TASK 1

Data manipulation

Modern digital computers can be found practically anywhere around us nowadays. 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.

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. That 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 the place value increases by a power of two (one's, two's, four's, eight's etc.).

131, for example, is 10000011 in the binary system:

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**.
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, each place value is 16 times the previous one, always starting at value 1 for the rightmost digit, of course.

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:

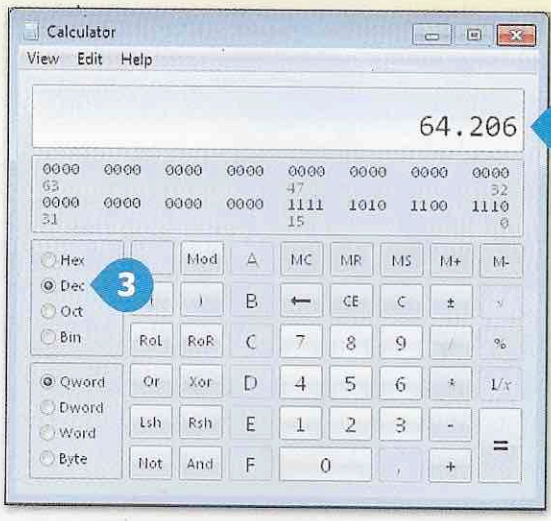
> Open **Calculator**.

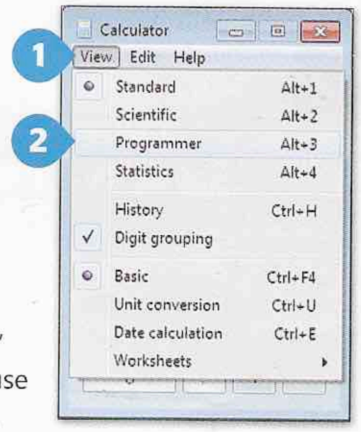
> Click **View** 1 and select **Programmer** mode. 2

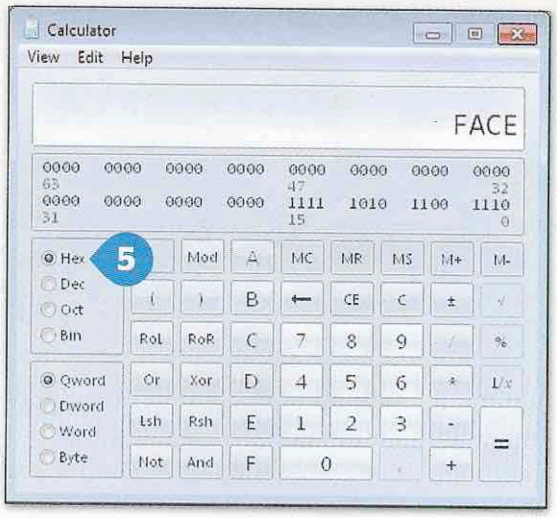
> Select a number system. 3

> Input a number into that number system. 4

> Select another number system to convert your number to. 5







Since the hexadecimal system contains some letters, computer programmers like to create "magic numbers" which 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. Since computers are only able to store and work with 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 seen 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.

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

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** which is shown below. ASCII stands for American Standard Code for Information Interchange.

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

Left Digit(s) \ *Right Digit*	*ASCII* 0	1	2	3	4	5	6	7	8	9
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT
1	LF	VT	FF	CR	SO	SI	DLE	DC1	DC2	DC3
2	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS
3	RS	US	☐	!	"	#	$	%	&	'
4	(	)	*	+	,	–	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	<	=	>	?	@	A	B	C	D	E
7	F	G	H	I	J	K	L	M	N	O
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[	\	]	^	_	`	a	b	c
10	d	e	f	g	h	i	j	k	l	m
11	n	o	p	q	r	s	t	u	v	w
12	x	y	z	{	|	}	~	DEL		

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.

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*

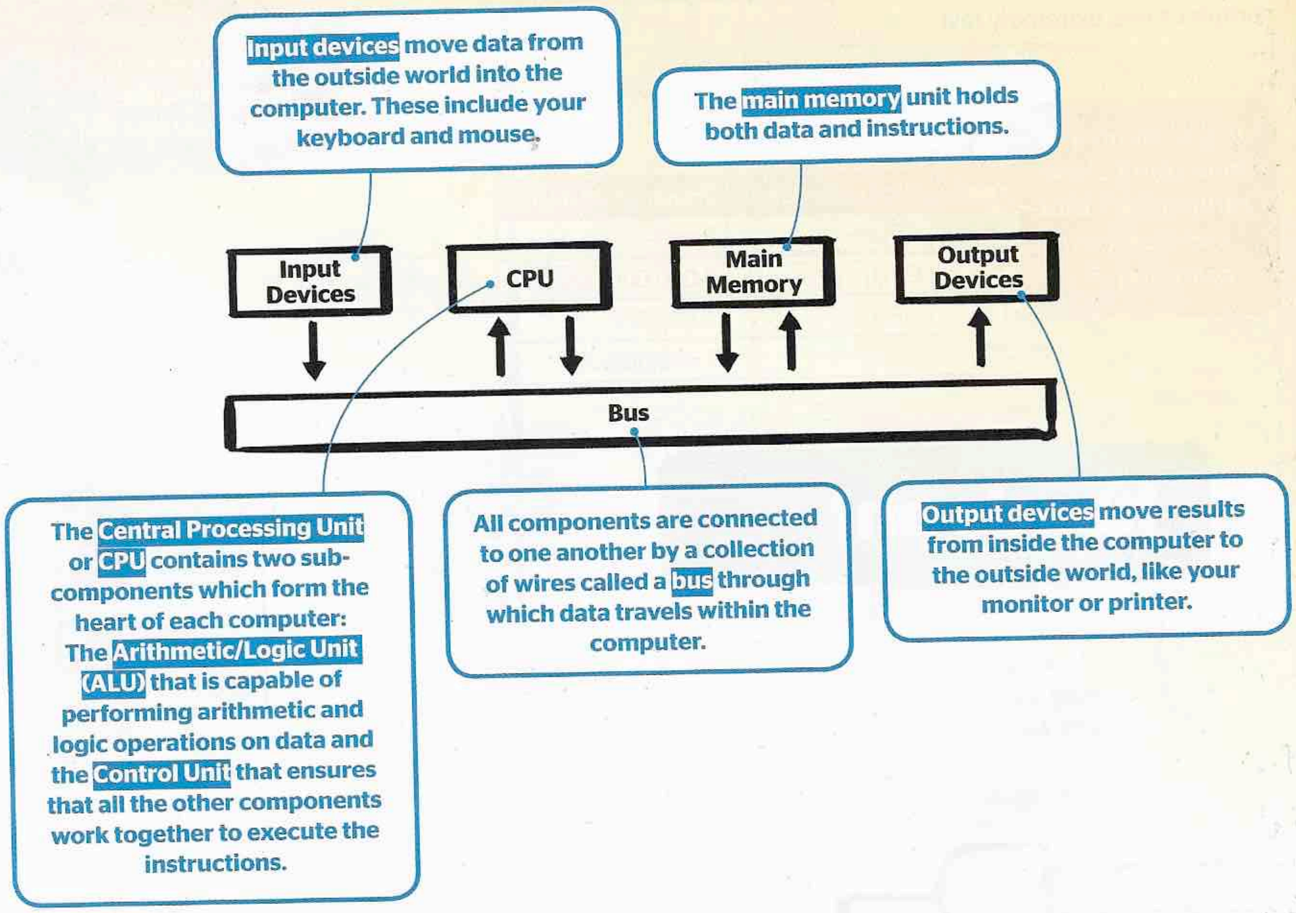
Video is the most complex data type to represent but generally it can be thought of as a series of images, saved in binary and played back one after the other. These images are usually compressed in order to save storage space and process the images as fast as possible.

TASK 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, and because computers are binary machines, instructions are, of course, binary sequences. So, 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 which work together in order for the computer to function.


Input devices move data from the outside world into the computer. These include your keyboard and mouse.
The main memory unit holds both data and instructions.
Input Devices
CPU
Main Memory
Output Devices
Bus
The Central Processing Unit or CPU contains two sub-components which form the heart of each computer: The Arithmetic/Logic Unit (ALU) that is capable of performing arithmetic and logic operations on data and the Control Unit that ensures that all the other components work together to execute the instructions.
All components are connected to one another by a collection of wires called a bus through which data travels within the computer.
Output devices move results from inside the computer to the outside world, like your monitor or printer.

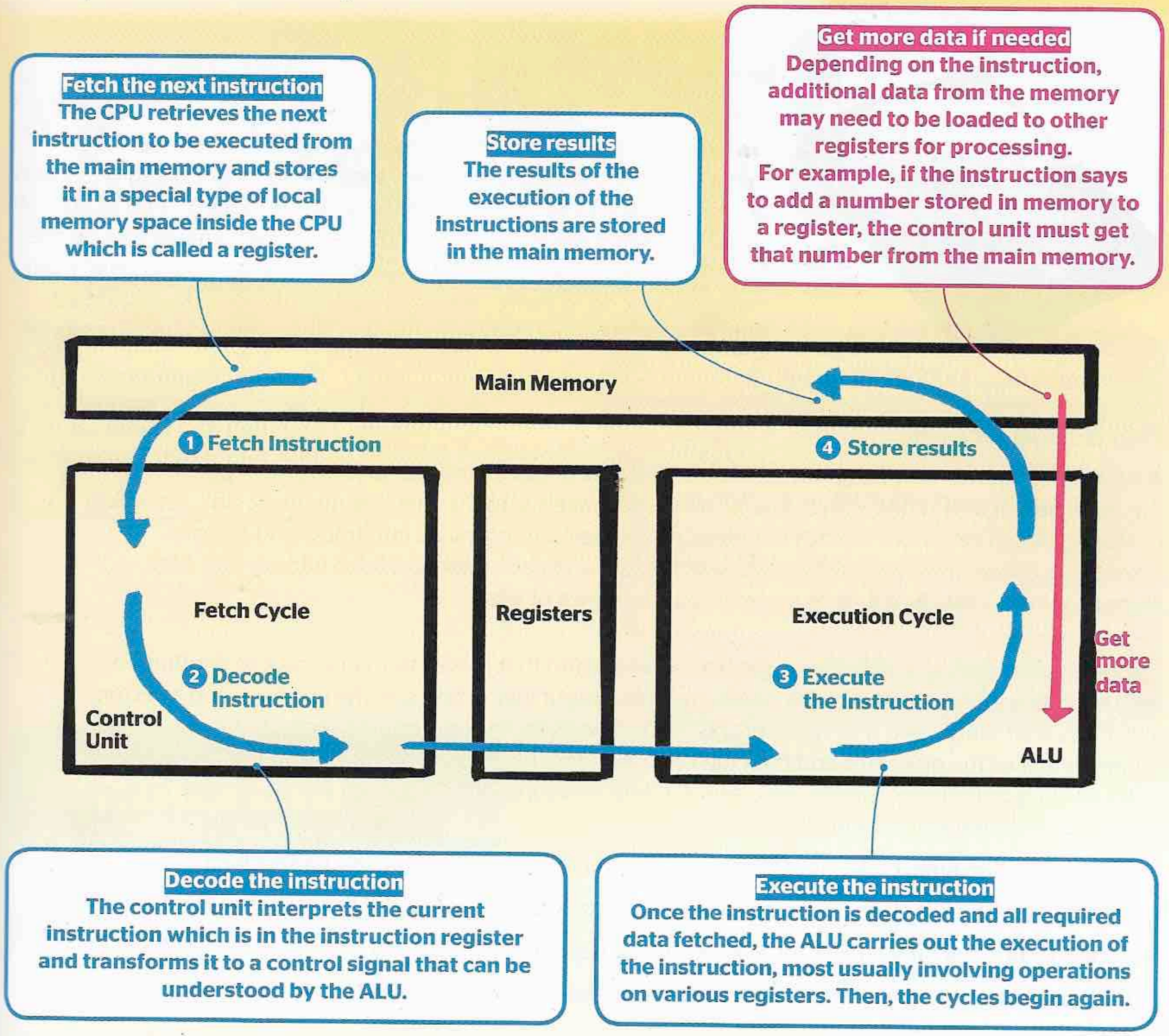
HISTORY

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 see 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 are:

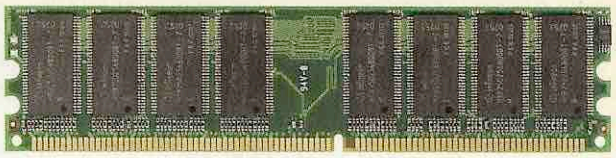


Fetch the next instruction
The CPU retrieves the next instruction to be executed from the main memory and stores it in a special type of local memory space inside the CPU which is called a register.
Store results
The results of the execution of the instructions are stored in the main memory.
Get more 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.
Main Memory
1 Fetch Instruction
4 Store results
Fetch Cycle
Registers
Execution Cycle
2 Decode Instruction
3 Execute the Instruction
Control Unit
ALU
Get more data
Decode the instruction
The control unit interprets the current instruction which is in the instruction register and transforms it to a control signal that can be understood by the ALU.
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 cycles begin again.

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 and the instructions that are stored inside 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. 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 a 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.

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

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

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

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

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.

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.

TASK 3

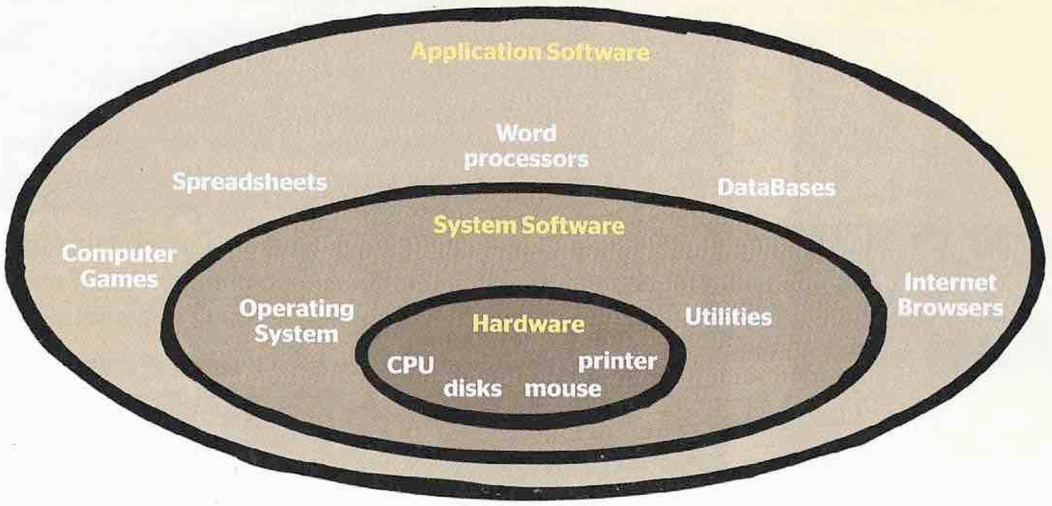
Operating systems

In the previous tasks 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.



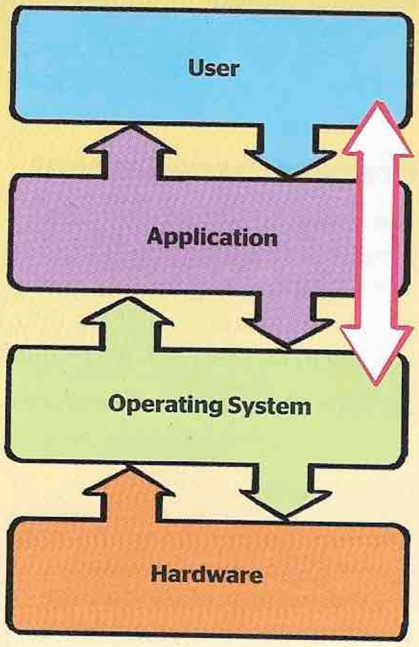
The operating system of a computer is the core of its system software. An operating system manages computer resources, such as memory and input/output devices, It allows application software to access system resources and it provides a direct user interface to the computer system.

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 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

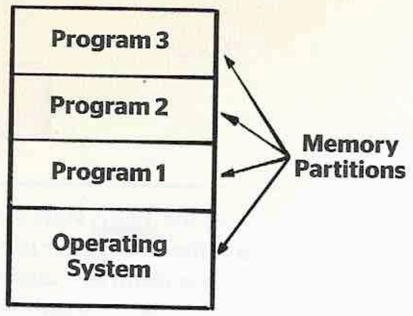
The operating system must:

> 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.