers.

> Well-defined learning goals and hands-on, applicable digital skills.

Each unit offers straightforward explanations and contemporary examples, making technology concepts accessible and relevant.

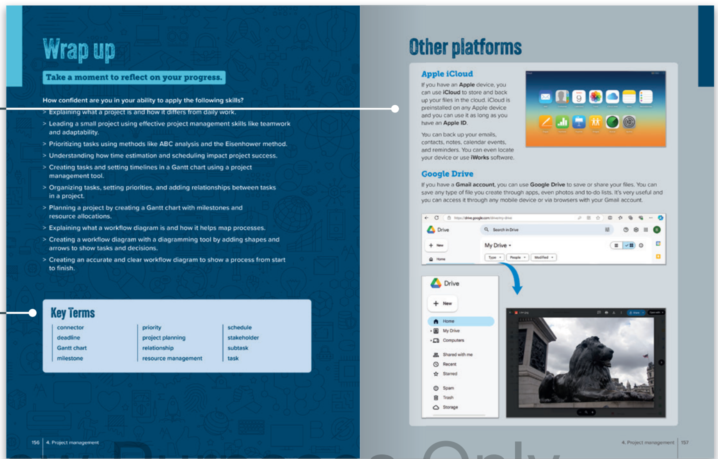
Every unit includes a variety of tasks and activities designed to help students build essential digital competencies.

Projects and exercises throughout the course reinforce students' understanding and practical application of digital skills in real-world scenarios.



Students learn about platform diversity expanding their digital toolkit and adaptability.

Each unit organizes key terms that are crucial for digital literacy, equipping students for today's technology-driven workplace.



1

UNIT 1

Computer science basics

Understanding how computers work is important because they play a significant role in daily life, including activities such as homework, entertainment, and communication. This unit covers how computers store and process data, how they communicate over networks, and how key components like logic gates and operating systems contribute to their efficient operation.

Learning Objectives

In this unit, you will:

- identify the difference between decimal, binary, and hexadecimal systems, and how to convert between them.
- represent numbers, text, and images in binary using the ASCII table.
- use Boolean logic and logic gates to process binary data.
- understand what the role of transistors and integrated circuits in computers is.
- understand what the von Neumann architecture is.
- how to describe the fetch-execute cycle and its importance in computer operation.
- compare memory and storage devices and their roles in a computer.
- evaluate storage performance by calculating access times, latencies, and transfer rates.
- understand what an operating system is and why it is essential for running software.
- clarify the OS's role in managing memory and processes.
- identify what the OS does to organize and store files on a hard drive.
- illustrate data transmission through packet switching.
- understand what the role of IP addresses and hostnames in communication is.
- outline the functions of each layer in the OSI model and their impact on networking.
- learn about the importance of firewalls and setting them up.
- explore how jobs are affected by technology.
- recognize the importance of lifelong learning in staying current with technological changes.

Tools

- Microsoft Windows

LESSON 1

Data manipulation

Modern digital computers can be found practically anywhere around us. We have desktop computers at home, at school, and at work, powerful laptops that are easily transported from place to place, and even smartphones, which are fully-fledged computers that we can carry around in our pockets.

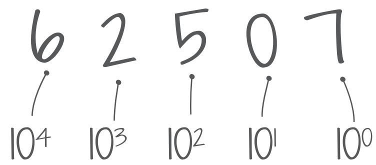
Decimal system -Binary system

Since computers run on electricity, all of the internal components can only "understand" two states: they are either in a **low-voltage state** or in a **high-voltage state**. All modern computers are what we call **binary** machines. This means that the "language" that they use internally in order to function is the binary numeral system, which is a way to write numbers using only two digits: **0** (low-voltage state) and **1** (high-voltage state).

By using a series of 0s and 1s, we can create all the other numbers. In the decimal system, which people normally use, each digit can take one of ten values (0-9). When digits are put together to form a number, the place of each digit has a different place value, increasing by a power of ten.

To represent the number 131 in the decimal system:

Digits	1	3	1	
Place value	10^2 =100	10^1 =10	10^0 =1	
	1×100 +	3×10 +	1×1 =	131

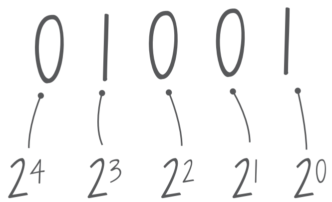


The same principle is used in the **binary system**. The difference is that each digit can take one of two values (0, 1) and each place value increases by a power of two (ones, twos, fours, eights, etc.).

Now let's try to make a fill-in form. The steps you must follow are:

Digits	1	0	0	0	0	0	1	1	
Place value	2^7 =128	2^6 =64	2^5 =32	2^4 =16	2^3 =8	2^2 =4	2^1 =2	2^0 =1	
	1×128 +	0×64 +	0×32 +	0×16 +	0×8 +	0×4 +	1×2 +	1×1 =	131

Notice that the place value of the rightmost digit in either system is **1**. Any number (except zero) to the zero power equals one. Consequently, 10^0=2^0=1. You can now read and understand any number in the binary system!



Smart Tip

In computers, the basic unit of information is called a bit and it can either be 0 or 1. The word comes from a contraction of "binary digit".

Hexadecimal system

As computers got more powerful, they were able to work with more and more bits of information. The **hexadecimal numeral system** was employed by people working with computers in an attempt to shorten the very long binary series. The hexadecimal system is a base-16 number system, meaning that each digit can take any of 16 distinct values. Of course, in this case, we need symbols to represent the values 10, 11, 12, 13, 14, and 15. So, we use the letter A to represent the number 10, B to represent 11, C for 12, etc.

The 16 digits in base-16 are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

In the hexadecimal system, the place value increases by a power of sixteen (ones, sixteens, two hundred and fifty-sixes, four thousand and ninety-sixes, etc.).

So, let's convert the hexadecimal number 1C8A to decimal:

Digits	1	C	8	A	
Place value	16^3 =4096	16^2 =256	16^1 =16	16^0 =1	
	1×4096 (=4096) +	12×256 (=3072) +	8×16 (=128) +	10×1 (=10) =	7306

Conversions between systems

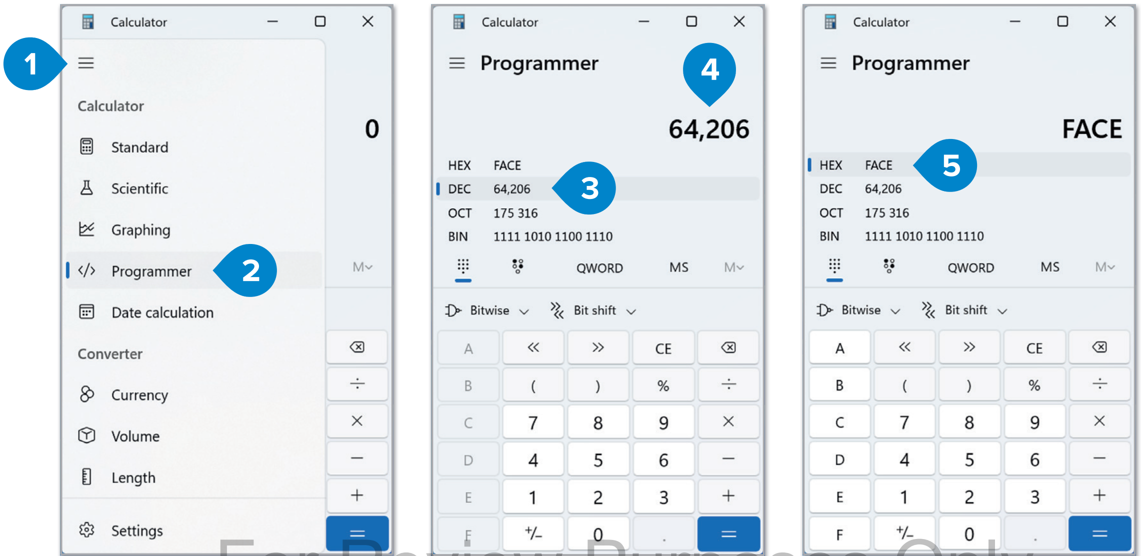
It is very easy to convert numbers from one system to another using Windows Calculator.

To convert a number to a hexadecimal number:

> Open **Calculator**.
> Click **Open Navigation** 1 and click **Programmer** mode. 2
> Click a number system (e.g., **DEC**). 3
> Type a number into that number system. 4
> Click another number system to convert your number to. 5

Since the hexadecimal system contains some letters, computer programmers like to create "magic numbers" that spell words and use them in their programs.

For example, DEADBEEF ("dead beef") is often used to indicate a software crash and 8BADF00D ("ate bad food") is used by Apple in iOS when an application crashes.



Data representations

Every computer works by manipulating data in various ways. Because computers can only store and manipulate data using binary digits, we need ways to represent many different types of data like numbers, text, images, and video in binary.

As for numbers, we have already discussed how we can represent decimal positive integers using the binary system. To work with negative or real numbers, there are various representation systems, which help us use any number through a string of 0s and 1s.

To represent **text** in computers, we use character sets. A **character set** is simply a list of characters and the binary code used to represent each one. One of the most used character sets is the **ASCII character set**. ASCII stands for American Standard Code for Information Interchange.

Positive integers are whole numbers greater than zero: (1, 2, 3, 4...). Real numbers are all numbers below or above zero, including fractions. A negative number is a real number less than zero.

The first 32 characters in the ASCII character chart do not have a simple character representation that you can print to the screen. These characters are reserved for special purposes such as the Enter and Tab characters in a text file.

The codes in this chart are expressed as decimal numbers, but these values get translated to their binary equivalent for storage in the computer.

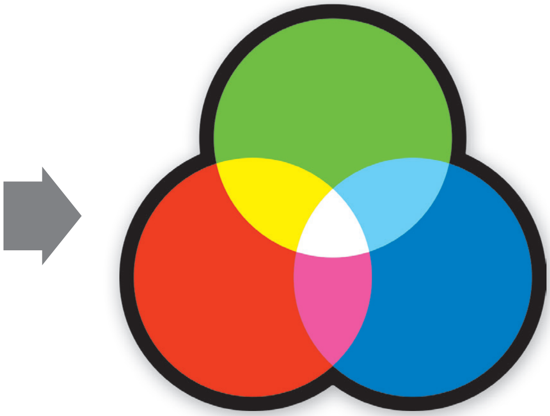
ASCII character reference table with hexadecimal and decimal values

Dec	Hex	Char	Action (if non-printing)	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	NUL	(null)	32	20	Space	64	40	@	96	60	`
1	1	SOH	(start of heading)	33	21	!	65	41	A	97	61	a
2	2	STX	(start of text)	34	22	"	66	42	B	98	62	b
3	3	ETX	(end of text)	35	23	#	67	43	C	99	63	c
4	4	EOT	(end of transmission)	36	24	$	68	44	D	100	64	d
5	5	ENQ	(enquiry)	37	25	%	69	45	E	101	65	e
6	6	ACK	(acknowledge)	38	26	&	70	46	F	102	66	f
7	7	BEL	(bell)	39	27	'	71	47	G	103	67	g
8	8	BS	(backspace)	40	28	(	72	48	H	104	68	h
9	9	HT	(horizontal tab)	41	29	)	73	49	I	105	69	i
10	A	LF	(NL linefeed, new line)	42	2A	*	74	4A	J	106	6A	j
11	B	VT	(vertical tab)	43	2B	+	75	4B	K	107	6B	k
12	C	FF	(NP linefeed, new page)	44	2C	,	76	4C	L	108	6C	l
13	D	CR	(carriage return)	45	2D	-	77	4D	M	109	6D	m
14	E	SO	(shift out)	46	2E	.	78	4E	N	110	6E	n
15	F	SI	(shift in)	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	(data link escape)	48	30	0	80	50	P	112	70	p
17	11	DC1	(device control 1)	49	31	1	81	51	Q	113	71	q
18	12	DC2	(device control 2)	50	32	2	82	52	R	114	72	r
19	13	DC3	(device control 3)	51	33	3	83	53	S	115	73	s
20	14	DC4	(device control 4)	52	34	4	84	54	T	116	74	t
21	15	NAK	(negative acknowledge)	53	35	5	85	55	U	117	75	u
22	16	SYN	(synchronous idle)	54	36	6	86	56	V	118	76	v
23	17	ETB	(end of trans. block)	55	37	7	87	57	W	119	77	w
24	18	CAN	(cancel)	56	38	8	88	58	X	120	78	x
25	19	EM	(end of medium)	57	39	9	89	59	Y	121	79	y
26	1A	SUB	(substitute)	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	(escape)	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	(file separator)	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	(group separator)	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	(record separator)	62	3E	>	94	5E	^	126	7E	~
31	1F	US	(unit separator)	63	3F	?	95	5F	_	127	7F	DEL

To store **images**, we need to represent the color of each pixel in an image. The most common way is to use the **RGB model** in which each color is the sum of the different shades of the three primary colors (Red, Green, and Blue). So, for each pixel, we actually store three values, one for each color, each ranging from 0 to 255, which indicates the shade of each color from black to pure red, for example. Thus, an image is the binary representation of three colors that make up the pixels of the picture.

Monitor color analysis

Monitor color	R	G	B
White	255	255	255
Red	255	0	0
Green	0	255	0
Blue	0	0	255
Cyan	0	255	255
Magenta	255	0	255
Yellow	255	255	0
Black	0	0	0



Video is the most complex data type to represent but generally it can be thought of as a series of images, saved in binary and played back one after the other. These images are usually compressed in order to save storage space and process the images as fast as possible.

Boolean logic and logic gates

The data is in binary format, but computers have to convert the data for it to do something useful.

Boolean logic, named after its creator, mathematician George Boole, provides us with a set of operators that do simple transformations and comparisons of data. Just like in decimal arithmetic that humans use, where we have operations like addition, subtraction, multiplication, and division, in Boolean logic, we have **NOT**, **AND**, **OR,** and **XOR**. With these simple operators, we can build complex systems, as complex as a computer CPU!

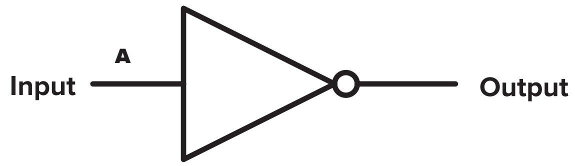
In computers, each basic Boolean operation is implemented by a **logic gate**, which is a device that accepts one or more input signals and produces a single output signal.

Each gate is associated with what is called a truth table. A truth table lists all possible input values and the corresponding output values for a specific gate.

Logic Gate NOT

A **NOT** gate accepts one input value and produces one output value. The NOT operator inverts the input. If the input is 0, the output is 1 and if it is 1, the output is 0.

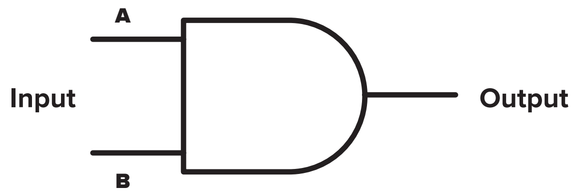
Input	Output
A	**NOT** A
0	1
1	0



Logic Gate AND

An **AND** gate accepts two input values, which both determine the output. If the two input values are both 1, the output is 1; otherwise, the output is 0.

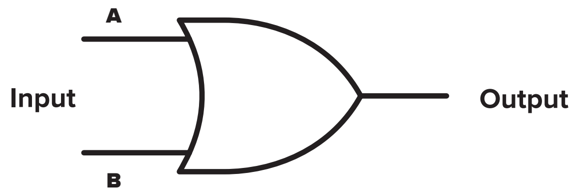
Input		Output
A	B	A **AND** B
0	0	0
0	1	0
1	0	0
1	1	1



Logic Gate OR

Like the AND gate, an **OR** gate has two inputs. If the two input values are both 0, the output value is 0; otherwise, the output is 1.

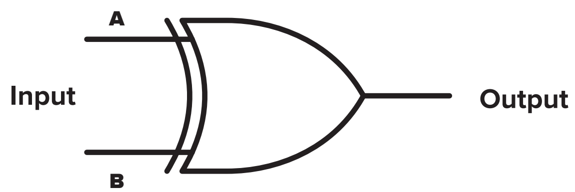
Input		Output
A	B	A **OR** B
0	0	0
0	1	1
1	0	1
1	1	1



Logic Gate XOR

An **XOR**, or exclusive OR, gate produces 0 if both inputs are the same, and 1 if they are different.

Input		Output
A	B	A **XOR** B
0	0	0
0	1	1
1	0	1
1	1	0



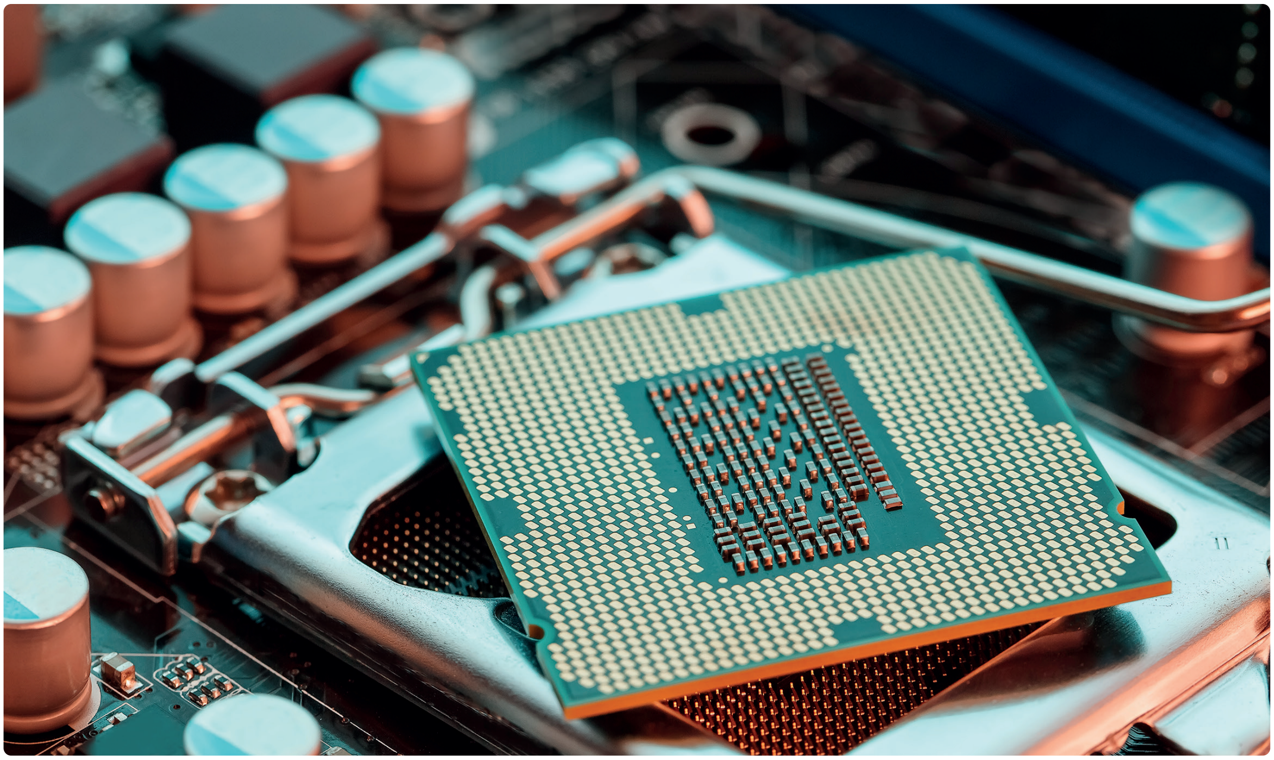
Smart Tip

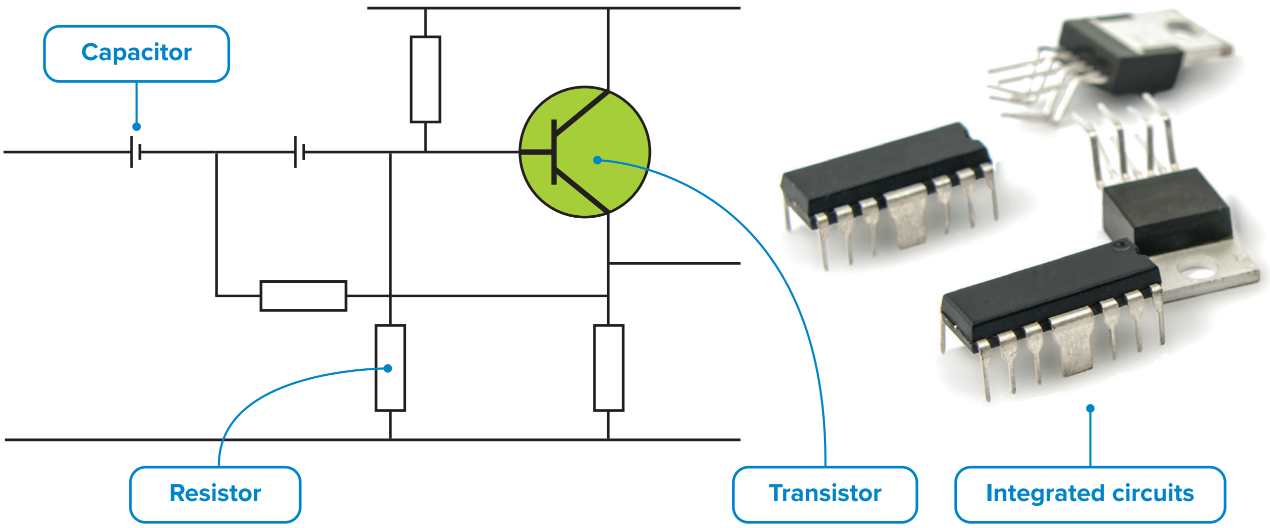
Did you know that you can use Boolean operators to refine your web searches? Using the NOT operator (-) you can exclude results that contain a particular word. For example, searching [jaguar speed -car] will give you the speed of the animal excluding references to the car with the same name. Also, by default every space in your search query is considered to be an AND operator, meaning that you want the results to contain all the words of your query.

Transistors

Gates are made of one or more transistors. A **transistor** is a device that acts, depending on the voltage level of an input signal, either as a wire that conducts electricity or as a resistor that controls the flow of electricity.

Gates are used in all computer components, from the RAM modules to the little external flash memory sticks. These flash memory sticks use a special combination of gates that allows them to retain their state so that data can be stored permanently, with no need for electrical power to keep them saved after you remove the drive from your computer.





Integrated circuits

By combining multiple gates together and, in some cases, with other electrical elements like resistors and capacitors, we can create electronic circuits with multiple inputs and multiple outputs.

An **integrated circuit** (also called a chip or microchip) is a set of electronic circuits all enclosed in a small package. Microchips have revolutionized modern electronics design because of their very small size, low energy consumption and very high potential to carry out complex tasks extremely fast.

The 0s and 1s are, as we said, electrical signals: low-voltage or no-voltage for 0 and high-voltage for 1.

Evolution of CPU transistor counts over the decades			
Year	**Processor Unit**	**Transistor Count**	**Designer**
1971	CPU	2,300	Intel
1982	CPU	40,000	NEC
2012	CPU	5,000,000,000	Intel
2023	CPU	134,000,000,000	Apple
1997	GPU	3,500,000	NVIDIA
2012	GPU	7,080,000,000	NVIDIA
2024	GPU	104,000,000,000	NVIDIA

Hands on!

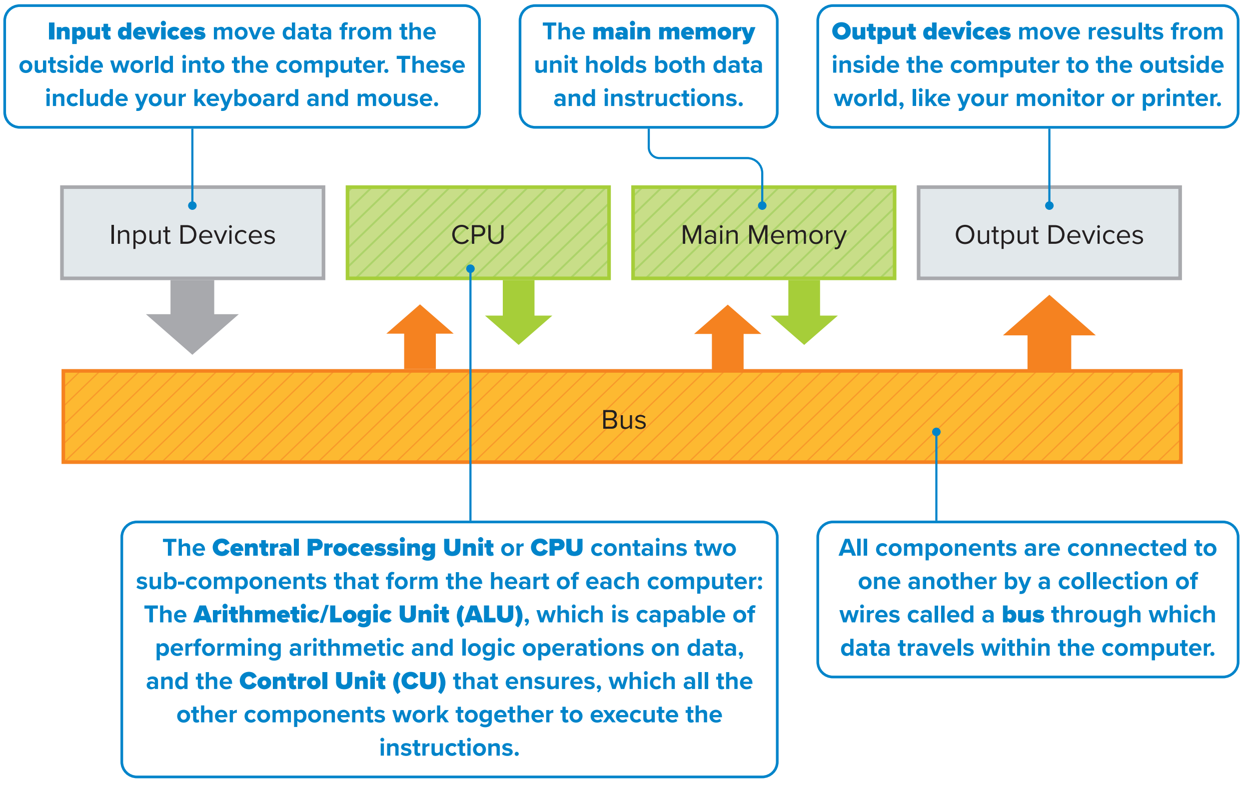
1. Let's try to convert from one system to another.
   What is the process for converting decimal 71 to its equivalent representation in the binary numeral system?
   How can the hexadecimal number 2B3 be systematically converted to its corresponding value in the decimal numeral system?
2. Based on the ASCII character set, represent the characters "M", "n", and "@" in the decimal and binary systems.

LESSON 2

Computer architecture

All computers can perform three fundamental actions: storing, retrieving, and processing data. Of course, in order for computers to be useful, we need to tell them what to do with the data and we need to provide instructions. Those instructions need to be stored somewhere that the computer can find and retrieve them, and because computers are binary machines, instructions are, of course, binary sequences. Data and instructions on how to process the data are logically the same and can be stored in the same place. Another major characteristic of computers is that the units that process information are separate from the units that store information.

These basic traits form the **von Neumann architecture** on which all modern computers are based. In the von Neumann architecture, we can distinguish some discrete components that work together in order for the computer to function.



History

John von Neumann described the computer architecture of the same name together with other engineers during the development of ENIAC in 1945. He was a brilliant Hungarian mathematician with many contributions to various fields such as mathematics, physics, and computer science.

The fetch-execute cycle

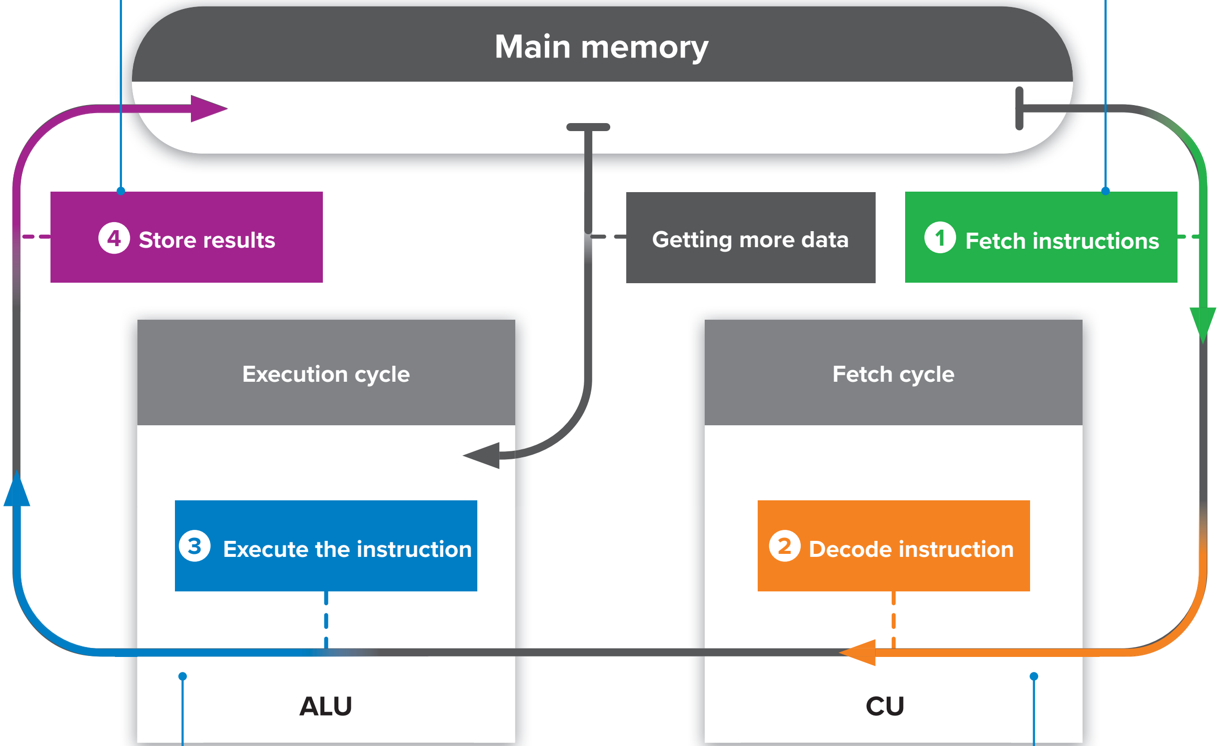
Now that you are familiar with the main architecture of a computer, let's find out how the instructions are executed and process data. This is also known as the **fetch-execute cycle**. Remember that both data and instructions are stored in the main memory.

The steps of the fetch-execute cycle can be diagrammed:

Store results
The results of the execution of the instructions are stored in the main memory.

Get more data if needed
Depending on the instruction, additional data from the memory may need to be loaded to other registers for processing. For example, if the instruction says to add a number stored in memory to a register, the control unit must get that number from the main memory.

Fetch the next instruction
The CPU retrieves the next instruction to be executed from the main memory and stores it in a special type of local memory space inside the CPU which is called a register.



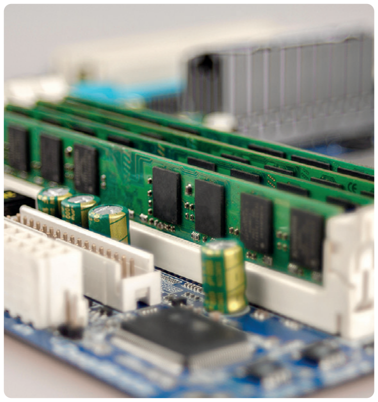
Execute the instruction
Once the instruction is decoded and all required data fetched, the ALU carries out the execution of the instruction, most usually involving operations on various registers. Then, the cycle begins again.

Decode the instruction
The control unit (fetch cycle) interprets the current instruction which is in the instruction register and transforms it into a control signal that can be understood by the ALU.

Smart Tip

ROM, which stands for Read Only Memory, is another type of memory. Like RAM, it provides random access to data but is non-volatile. The instructions that are stored inside it are permanent —they cannot be erased and re-written, only read. That is why ROM is used to store the instructions that the computer needs to start itself up. These instructions are called Firmware.

Main memory



The type of main memory that all computers use is called **RAM,** which stands for **Random Access Memory**.

This means that each byte of data inside the memory can be directly accessed and altered. RAM needs to be very fast so that it can support rapid access of data and instructions needed by the CPU. RAM is also volatile memory, meaning that its contents are only maintained for as long as the computer is powered on. In contrast, there are memory modules that are non-volatile and can maintain their contents even without a constant power supply. A common example of such memory is the flash memory on USB memory sticks. Here, the data is stored after we unplug the device from a computer, but access to data is much slower.

Secondary storage

As we already mentioned, the main memory is volatile and limited in size. So, we need another type of storage device where data and instructions can be kept safely when they are no longer being processed or when the computer is turned off. These other types of devices are called secondary storage devices and the most well-known include the hard disk drive and the CD/DVD drive. Because data must be read from them and written to them, secondary storage devices are considered both input and output devices in the von Neumann architecture model.

Hard disk drives

A desktop computer can have one or multiple hard disk drives connected to it. The main principle of the operation of a hard disk is a read/write head that travels across a spinning magnetic disk, retrieving or storing data. The surface of each magnetic disk is logically organized into tracks and sectors. A **track** is a concentric circle on the surface of the disk and each track is divided into sectors. Each **sector** contains a **block** of data as a continuous sequence of bits.

The read/write head of a disk drive is positioned on an arm that moves from one track to another. To read or write something, a hard disk needs an input/output instruction specifying a track and a sector. When the read/write head is over the proper track, it waits for the appropriate sector to be positioned underneath it as the disk spins and then the corresponding block of data is accessed. This process of how a hard disk reads or writes data results in four measures of a hard disk's efficiency: **seek time**, **latency**, **access time**, and **transfer rate**.

As you can understand, different applications have different needs. For example, a database system needs fast access times as it constantly reads and writes thousands of records that are positioned all over the disk. On the other hand, when we play HD video, we need the hard drive to provide a high transfer rate because there is a lot of data in each second of video.

Seek time is the time it takes for the read/write head to get positioned over the specified track.

Access time is the time it takes for a block to start being read; the sum of seek time and latency.

Latency is the time it takes for the specified sector to be in position under the read/write head.

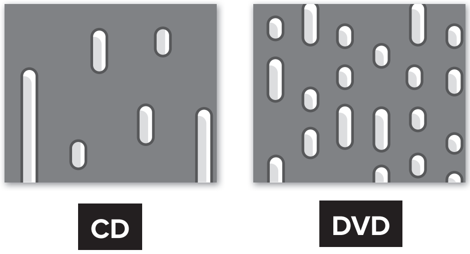
Transfer rate is the rate at which data moves from the disk to the main memory.

Actual hard disk drives that can be found in our computers consist of several magnetic disks, called **platters**, one on top of the other and each with its own read/write head that is attached to a spindle that rotates. All of the tracks that line up under one another are called a **cylinder**. So, in order to locate specific data on a hard disk, an instruction to the disk needs to specify a platter number, a cylinder number, and the sector.

Optical media

Other secondary storage devices include optical drives like CD/DVD or Blu -Ray drives. Unlike magnetic disks, these devices read and write data optically from a plastic-aluminum disk using a laser beam. In writing mode, the laser beam etches a series of **pits** on the aluminum layer as the disk rotates, resulting in a spiral of pits and **lands** which represent binary 0s and 1s. When reading, the laser beam bounces off the lands in the aluminum layer but not off the pits, so the binary sequence can be picked up by a sensor.

These pits are really tiny. Imagine that on the surface of a Blu -Ray disc there are about 200 billion pits and lands.



In these images, we can compare the pits on the surface of a CD (left) to those on a DVD (right), both magnified 20,000 times!

If you happen to come across an old hard disk drive and feel like you want to pop the cover and examine the platters and heads, be warned! All components inside a hard disk case are carefully aligned and sealed from the outside environment.

A small particle of dust from the air or even a gentle nudge on the heads can make the hard drive unusable! On the other hand, if the drive is already useless, go right ahead!

Hands on!

1. Explain the key components of the von Neumann architecture and describe how this design has influenced the development of modern computers. In your essay, examine how the interaction between the CPU, memory, and input/output devices enables a computer to perform tasks.
2. Describe the steps involved in the fetch-execute cycle and explain why each step is crucial for a computer to function correctly. What challenges might arise if any of these steps fail, and how could these challenges be addressed?

LESSON 3

Operating systems

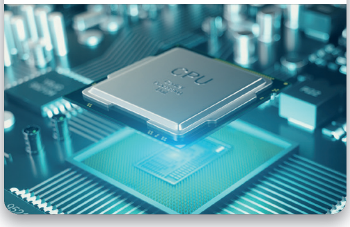
In the previous lessons, we talked about the internal components and devices that make up a computer. All these parts, from the transistors and logic gates to the CPU and hard disk drives, form the **hardware** of the computer. It is now time to talk about the **software**. This consists of all the instructions we provide to a computer for it to function and perform specific operations. A set of instructions designed for a particular task is called a **program**.

Modern software is divided into two categories: application software and system software.

- **Application software** refers to all the programs that are designed to solve problems in the real world, helping the computer user. Most of the programs that you use in your computer, like word processing programs, browsers for surfing the Internet, games and media player programs are all application software.
- **System software**, on the other hand, manages the computer system itself. It provides the tools and an environment in which application software can be created and run. System software often interacts directly with the hardware and provides more functionality than the hardware itself does.

The operating system of a computer is the core of its system software. An operating system manages computer resources such as memory and input/output devices, it allows application software to access system resources, and it provides a direct user interface to the computer system.

Hardware	⟷	System Software	⟷	Application Software
CPU		Utilities		Internet Browsers
Disk		Operating system		Databases
Printer				Word processors
Mouse				Spreadsheets
				Computer Games



Windows

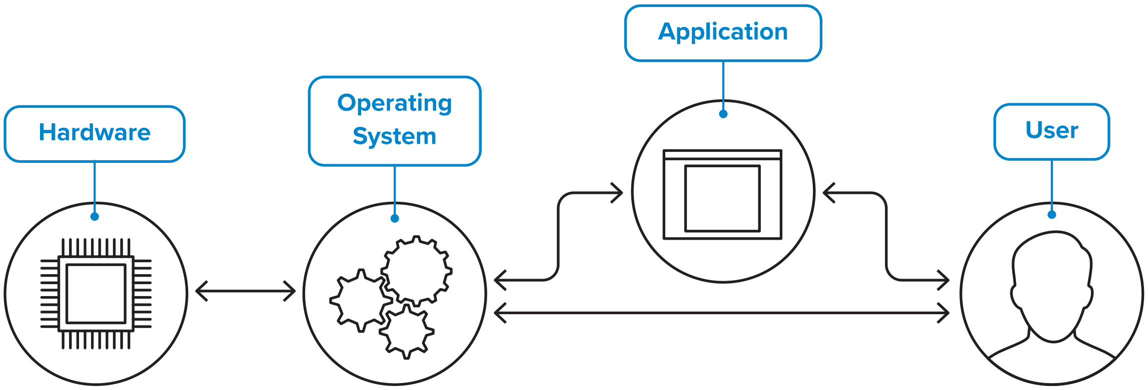
Smart Tip

We can say that system software includes software development tools, the programs that help us create application software and other system software.

Remember the fetch-execute cycle? We said that an executing program is loaded in the main memory and its instructions are processed one after another by the CPU.

All computers support **multiprogramming**, which is the technique of keeping multiple programs in the main memory at the same time. These programs compete for access to the CPU, in order for them to be executed. So, it is the operating system's job to perform **memory management** to keep track of what programs are in memory and where in memory they are located.

The operating system must also perform **process management**. A **process** is defined as a program you can execute. Since many active processes in the CPU want to execute their instructions at the same time, the operating system has to keep track of the progress of each and carefully manage them so that as they take turns in using the CPU, they continue from where they left off.



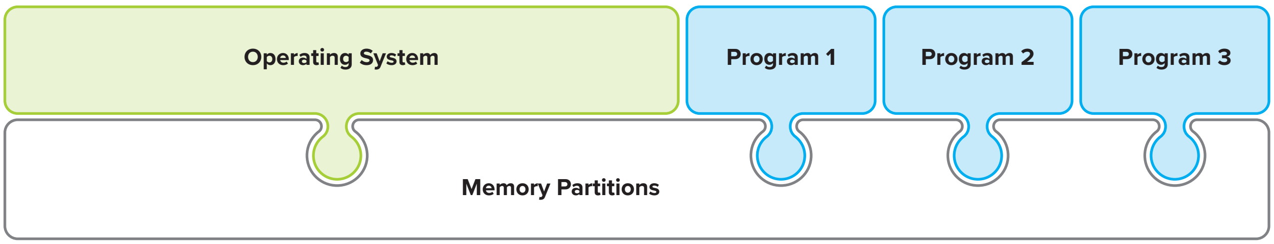
Memory management

The operating system must:

- track where and how a program is located in memory.
- convert logical program addresses into actual memory addresses.

The main memory is seen by the operating system as a continuous storage space that is divided into groups of bits, containing instructions or data. Each of these parts needs to be uniquely identified, so it is given an **address**. Addresses are just integers starting from 0, which is the first **memory address**.

The problem is that each program does not know beforehand which addresses in memory are going to be assigned to it. So, a program refers to its other instructions and data by using relative addresses called **logical addresses**. It is the operating system's job to map a program's logical addresses to the corresponding physical addresses, which are the actual addresses of the main memory, after the program has been placed in memory. This process is called **address binding**.



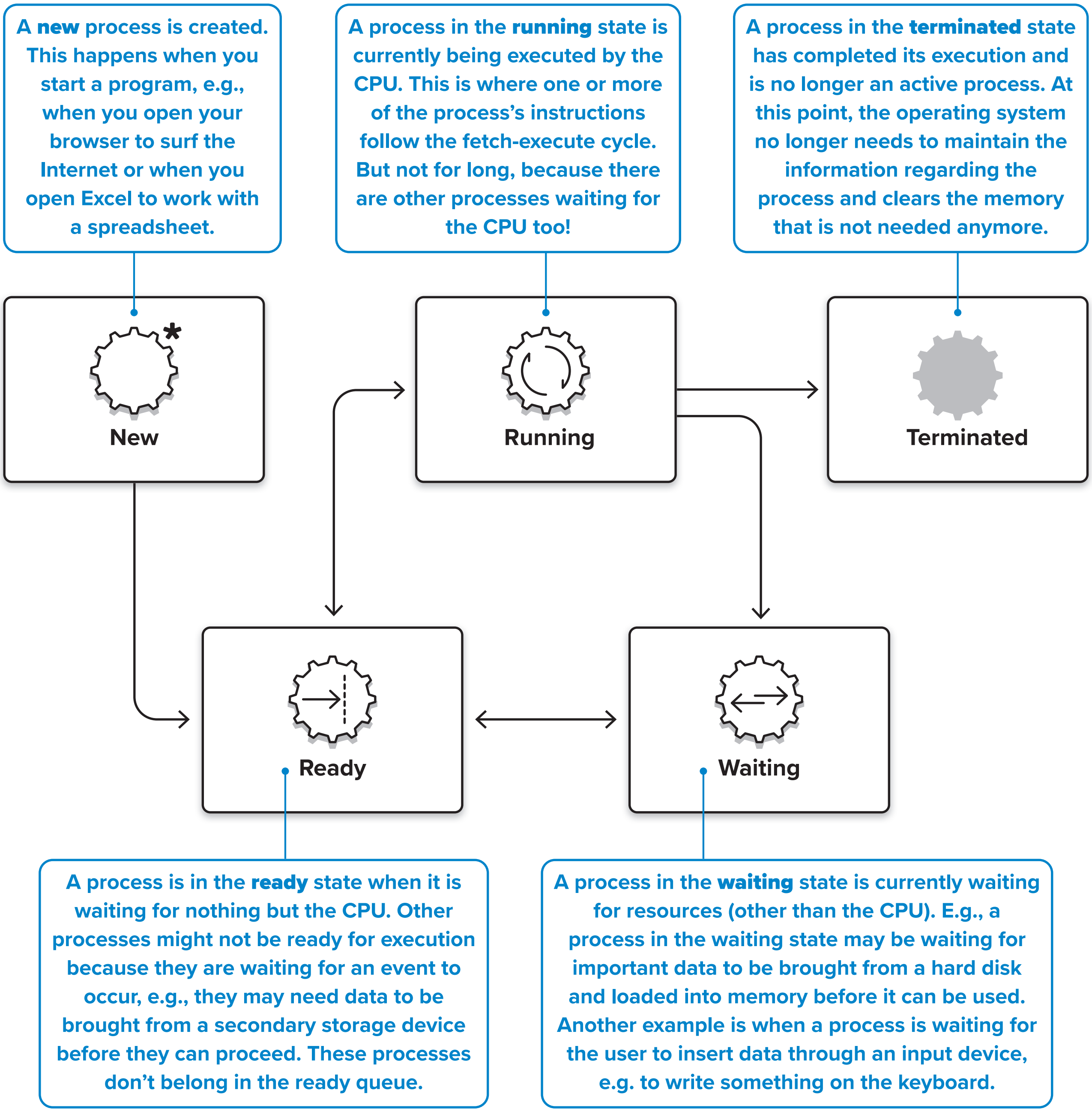
History

Almost all popular operating systems in use today are based on UNIX, an operating system first introduced in 1969.

Process management

The operating system must also manage the use of the CPU by the individual processes. Only one process can execute part of its instructions at any given time in the CPU, so every process goes through a life cycle of various process states as it gains and loses control of the CPU. More specifically, a process enters the system, is ready to be executed, is executing, is waiting for a resource, or is finished.

Let's find out what happens to a process as it goes through each stage.



Smart Tip

Note that many processes may be in the ready state or the waiting state at the same time, but only one process can be in the running state. That's why there is the ready queue and the waiting queue where processes line up to wait in each of these states.

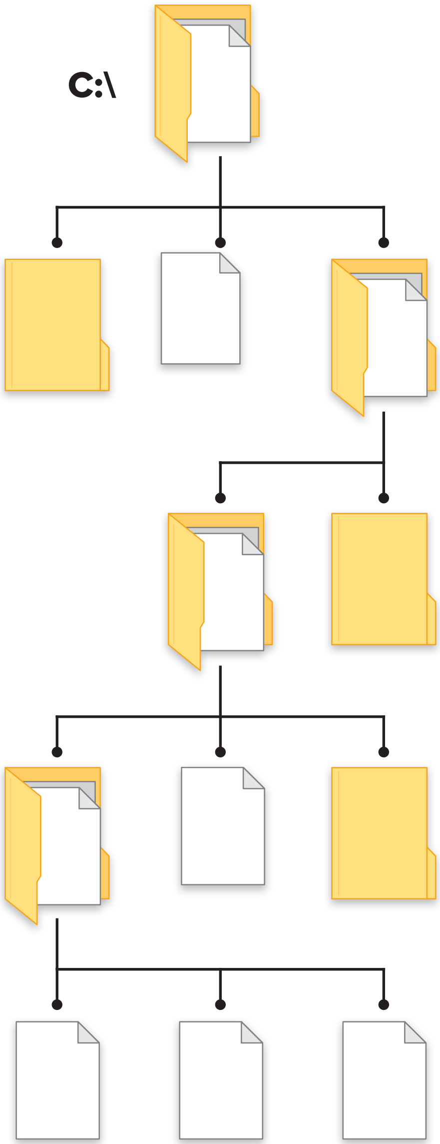
File systems

The organization of secondary storage, like hard disks, is also one of the most important jobs of an operating system. Remember, secondary storage is non-volatile, so this is where all our programs and data reside when the computer is turned off.

Information on a hard disk is organized and stored in files. A **file** is a named collection of related data and it is the main organizational unit of a hard disk. A file can contain a program or data of one type or another. For example, your browser program and a digital image are both files on your hard disk.

A **file system** is the logical view that an operating system provides so that users can manage information as a collection of files. A file system is often organized by grouping files into directories. A **directory** is a named collection of files. Think of a folder in which you can store data files, program files, or other folders.

A directory of files can be contained within another directory. The directory containing another is usually called the **parent directory**, and the one inside is called a **subdirectory**. You can create as many such nested directories as you need to organize your data. One directory can contain multiple subdirectories and each of these subdirectories can contain their own subdirectories creating a hierarchical structure. From this concept, you can easily understand that a file system is like a directory tree, showing directories and files within other directories. The directory at the highest level is called the **root directory**.



Hands on!

1. Why do you think an operating system is essential for running both system and application software? Can you imagine using a computer without an operating system? What challenges would you face?
2. Describe the process management capabilities of an operating system. How does the OS manage the life cycle of a process, and why is this management crucial for system stability and performance? Provide examples of how improper process management can lead to system issues.

LESSON 4

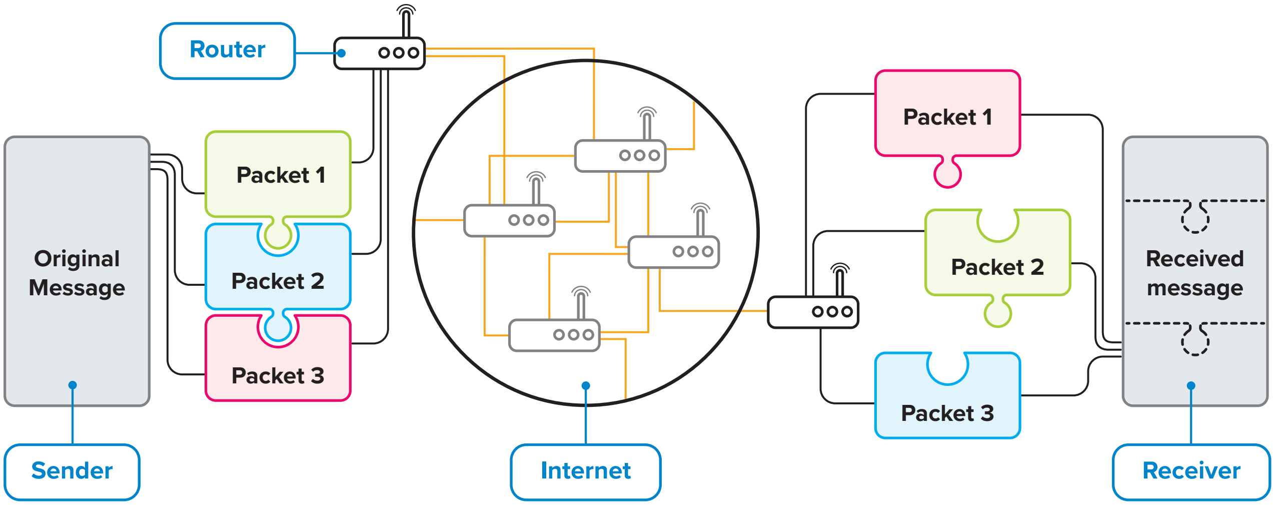
Network fundamentals

Computers connect to each other and form networks in order to communicate and share resources. On a larger scale, every computer in the world can communicate with another through the Internet, the global network of networks. Now, it's time to explore at some of the underlying mechanisms that enable information to travel through networks and reach its destination.

Packet switching

In order to achieve more efficient transfer of messages between different devices over networks, each message is divided into fixed-sized, numbered **packets**. The packets are then sent over the network to their destination, where they are received and reassembled to form the original message. This process is known as **packet switching**.

The individual packets of a message may take different routes in the network on their way to their destination. As a result, they may arrive in a different order than the one they were sent in. So, upon arrival, the destination device must put them into the proper order to recreate the original message.



The reason different packets can reach the same destination through different paths lies in the design of all modern networks and, of course, the Internet as well. Traffic between networks is directed by network devices called **routers**. Neither your computer nor routers know the exact path a message from one computer to another must follow, especially inside a vast web of networks such as the Internet. What routers do know is the next best step toward a destination.

A router that receives a packet reads the destination and decides to which other router, directly connected to it, it must be forwarded to. The next router repeats this process until the packet reaches a router that is connected to the recipient device. If a certain path is blocked by a router malfunction, or if a path currently has a lot of network traffic, a router might send a packet along an alternative route. In this way, we achieve speed and reliability in transferring information over the network.

Network addresses

In order for two computers to communicate with each other, they need to be able to identify each other from among all the other computers in the world. This is done in two ways.

A **hostname** is a unique name that specifies a particular computer on the Internet. Hostnames are generally readable words separated by dots.

For example: wikipedia.org

Although it is convenient for humans to use and remember hostnames, network devices like routers that transfer the actual messages back and forth use another kind of identification mechanism called an **IP address**. An IP address is a series of four decimal numbers separated by dots.

For example: **91.198.174.225**

Each of the four numbers that make up an IP address can be in the range 0–255.

As you can imagine, each hostname has a corresponding IP address. In order for us to be able to use convenient and easy-to-remember hostnames, we need a way to translate (resolve, as it is called) each hostname into the corresponding IP address. This is done automatically by the **domain name system** (**DNS**), which is a network of computers that constantly store and provide the mappings from hostnames to IP addresses.

OSI model layers		
Layer	**Name**	**Description**
7	Application	Interacts with software applications.
6	Presentation	Transforms data into the form that the application layer can accept, formats and encrypts data.
5	Session	Establishes, manages, and terminates the connections between the local and remote application.
4	Transport	Transfers data from a source to a destination host, with no errors in transmission.
3	Network	Transmits data from node to node. Uses routers and switches to control the flow.
2	Data Link	Packages raw bits from the physical layer into frames. Performs error detection.
1	Physical	Transmits raw bits through the physical medium.

Protocols

Generally speaking, a protocol is a set of rules that defines how two things speak to each other or interact. So, a **network protocol** is a set of rules that defines how data is formatted and processed on a network. Here is an illustration of the layers of the **Open Systems Interconnection Model** (**OSI**) in which networking protocols are demonstrated.

The OSI model

The development of a network application program is a fairly complex process. Starting from the transmission medium (cable) and ending at the final program, there are several stages in-between. In order to simplify this process, the ISO organization introduced the reference model OSI (Open Systems Interconnection).

The OSI model consists of seven layers. Each layer performs a specific task and serves the layer above it and is served by the layer below it. Changes happening in one layer do not affect the other layers.

	The OSI model	The TCP/IP model
7	Application	Application
6	Presentation	
5	Session	
4	Transport	Transport
3	Network	Internet
2	Link	Network Access
1	Physical	

TCP/IP

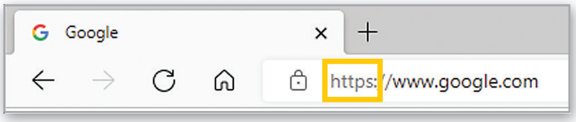
TCP stands for **Transmission Control Protocol** and **IP** stands for **Internet Protocol**. The name TCP/IP refers to a suite of protocols and utility programs that support low-level network communication. The name TCP/IP implies that TCP relies on the IP foundation below it. Those lower two layers of the **protocol stack** form the foundation of Internet communication.

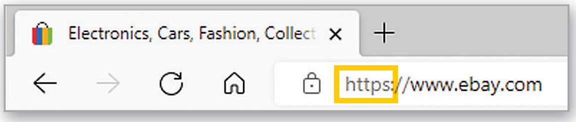
IP software is responsible for the routing of packets through the web of the various networks to their final destination. TCP software splits the messages into packets, passes them to the IP software for transmission, and then reorders and reassembles the packets at their destination. TCP software also deals with any errors that occur, such as if a packet never arrives at the destination or the contents of a packet are corrupted.

UDP stands for **User Datagram Protocol**. It is an alternative to TCP. The main difference is that TCP is highly reliable, at the cost of decreased performance, while UDP is less reliable, but generally faster. Note that UDP is part of the TCP/IP suite of protocols.

Essential high-level protocols above TCP/IP		
Acronym	**Protocol name**	**Description**
FTP	File Transfer Protocol	A protocol that allows a user on one computer to transfer files to and from another computer.
SMTP	Simple Mail Transfer Protocol	A protocol used to transfer electronic mail messages.
HTTP	Hyper Text Transfer Protocol	A protocol defining the exchange of World Wide Web documents (web pages).
HTTPS	Hypertext Transfer Protocol Secure	A variant of HTTP offering secure communications between two computers.
DNS	Domain Name System	A system that translates hostnames of computers to corresponding IP addresses.

The HTTPS protocol is a powerful tool against online attacks. What it does essentially is encrypt all traffic to and from a specific website so that when you access that site, no one can access your passwords and what you do on the site, even if a hacker has access to all the packets of your connection. Make it a habit to use HTTPS connections on sites that you have to log into, like Facebook and eBay, or even in search engines like Google to protect your search queries from others. All you have to do is add the "s" after http on your browser's address bar. It's also a good idea to update your bookmarks so you don't have to add the "s" by hand all the time.





The World Wide Web

One of the most common mistakes many people make when discussing networks is that they use the terms Internet and web interchangeably.

The Internet is the global network that allows any computer connected to it to communicate with all others. The Internet is a medium through which information travels.

On the other hand, the World Wide Web, or simply the "web", is a specific way of accessing information over the medium of the Internet. The web is a system of interlinked documents called web pages that are accessible via the Internet. Interlinked means that each web page can link to one or more other web pages. The web uses the HTTP protocol, one of the many "languages" spoken over the Internet, to transmit data.

To access web pages, we use programs called web browsers, which allow us to read these documents and click links to navigate to other pages. These links are called **hyperlinks**.

Each page on the web is uniquely identifiable by an address called a **URL**, which stands for Uniform Resource Locator.

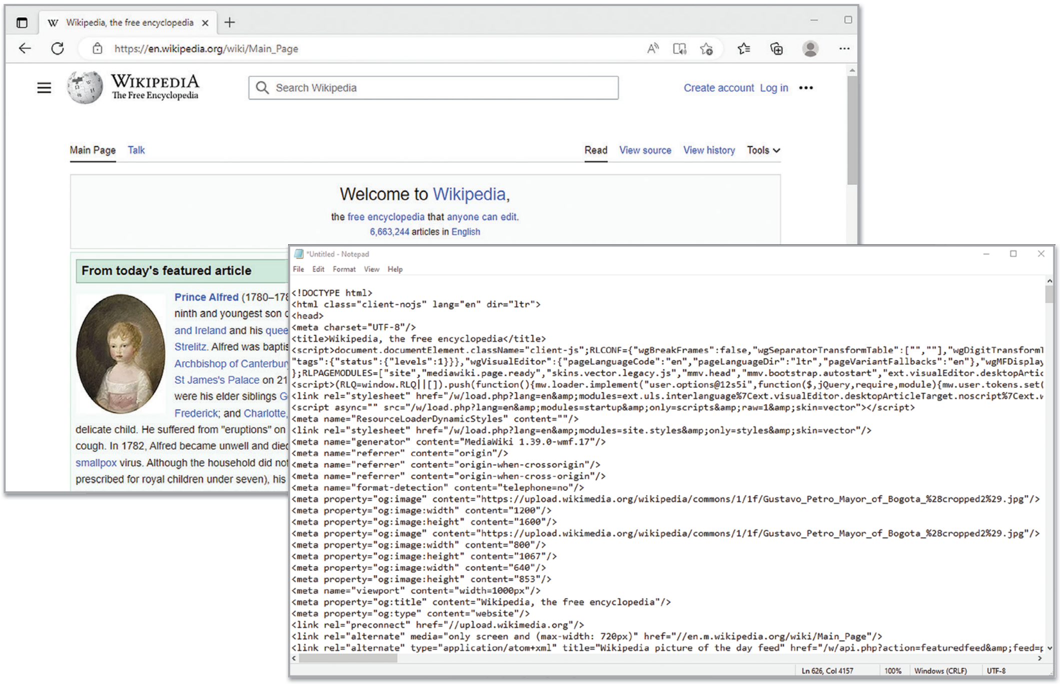
For example: **https://en.wikipedia.org/wiki/World_Wide_Web**

Note that a URL contains a hostname (en.wikipedia.org) as well as other information used to locate a particular document offered by the specific host.

History

The TCP/IP protocol suite resulted from research and development conducted by DARPA, a US Department of Defence agency, in the early 1970s. It was first used in ARPANET, the world's first operational packet switching network and the "father" of what was to become the global Internet.

Web pages are created with Hypertext Markup Language, or HTML. Although the name hypertext suggests pages which contain text and links to other pages, today it is most common for a web page to contain or link to other types of media as well, such as images, audio, and video.



Firewalls

A **firewall** is a software or hardware-based network security system that controls the incoming and outgoing network traffic by analyzing the data packets and determining whether they should be allowed through or not. It can be found as a software program running on your computer, embedded in networking hardware devices like routers or as a standalone device. A firewall creates a security barrier that separates and protects a single computer or a local network from the Internet.

Let's explore the evolution of firewalls to get a better idea of the functionality and capabilities of today's firewalls.

First generation

The first firewalls acted as **packet filters** by inspecting each individual packet that wanted to come in or out of the local network. The firewall filters packets based on the TCP/IP information they contain and decides if a packet will pass through by consulting a set of rules configured by the owner of the network under protection. For example, a firewall can be configured to only allow packets of a specific protocol and block all others or allow packets coming from a specific server.

Second generation

Second-generation firewalls perform the work of their first-generation predecessors but retain packets until enough information is available to make a judgment about its state. Known as **Stateful Packet Inspection**, it records all connections passing through it and determines whether a packet is the start of a new connection, a part of an existing connection, or not part of any connection.

History

Sir Tim Berners-Lee is a British computer scientist, best known as the inventor of the World Wide Web. He implemented the first successful communication between two computers using the HTTP protocol in 1989.

Third generation

The latest firewalls, called **application layer firewalls**, are able to inspect traffic by filtering high-level protocols like FTP, DNS, and HTTP and block any unwanted protocols or detect if a protocol is being abused in any harmful way. They can "understand" the use of data packets and reject anything that is strange.

Check your own firewall

Microsoft Windows comes equipped with a software firewall. Check whether it is on, and turn it on if it isn't.

To turn on the firewall:

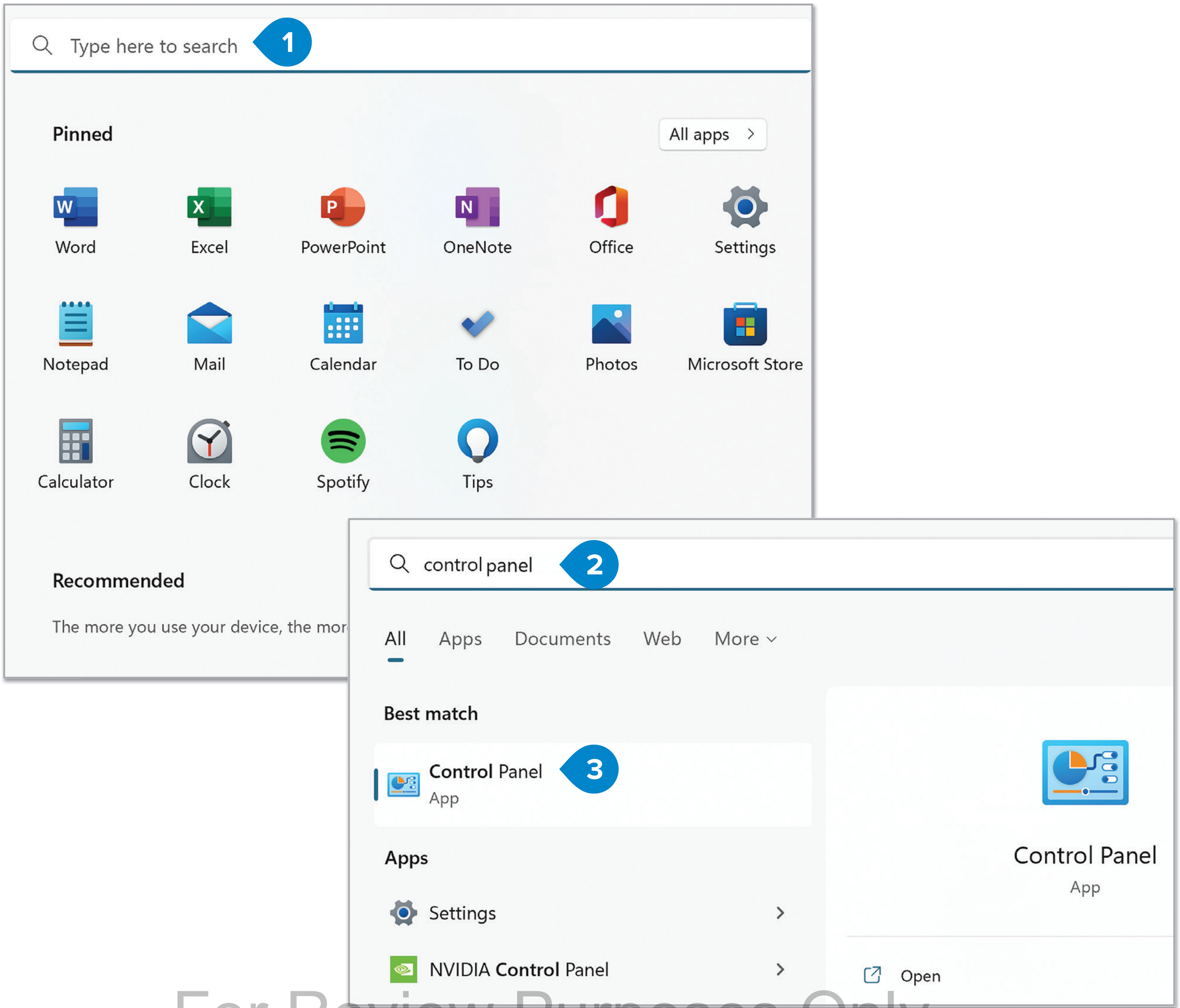
> Click the search box 1 and type "**control panel**". 2 Click the **Control Panel** application. 3

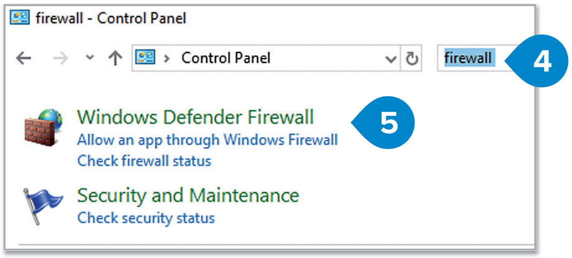
> In the search box, type "**firewall**", 4 and then click **Windows Defender Firewall**. 5

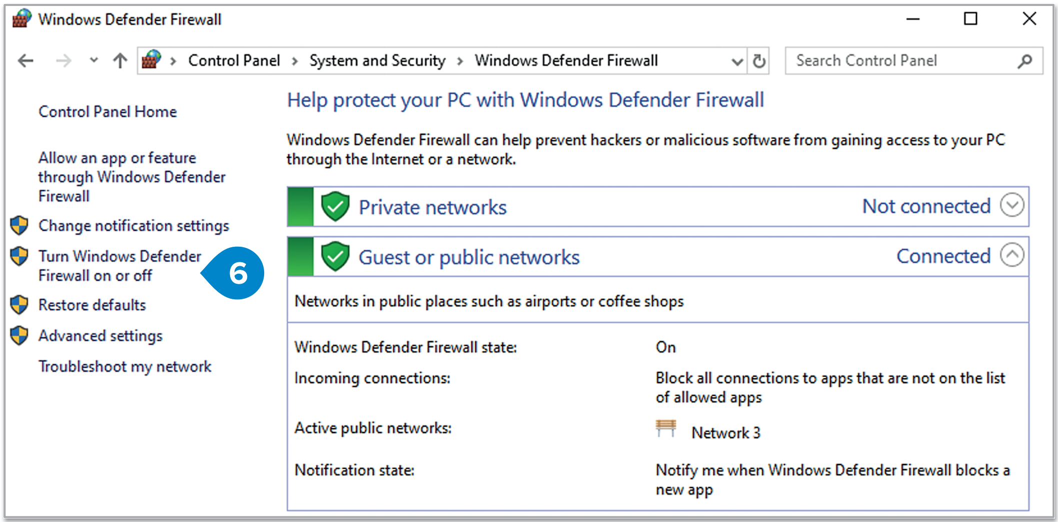
> If everything is green, then your firewall is enabled.

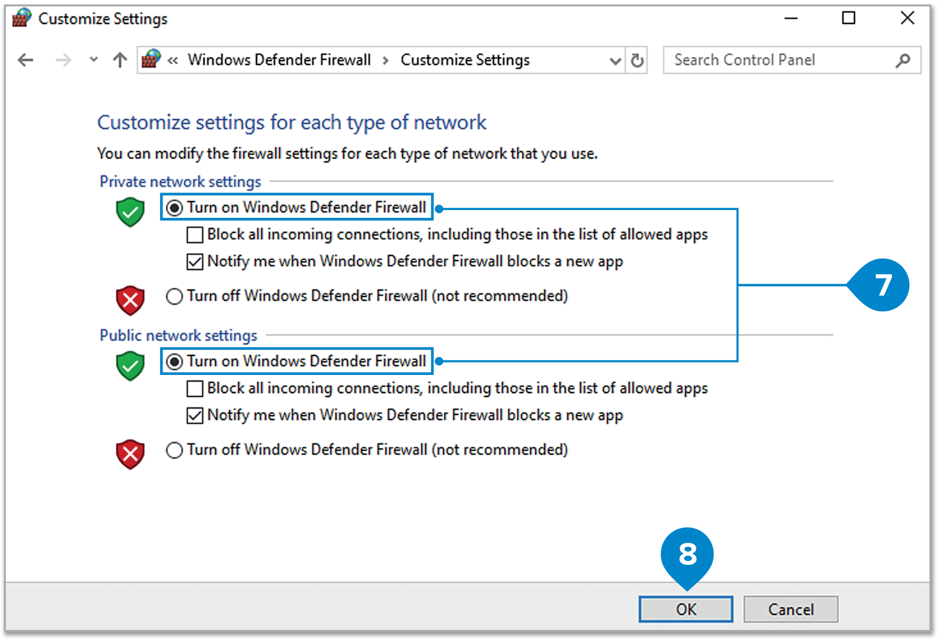
> If the firewall is off, click **Turn Windows Defender Firewall on or off**. 6

> Click **Turn on Windows Defender Firewall** 7 for all your networks and click **OK**. 8









Hands on!

1. Describe the relationship between IP addresses and hostnames, and explain how the Domain Name System (DNS) helps in this process.
2. Why do you think firewalls are a critical part of network security? Can you explain how a firewall might protect a network from potential threats?

Project

Computer evolution and history

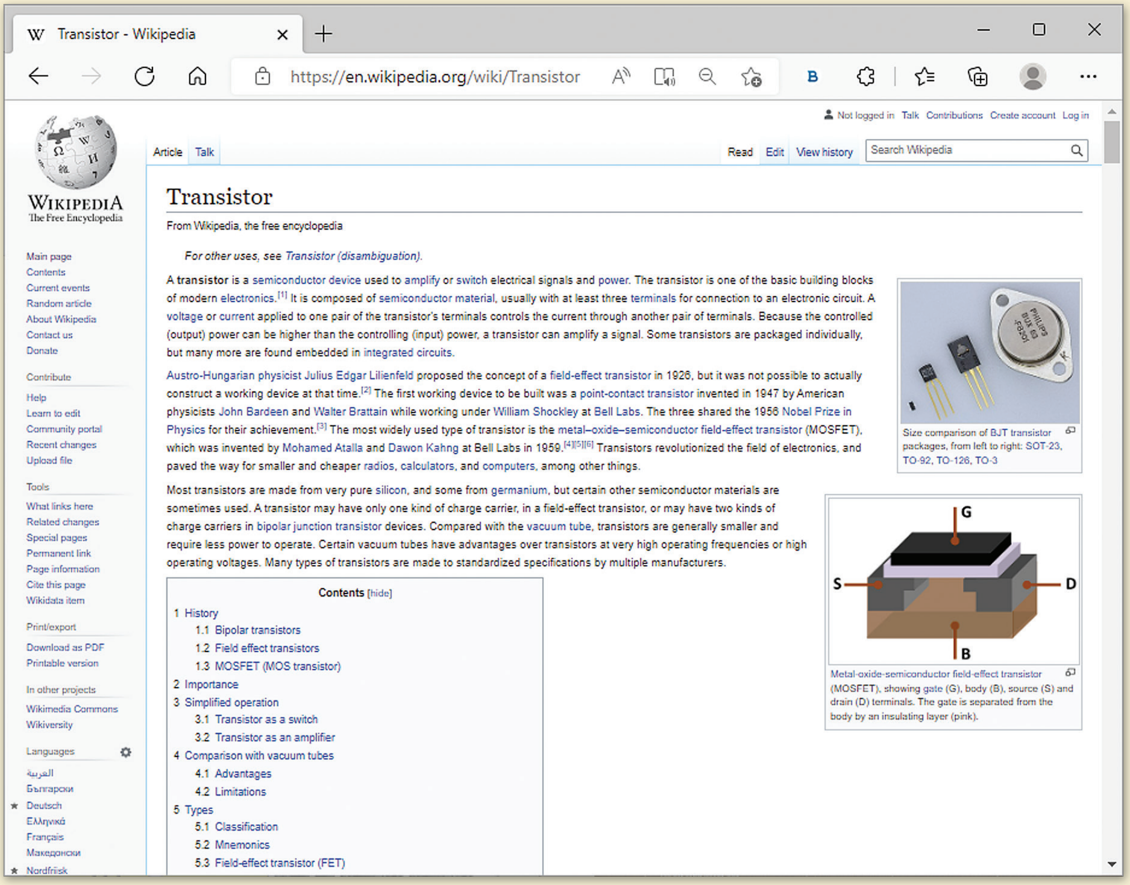
Form teams! In this project you are going to create a presentation about computers and their evolution through history.

> Try to answer the following questions:

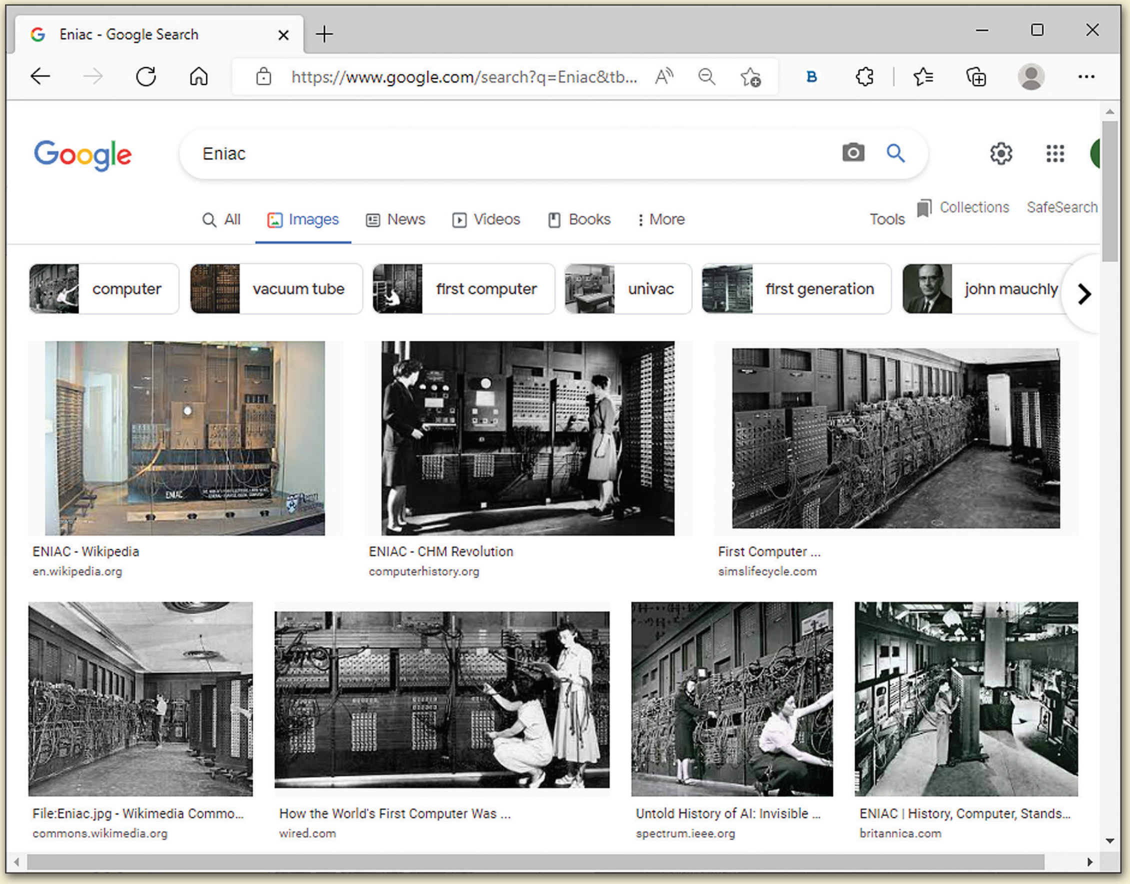
- What were the first computers like?
- What changes from year to year took place as technology progressed?

> Do some research on the evolution of computer hardware and include your findings in your timeline presentation.

> Search the web for relevant information. Your goal should be to create a timeline of the most important milestones in the history of computer evolution.



> Make your presentation more engaging with images of famous computers, computer scientists, and hardware components.



> Don't forget to include a section about the evolution of the various operating systems that were used in the computers of each era.

> Devote a section of your presentation to the most important computer scientists and how each one contributed to the evolution of computer systems.

> When done, present your work in class. Do you remember the presentation tips you have learned?

Wrap up

Take a moment to reflect on your progress.

How confident are you in your ability to apply the following skills?

> Explaining the differences between decimal, binary, and hexadecimal, and converting numbers between them.

> Describing how data is represented in binary.

> Using Boolean logic and logic gates to solve problems.

> Understanding how transistors and integrated circuits work.

> Describing the parts of the von Neumann architecture and how it functions.

> Explaining the fetch-execute cycle.

> Describing how the OS manages memory and processes.

> Evaluating storage performance by calculating key metrics and understanding the differences between memory and storage types.

> Explaining what an operating system does and describing how files are organized on a computer.

> Explaining how packet switching works and understanding how IP addresses and hostnames are used in networking.

> Describing the seven layers of the OSI model.

> Illustrating how firewalls work and setting one up.

> Explaining how technology has improved and complicated the workplace.

> Explaining why lifelong learning is important.

Foundations of Technology

Computer Science

Code the future, shape the world

Have you ever wondered what powers the digital world behind the scenes? Imagine understanding how data flows, how systems connect, and how technology shapes our lives. What if you could explore the fundamentals of computer science and uncover the technologies we rely on every day?

Exploring the Digital World introduces you to computer science basics like data manipulation, network fundamentals, and the role of technology in society. Learn about monitoring systems, artificial intelligence, and emerging technologies through hands-on projects. You'll also explore networking concepts, from wired and wireless connections to cloud storage and data security.

By the end, you'll be ready to navigate the digital landscape and apply your skills in technology and project management with confidence.

McGraw Hill
binarylogic
ISBN-13: 978-1-2649-2924-5
ISBN-10: 1-2649-2924-2