**Year 10 PPE2 Computer Science**

**Paper 1: Binary and Data Representations**

## **Units**

In a computer, all data is stored in binary form. A binary digit has two possible states - 1 and 0.

A binary digit is known as a bit. A bit is the smallest unit of data a computer can use.

Eight bits are known as a byte. A byte is significant in that a single character can be represented in binary in eight bits - one byte.

Binary values are used to represent many kinds of data, namely numbers, text, images or sound.

Four bits or half a byte is known as a nibble.

To be able to reference large numbers of 0s and 1s, the binary unit system is used:

****

****

## **Negative numbers**

Negative numbers can be represented in two ways:

* sign and magnitude
* two's complement

### **Sign and magnitude**

The left-most value in a binary number is called the most significant bit (MSB). It holds the highest place value and is the ‘sign’ bit. In a byte, that will be the place value for 128 or 27. In sign and magnitude, a 0 in the MSB indicates that the number is positive and a 1 that the number is negative.

### **Sign and magnitude conversion to denary**

Converting a sign and magnitude number is exactly the same as any binary number. However, the value in the MSB is ignored as part of calculation.

**Example:** signed binary number 01101100

****

The value in the MSB is 1 so this is a negative number.

Since there are now only seven bits available, the largest positive number that can be held in this format is 01111111 (binary) or +127 (*denary*). The largest negative number is 11111111 (binary) or -127 (denary).

Some problems with sign and magnitude are that there are two values for 0 (10000000 and 00000000), a bit is wasted and there are problems with addition since it is not always clear how to deal with the sign bit.

### **Two’s complement**

In two’s complement negative numbers, the MSB is a negative value and also a sign bit.

****

### **Negative numbers and binary shifts**

There are two different types of binary shift which work differently for negative numbers. The two types are:

* arithmetic
* logical

#### **Arithmetic shifts**

To multiply a number, an arithmetic binary shift moves all the digits in the binary number along to the left and fills the gaps after the shift with 0:

* to multiply by two, all digits shift one place to the left
* to multiply by four, all digits shift two places to the left
* to multiply by eight, all digits shift three places to the left
* and so on

****

To divide a number, an arithmetic binary shift moves all the digits in the binary number along to the right and fills the gaps after the shift with the previous MSB value:

* to divide by two, all digits shift one place to the right
* to divide by four, all digits shift two places to the right
* to divide by eight, all digits shift three places to the right
* and so on

****

Two’s complement is therefore better for arithmetic shifts left since the sign is always retained.

#### **Logical shifts**

In logical shifts - left or right - a 0 is always copied in. This means that a logical shift is ideal for unsigned binary numbers.

****

## **Binary and denary**

Humans tend to use the *denary* number system. This is the base 10 system that you are familiar with. However, computers work in the *binary* number system, which is base 2. Denary numbers must be converted into their binary equivalent before a computer can use them.

The denary system has ten symbols - 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9. The value of each denary place value is calculated by multiplying the previous place value by ten.

Example



This gives (1 × 100) + (2 × 10) + (4 × 1) = 124.

### Binary to denary

The value of each binary place value is calculated by multiplying the previous place value by two. The first eight binary place values are:

****

In binary, each place value can only be represented by 1 or a 0.

To convert binary to denary, simply take each place value that has a 1, and add them together.

****

### **Denary to binary**

To convert from denary to binary, start by subtracting the biggest place value you can from the denary number, then place a 1 in that place value column. Next, subtract the second biggest place value you can, and place a 1 in the column. Repeat this process until you reach zero. Finally, place a 0 in any empty place value columns.

<https://www.youtube.com/watch?v=AMNunPK_5X4>

## **Binary addition**

To add together two denary numbers, take the first number, add the second number to it, and get an answer. For example, 1 + 2 = 3.

When adding two binary numbers together the process is different.

There are four rules that need to be followed when adding two binary numbers. These are:

* 0 + 0 = 0
* 1 + 0 = 1
* 1 + 1 = 10 (binary for denary 2)
* 1 + 1 + 1 = 11 (binary for denary 3)

**Example:** adding 01 + 101



**Example:** adding 01010011 + 01110110



### **Overflow**

Overflow occurs when the result of a calculation requires more bits - place values - than are in the available range.

For example, when using eight bits, the largest number that can be recorded is 11111111 (denary 255). When adding together two eight-bit numbers, a situation may occur when the result requires more than eight bits to hold it. For example, adding the binary numbers 11111110 (denary 254) and 00000010 (denary 2) would give:



The result is actually 10000000 (denary 256), which requires nine bits.

## Binary shifts

Binary numbers are multiplied and divided through a process called shifting. There are two types of binary shift - arithmetic and logical. They work the same way for positive numbers but differently for negative numbers.

### **Multiplication**

To multiply a number, a binary shift moves all the digits in the binary number along to the left and fills the gaps after the shift with 0:

* to multiply by two, all digits shift one place to the left
* to multiply by four, all digits shift two places to the left
* to multiply by eight, all digits shift three places to the left
* and so on

**Example:** 00001100 (denary 12) × 2



### **Division**

To divide a number, a binary shift moves all the digits in the binary number along to the right and fills the gaps after the shift with 0:

* to divide by two, all digits shift one place to the right
* to divide by four, all digits shift two places to the right
* to divide by eight, all digits shift three places to the right





## **Hexadecimal**

In computer science, different number bases are used:

* denary is base 10, which has ten symbols (0-9)
* binary is base 2 , which has two symbols (0-1)

Hexadecimal, also known as hex, is the third commonly-used number system and is base 16. It has 16 symbols - 0-9 and the capital letters A, B, C, D, E and F.



### Converting between binary and hexadecimal

The simplest way to convert from binary to hexadecimal, and vice versa, is to split the binary number into nibbles (four bits) first, then convert each nibble to hexadecimal. A nibble can hold 16 values in its 4 bits so is useful for converting hexadecimal.

#### Binary to hexadecimal

1. Start at the rightmost digit and break the binary number into nibbles.
2. Next, convert each nibble into hexadecimal
3. Put the hexadecimal digits together.

**Example:** 11000011 to hexadecimal

Break into nibbles: 1100 0011.

1100 = hexadecimal C and 0011 = hexadecimal 3. Remember, this is hexadecimal base 16 symbol 3, not denary symbol 3.

**Result:** C3

**Example:** 00110011 to hexadecimal

Break into nibbles: 0011 0011.

0011 = hexadecimal 3 and 0011 = hexadecimal 3

**Result:** 33

#### Hexadecimal to binary

1. Split the hexadecimal number into individual digits.
2. Convert each hexadecimal digit into its binary equivalent (a nibble).
3. Combine the nibbles to make one binary number.

**Example:** hexadecimal 28 to binary

2 = binary 0010 and 8 = binary 1000

**Result:** 00101000

**Example:** hexadecimal FC to binary

F = binary 1111 and C = binary 1100

**Result:** 11111100

**ASCII**

## Characters

Computers can only recognise binary. As a result, all characters - whether they are letters, punctuation or numbers - are stored as binary numbers. All of the characters that a computer can use are called a character set.

One of the standards in common use is the American Standard Code for Information Interchange (ASCII).

ASCII uses seven bits, giving a character set of 128 characters. The characters are represented in a table, called the ASCII table. The 128 characters include:

* 32 control codes - mainly to do with printing
* 32 punctuation codes, symbols, and space
* 26 upper case letters
* 26 lower case letters
* numeric digits 0-9

### Extended ASCII

Extended ASCII uses eight bits, giving a character set of 256 characters. This allows for special characters such as those with accents in languages such as French and Spanish.

****

**Low-level languages**

*Low-level languages* sit close to the computer's *instruction set*. An instruction set is the set of *instructions* that the *processor* understands.

Two types of low-level language are:

* *machine code*
* *assembly language*

**Machine code**

An instruction set relates to a specific processor and is written in machine code. The central processing unit (CPU) understands machine code directly and can act upon the instructions. A *program* written in machine code consists of 0s and 1s only. Machine code is very difficult to learn, write and *debug*. Even a very simple program could have thousands of 0s and 1s in it.

**Assembly language**

Assembly language sits between machine code and *high-level languages* in terms of ease of use. While high-level languages use *statements* to form instructions, assembly language uses *mnemonics* (short abbreviations). Each mnemonic directly corresponds with a single machine code instruction. Here are some examples of mnemonics:



In assembly language, programmers write programs as a series of mnemonics. Mnemonics are much easier to understand and debug than machine code, giving programmers a simpler way of directly controlling a computer.

Assembly language is still used today, for example, in low-level embedded systems and device drivers. It is faster to run because it doesn’t need as much translation as a high-level language

**High-level languages**

Programmers find *machine code* very difficult to learn, write and *debug*. As a result, the majority of programmers write *programs* in *high-level programming languages*. These languages use English-like statements. For example, *Python* uses 'print', ‘if’, 'input' and 'while' *statements* - all words from the English language - to form *instructions*. In fact, high-level program instructions often look like abbreviated English sentences.





**Commonly used high-level languages**

Many types of high-level language exist and are in common use today, including:

* Python
* Java
* C++
* C#
* Visual Basic.NET
* PHP

Most high-level languages can be used for different purposes, but are best used for the purpose they were created. For example, PHP was created for web development.

**Source code**

*Source code* is the term given to a *set of instructions* that are written in human-readable programming language. The Python program shown before is an example of source code. This code must be *translated* into machine code before the computer can understand and execute it.

**Translators**

Any *program* written in a *high-level language* is known as *source code*. However, computers cannot understand source code. Before it can be run, source code must first be translated into *machine code*.

A *translator* is a program that converts source code into machine code. Generally, there are three types of translator:

* *compilers*
* *interpreters*
* *assemblers*

**Compilers**

A compiler takes the source code as a whole and translates it into machine code all in one go. Once converted, the machine code can be run at any time. This process is called *compilation*.

Compilers have several advantages:

* Compiled programs run quickly, since they have already been translated.
* A compiled program can be supplied as an *executable* file. An executable file is a file that is ready to run. Since an executable file cannot be easily modified, programmers prefer to supply executables rather than source code.
* Compilers *optimise* code. Optimised code can run quicker and take up less *memory* space.

Compilers have several disadvantages:

* The source code must be re-compiled every time the programmer changes the program.
* Source code compiled for one platform will not run on another - the machine code is specific to the *central processing unit’s (CPU)* architecture.

**Interpreters**

An interpreter translates source code into machine code one *instruction* at a time. It is similar to a human translator translating what a person says into another language, sentence by sentence, as they speak. The resulting machine code (sometimes called object code) is then executed immediately. The process is called *interpretation*.

Interpreters have several advantages:

* Instructions are executed as soon as they are translated.
* Errors can be quickly spotted - the moment an error is found, the program stops running and the user is notified at which part of the program the interpretation has failed. This makes interpreters extremely useful when developing programs.

Interpreters also have several disadvantages:

* Interpreted programs run more slowly as the processor has to wait for each instruction to be translated before it can be executed.
* Additionally, the program has to be translated every time it is run.
* Interpreters do not produce an executable file that can be distributed. As a result, the source code program has to be supplied, and this could be modified without permission.
* Interpreters do not optimise code - the translated code is executed as it is.

**Assemblers**

* Assemblers are a third type of translator. The purpose of an assembler is to translate *assembly language* into machine code. Whereas compilers and interpreters generate many machine code instructions for each high-level instruction, assemblers create one machine code instruction for each assembly instruction.



**Computer Systems**

### **Hardware and software**

General purpose computers consist of hardware and software. Hardware is the physical components of the computer, such as the central processing unit (CPU), hard disk, monitor, keyboard and mouse. Software is the programs that run on a computer.

****

### **The purpose of the CPU**

The CPU is the most important hardware component in a computer. It has two main functions:

* to process data and instructions
* to control the rest of the computer system

All programs and data processing are run in the CPU and all hardware components are, to some extent, controlled by it.

## **Von Neumann architecture**

Von Neumann architecture is the design upon which many general purpose computers are based. This architecture uses the stored program concept.The key elements of Von Neumann architecture are:

* data and instructions are both stored as binary digits
* data and instructions are both stored together in the same RAM
* instructions are fetched from memory one at a time and in order - serially
* the processor decodes and executes an instruction, before cycling around to fetch the next instruction
* the cycle continues until no more instructions are available

A processor based on Von Neumann architecture has five special registers which it uses for processing:

* program counter - holds the memory address of the next instruction to be fetched from main memory
* memory address register (MAR) - holds the address of the current instruction that is to be fetched from memory, or the address in memory to which data is to be transferred
* memory data register (MDR) - holds the contents found at the address held in the MAR, or data which is to be transferred to main memory
* current instruction register (CIR) - holds the instruction that is currently being decoded and executed
* accumulator (ACC) - holds the results of processing

The registers and key elements of the Von Neumann architecture all play a part in how an instruction is processed in the fetch-decode-execute cycle.

### **The fetch-decode-execute cycle**

The fetch-decode-execute cycle is followed by a processor to process an instruction. The cycle consists of several steps:

1. The memory address held in the program counter is copied into the MAR.
2. The address in the program counter is then incremented - or increased - by one. The program counter now holds the address of the next instruction to be fetched.
3. The processor sends a signal containing the address of the instruction to be fetched along the address bus to the computer’s memory.
4. The instruction held in that memory address is sent along the data bus to the MDR.
5. The instruction held in the MDR is copied into the CIR.
6. The instruction held in the CIR is decoded and then executed. The results of processing are stored in the ACC.
7. The cycle then returns to step one.

Depending on the type of instruction, additional steps may be taken:

* If the instruction is to transfer data held in the ACC back to RAM, the intended memory address is copied into the MAR. The data to be transferred is copied into the MDR and then transferred to the specified address using the address bus and data bus.
* The executed instruction may require the program to jump to a different place in the program. In this case, the memory address of the new next instruction to be fetched is copied into the program counter. The process then restarts at step one.

## **Common CPU components**

The central processing unit (CPU) consists of a number of components:

* control unit (CU)
* arithmetic logic unit (ALU)
* registers
* cache
* buses
* clock

****

### **Control unit**

The control unit provides several functions. It:

* fetches, decodes and manages the execution of the instructions
* issues control signals that control hardware
* moves data around the system

### **Arithmetic logic unit**

The ALU performs arithmetic and logical operations or decisions. It is where calculations are done and where decisions are made based on logic.

### Registers

Registers are small amounts of high speed memory contained within the CPU. They are used by the processor to store small amounts of data that are needed during processing, such as:

* the address of the next instruction to be executed
* the current instruction being decoded
* the results of calculations

Different processors have different numbers of registers for different purposes, but most have some, or all, of the following:

* program counter
* memory address register (MAR)
* memory data register (MDR)
* current instruction register (CIR)
* accumulator (ACC)

### **Cache**

Cache is a small amount of high speed memory built directly within the processor. It is used to temporarily hold data and instructions that the processor is likely to reuse. This allows for faster processing as the processor does not have to wait for the data and instructions to be fetched from the random-access memory (RAM).

### **Clock**

The CPU contains a clock which is used to coordinate all of the computer's components. The clock sends out a regular electrical pulse which synchronises - keeps in time - all the components.

The frequency of the pulses is known as the clock speed. Clock speed is measured in hertz. The higher the clock speed, the greater the number of instructions that can be performed in any given moment of time.

In the 1980s, processors commonly ran at a rate of between 3 megahertz (MHz) to 5 MHz, which is 3 million to 5 million pulses or cycles per second. Today, processors commonly run at a rate of 3 gigahertz (GHz) to 5 GHz, which is 3 billion pulses or cycles per second.

### **Buses**

A bus is a high speed internal connection. Buses are used to send control signals and data between the processor and other components.

Three types of bus are used:

* Address bus - carries memory addresses from the processor to other components such as RAM and input/output devices.
* Data bus - carries the actual data between the processor and other components.
* Control bus - carries control signals from the processor to other components.

## **Memory**

Memory is the component of the computer that holds data and programs.

Memory is internal to the computer. As a result, data can be read from and written to memory extremely quickly. This gives the central processing unit (CPU) fast access to the data and instructions that the memory holds.

There are three types of memory:

* read only memory (ROM)
* random access memory (RAM)
* cache memory

ROM is non-volatile, whereas RAM and cache memory are volatile. Non-volatile memory keeps its contents even when the computer is switched off. However, volatile memory loses its contents when the computer has no power.

Memory is limited in size, especially when compared with secondary storage.

## Read only memory

Read only memory (ROM) is non-volatile. Its contents are not lost when the computer is switched off.

ROM can be read from, but not written to, hence the term ‘read only’. This makes ROM ideal for storing instructions and data that are needed for the computer to run. These instructions and data are usually programmed by the computer's manufacturer and cannot be overwritten.

The Basic Input Output System (BIOS) is an example of a program that is stored in ROM. The BIOS runs as soon as the computer is switched on. It checks that the hardware is functioning correctly, then loads the computer's operating system (OS). Because the BIOS is always needed, it is stored in ROM.

## **Secondary storage**

Computers use main memory, such as random access memory (RAM) and cache, to hold data that is being processed. However, main memory is volatile - it loses its contents when the computer is switched off. General purpose computers, such as desktop computers and tablets, need to be able to store programs and data for later use.

*Secondary storage* is non-volatile, long-term storage. It is used to keep programs and data indefinitely. Without secondary storage, all programs and data would be lost the moment the computer is switched off.

There are many forms of secondary storage and each type of secondary storage device has its own characteristics. Because all devices are different, some are more suited to certain applications than others.

## Common types of secondary storage

Secondary storagedevices are generally separated into three types:

* **magnetic** storage devices, such as hard disk drives
* **optical** storage devices such as CD, DVD and Blu-ray discs
* **solid state** devices such as solid state drives (SSD) and USB memory sticks

### **Magnetic devices**

Devices such as hard disk drives use magnetic fields to magnetise tiny individual sections of a metal spinning disk. Each tiny section represents one bit. A magnetised section represents a binary '1' and a demagnetised section represents a binary '0'. These sections are so tiny that disks can contain terabytes (TB) of data.

As the disk is spinning, a read/write head moves across its surface. To write data, the head magnetises or demagnetises a section of the disk that is spinning under it. To read data, the head makes a note of whether the section is magnetised or not.

Magnetic devices are fairly cheap, high in capacity and durable. However, they are susceptible to damage if dropped. They are also vulnerable to magnetic fields - a strong magnet might erase the data the device holds.

### **Optical devices**

Optical devices use a laser to scan the surface of a spinning disc made from metal and plastic. The disc surface is divided into tracks, with each track containing many flat areas and hollows. The flat areas are known as lands and the hollows as pits.

When the laser shines on the disc surface, land reflects the light back, whereas pits scatter the laser beam. A sensor looks for the reflected light. Reflected light (land) represents a binary '1', and no reflection (pits) represents a binary '0'.

Optical media also come in different types:

* ROM media have data pre-written on them. The data cannot be overwritten. Music, films, software and games are often distributed this way.
* Read (R) media are blank. An optical device writes data to them by shining a laser onto the disc. The laser burns pits to represent '0's. The media can only be written to once, but read many times. Copies of data are often made using these media.
* Read/write (RW) media works in a similar way to R, except that the disc can be written to more than once.

### **Solid state devices**

Most solid state devices use a type of flash memory to store data indefinitely. They tend to have much **faster access times** than other types of device and, because they have no moving parts, are more **durable**.

Since this type of memory is expensive, solid state devices tend to be smaller in capacity than other types. For example, a solid-state drive that holds 256 gigbytes (GB) might be of a similar cost to a hard disk with several terabytes capacity.

Solid state devices require little power, making them ideal for portable devices where battery life is a big consideration. They are also **portable** due to their small size and durability.

### **Cloud storage**

Storing data at a remote location online is known as cloud storage. When files and data are sent to the cloud, they are actually being sent to a server (or servers) connected to the internet.

Services such as Dropbox, Google Drive, Amazon Drive, OneDrive, Box and iCloud are examples of cloud storage. Files can be uploaded to a folder system and downloaded as required.

In cloud storage users pay for what they use. They need to pay for the storage and the facilities that the cloud company have in place to maintain and manage the service.

In the same way that operating systems use virtualisation to hide the complexities of the hardware from the user, storing data in the cloud also employs abstraction to hide the physical organisation of the hardware storage and the internet connections used to access the storage. The data could be stored on a single folder on a single server or in multiple locations on multiple servers. All the user needs to know is that the data is stored safely and can be retrieved when requested.

Cloud storage has many advantages:

* data can easily be accessed from anywhere with an internet connection
* the business running the cloud storage service manages backups and security
* additional storage can be added easily without having to invest in additional hardware locally

Cloud storage also has a number of disadvantages:

* hackers could take advantage of the fact that data can be accessed from anywhere with an internet connection
* if the internet connection fails, files cannot be accessed
* it relies on other people correctly storing and backing up data

A business that is considering using a cloud computing service will have to think about whether the advantages outweigh the risks. This is a complex decision and will depend on many factors, including the type of data to be stored in the cloud system.

****

## **Embedded systems**

As well as general purpose computers, there are other types of computer systems. The most common of these are known as embedded systems.

An embedded system is a small computer that forms part of a larger system, device or machine. It includes both hardware and software and its purpose is to control the device and to allow a user to interact with it. They tend to have one, or a limited number of tasks that they can perform.

Examples of embedded systems include:

* central heating systems
* engine management systems in vehicles
* domestic appliances, such as dishwashers and TVs
* digital watches
* electronic calculators
* GPS systems
* fitness trackers

Depending on the embedded system’s purpose, they use input devices like sensors and switches to produce output, for example sounding warning buzzers and switching lights on.

Embedded devices are not usually programmable by a user - the programming is usually done beforehand by the manufacturer. However, it is often possible to upgrade the software on an embedded device. For example, fitness trackers are embedded systems, but the software can often be upgraded by connecting the device to a PC and installing the new software.

Embedded systems can have advantages over general purpose computers in that:

* Their limited number of functions means they are cheaper to design and build.
* They tend to require less power. Some devices run from batteries.
* They do not need much processing power. They can be built using cheaper, less powerful processors.

## **Operating systems**

A computer cannot function without an operating system unless it is an embedded system. An operating system is a collection or suite of programs that manages and controls the computer. It keeps the details of how this is done hidden from the user. The main functions performed by an operating system are:

* file management
* process management
* hardware management
* user interface management

Common operating systems include:

* Microsoft Windows
* macOS
* Linux
* Android
* iOS

### **File management**

Computers store data on physical devices, such as hard disk drives. Users do not need to know how and where this data is stored physically, they just need a way of finding and managing that data. This is provided by the file management system.

All data is stored in files, within folders, within drives. The file management system does this by creating a virtual file structure of the physical components. It is a hierarchical structure starting with the root directory, a bit like an upside-down tree. The file system also assigns metadata to each file including:

* date created
* date modified
* last date accessed
* last backup
* file creator user ID
* access permissions
* file size

There are several functions that users can perform on files, for example:

* create
* name/rename
* save
* copy
* move
* delete

### **Process management**

All applications, including web browsers, word processors, spreadsheets and apps, are made up of computer programs. Multiprogramming enables several programs to run at the same time.

Each program is made up of program instructions. When some of these instructions are running, it is called a process. Since main memory is too small to run all the processes, the operating system uses multiprocessing to allocate use of the main memory and the CPU.

Multiprocessing can be used only when there is more than one processor in the system. This is common in most computer systems nowadays. That means that several processes can execute at the same time. A scheduler is used to time the different processes. Interfaces such as Task Manager in Windows let the user see what processes are running and delete them.

There are several different scheduling algorithms, which all have certain similarities:

* they all use a queue to hold waiting processes
* they all use the clock cycle to time the execution of processes
* they all switch between processes

#### First come, first serve (FCFS)

In this algorithm, processes run on a first come, first serve (FCFS) basis. It is based on a first in, first out (FIFO) queue, where the processes arrive, are executed, then leave the queue in order. This is a slow algorithm because all the processes have to wait for the previous one to finish.

#### **Round robin**

Round robin is a much more efficient algorithm. Each process is given a fixed amount of time, then it is switched out and sent to the back of the queue. The next process then has its turn, and so on.

#### **Priority based**

In a priority-based algorithm, each process is given a priority based on several things, including how much time it needs and how much memory it will use. A process with a higher priority will be executed before a process with a lower priority. This is ‘fairer’ than either FCFS or round robin. If there is more than one process with the same priority, then FCFS is employed as well.

### **Hardware management**

The operating system manages input and output from hardware devices and the use of any resources needed by a process.

#### **Input/output management**

Data has to be transferred between input devices, the CPU and main memory, and output devices. Examples of input devices are keyboards, computer mice and microphones. Examples of output devices are monitors, speakers and printers. Some transfer software may already be included in the operating system. Otherwise, device drivers will need to be installed for each new device. Device drivers allow devices created by different manufacturers to communicate.

#### **Resource management**

When a process is running it may need to use resources such as printers, hard disk drives, memory and the processor. The operating system allocates the required resource to the processes that are currently running. Internally, the operating system ensures that any data and/or instructions in main memory do not conflict with each other, and allocates the memory that the process requires.

### **User interface management**

Operating systems provide interfaces to users of the computer system, allowing them to communicate with the hardware. The user doesn’t need to know about the hardware, so the operating system uses virtualisation (zr6ct39) to give the user the impression that they are interacting directly with the hardware. The user sees a virtual machine giving access to logical devices. The operating system translates the user’s actions to match to the physical devices. In effect, the operating system uses abstraction to hide complexity from the user.

Users are granted different access rights to be able to perform tasks. In a networked system this would allow access to the data and applications they need and no more. Security features, such as user names and passwords, are included to protect the systems. The operating system can keep track of different users and what they are doing on the computer.

System administrators have special privileges to maintain the computer system. This may include being able to add users and give them access to different applications.

There are two main types of interface:

* graphic user interfact (GUI)
* command line interface/interpreter (CLI)

#### Graphical user interface

A GUI is familiar to most users of PCs and laptops. GUIs feature a desktop view where everything is displayed and use WIMP (windows, icons, menus and pointer) to describe the features of this interface. Applications run in windows and all objects (apps, hardware and files) are represented by icons. Application features are accessible through the use of menus. Users interact with the interface by using a mouse and onscreen pointer.

GUIs are powerful and easy to use but require a lot of processing power.

#### Command line interfaces

CLIs are text-based. Users control the computer by typing in commands.

CLIs require little processing power and are extremely powerful, but it can take longer to learn how to use a CLI than a GUI. Originally, most interfaces were CLIs. They still exist within modern operating systems, for example the command prompt app in Windows, and Terminal in macOS.

#### Other interface examples

Mobile user interfaces are similar in many ways to GUIs except that they respond to touch. Fingers are used to open apps and interact with them. Gestures such as swiping are used to scroll within documents. Pinching and stretching are used to resize images.

Mobile user interfaces are found on smartphones and tablets and are single-user operating systems, as opposed to multiple-user operating systems that allow different users to log on.

Voice user interfaces are becoming increasingly popular as a natural way of interacting with technology. They are used by voice device assistants. Mobile user interfaces can also respond to voice commands.

## **Utility software**

Utility software helps maintain the system. Utility software is used for:

* compression
* defragmentation
* backing up
* anti-virus
* anti-spyware

Most basic utility software is part of the operating system, but additional utilities can also be installed separately.

### Managing, repairing and converting files

#### Managing files

There are several functions that can be used to manage files. These include the following.

* Creating a new file or folder, and storing the file data there.
* Naming/renaming files or folders. Two files cannot have the same name in a single folder, but they can if they are in different folders.
* Saving a file, to update an existing version or to save a different version.
* Copying a file to a different folder or to an external device. The original version is retained.
* Moving a file to a different folder.
* Deleting a file that is no longer required. On a PC or laptop it is usually possible to retrieve the file if it has been deleted in error.

#### Repairing files

Files can be corrupted due to:

* the computer crashing as a file is saved
* physical problems on a storage device
* a malware or virus attack

A corrupt file can sometimes be repaired, and most operating systems have this utility. Another utility can detect and, if possible, recover from physical errors on the disk. It scans disk surfaces for defects and marks those sections as unavailable to prevent the rewriting of data and data loss.

It is sensible to always make a backup of files in case they cannot be recovered.

#### Converting files

A file conversion utility lets users save a file as a different file type. This is useful, for example, when:

* translating a music file to run using a different audio application
* compressing the size of an image file
* changing a file to PDF format so that the contents cannot be altered

### **Data compression**

Compression software reduces the size of a file stored on secondary storage.

Smaller files are easier to transmit across a network as they require fewer packets to be sent. Their reduced size also means more files can be stored in any given area of storage.

### **Disk defragmentation**

When a file is stored on a hard disk it is not stored as a whole file, but as a series of segments. Sometimes the segments run together in sequence (see File 1) and sometimes the segments are split up over a disk (see File 3). This is known as fragmentation.





## The importance of network security

Networks operate on the principles of communication and sharing. Unfortunately, these principles mean that network traffic and data risk being accessed by people who have no authority to do so, ie hackers.

A network and the servers connected to it are likely to contain large amounts of information. This information could be valuable and some of it is likely to be private and confidential.

There are many ways to secure a network but the starting point is only allowing network access to authorised people. This is known as authentication and validation. A person will authenticate themselves using a username and password. Before being given access to the network, a server will validate their username and password against a list of authorised users.

There are other methods of authentication and validation such as using PINs, a fingerprint or facial recognition.

Authentication of users isn’t enough to fully secure a network. Other important security measures are access control, firewalls and physical security.

### Access control

Access control determines the facilities a user has access to, such as:

* software
* email
* internet
* documents and data
* the ability to install and/or remove software
* the ability to maintain other users' accounts

A network manager should restrict most users to allow them to access only the facilities they need. For example, an office worker might need access to productivity software, email and the internet, but not to install software or access to other users' accounts. Restriction limits the actions a user can take, reducing the potential of threats. The restrictions can be precise, allowing different people access to different sets of files or information.

### Firewalls

A firewall is a tool that monitors traffic going into and out of a computer or network, and either allows the traffic to pass through or blocks it.

The decision to allow or block traffic is based on rules, known as the firewall policy. For example, some programs, such as email clients and web browsers, have legitimate cause to send a transmission. These programs are known to the system and the firewall policy allows their communications. However, any transmissions that are not sent from, or to, known and allowed sources are blocked.

Firewalls can be hardware-based or software-based. Hardware firewalls tend to be more expensive, but are more effective.

### Physical security

Physical security means restricting physical access to important parts of a network. For example, servers should be kept in a locked, secure room that can only be accessed by authorised people, such as the network manager.

This is important as anyone with physical access to a server could remove or access the hard disks containing private and confidential information.

## Identifying vulnerabilities

One of the roles of a network manager is to ensure the network is safe from attacks and threats of any kind. There are many techniques that can be used to help keep a network safe, including:

* penetration testing
* ethical hacking
* commercial analysis tools
* network and user policies

### Penetration testing

The purpose of penetration testing is to determine how resilient a network is against an attack. It involves authorised users - sometimes an external party or organisation - who probe the network for potential weaknesses and attempt to exploit them. Software that enables network managers to test a network's resilience themselves is also available.

### Ethical hacking

Ethical hackers attempt to access a network using the same tools and techniques as a hacker. However, an ethical hacker isn’t attempting to steal information, but is looking for weaknesses in the security of the network. Any weaknesses found can then be fixed. An ethical hacker might be employed by the business that owns the network being tested or they might work for a security company hired by network owners.

### Commercial analysis tools

A commercial analysis tool is usually a software package that is used to find security weaknesses within a network. The analysis tool could check for a range of issues such as unpatched software, weak passwords or poorly configured firewalls. The software produces a report so that the network manager can fix the problems.

### Network and user policies

Users of a network are often the source of threats, whether accidental or deliberate. A network manager should have network and user policies that ensure:

* users have secure, hard-to-guess passwords which meet specified conditions and are changed regularly
* users cannot connect unauthorised equipment to the network, such as USB memory sticks, smartphones and tablets
* levels of access are given which allow only authorised users to access sensitive data
* a regular backup procedure is in place
* a disaster recovery procedure exists in case of data loss
* regular maintenance is undertaken, including applying software upgrades and security patches to equipment
* physical access to servers is prevented
* a high level of security is maintained with up-to-date anti-virus software and firewalls

## Types of cyberattack

A cyberattack is an attempt to gain access to, steal, modify or delete data on a network. A cyberattack might make the network inaccessible, stopping a business from running efficiently. There are many different types of cyberattack. Some rely on technical weaknesses while others rely on social engineering.

### Social engineering attacks

Social engineering covers a wide range of activities including phishing and shoulder surfing.

#### Phishing

Phishing involves using emails or fake websites that try to trick users into giving away personal details. A phishing email pretends to be a genuine message from, for example, a bank, and tries to deceive the user into following a link to a website that looks like the real company. However, it is a fake website designed to catch data such as bank account numbers and security codes.

#### Shoulder surfing

Shoulder surfing is the act of gaining information by directly observing someone. The most common form of shoulder surfing is when someone watches over the shoulder of a person entering their PIN at a cashpoint. Once they’ve got the PIN, they could then steal the wallet or purse containing the bank card. The bank card and PIN could be used to withdraw cash or make purchases before the owner of the bank card realises it has been stolen.

### Technical attacks

Technical attacks rely on some kind of technical weakness with software or hardware. As in social engineering, there are a wide range of attacks including exploiting unpatched software, using malicious USB devices and eavesdropping.

#### Unpatched software

Most software will regularly receive patches to add new features, fix bugs or improve security. Unpatched software hasn’t had these patches - updates - applied. This means any bugs or security weaknesses can be used by a hacker to gain access to information.

#### Malicious USB devices

USB memory sticks are a common sight and many people use them in their day-to-day jobs. It is possible to get malicious USB devices - many of which look like normal USB memory sticks - that may infect a network with malware. They could also run some other technical attack against a network. The hacker would then be able to gain access to information from the network.

#### Eavesdropping

Eavesdropping is when a hacker intercepts information being sent to or from a network to a device or another network. An example of this is wiretapping, where communications are monitored.

#### Other digital devices

Digital devices such as webcams and DVD recorders can also be used in a cyberattack on a global scale. They become part of a botnet - a collection of devices connected via the internet which have been infected by malware. Hackers can use these devices to attack global organisations.

### **Backing up**

Data can be lost accidentally or deliberately. A user may accidentally delete or overwrite a file or a hard disk may fail, preventing access to any files stored on it. A hacker or malicious user may deliberately delete or overwrite data.

To prevent data loss, regular copies of the data should be made. A copy of data is known as a backup. Backups can contain a copy of all files on a computer, or just those specified by a user.

Network managers make regular backups of all files on a network using backup software. The software automatically makes a backup at a scheduled time of day, usually during the evening when the network is quiet. Backups are usually made to a high capacity secondary storage device or to the cloud.

If data loss occurs, data can be retrieved from the backup. The software allows all backed-up files to be retrieved, or just specified files. This can be done with a system restore.

Two types of backup are possible:

* full backup
* incremental backup

A full backup involves making a copy of every file on the computer or network. This can require a lot of storage space and can be time consuming to make.

Incremental backups take a copy of any new files created since the last backup and of any files that have been edited, such as user documents.

Most network managers make an initial full backup and then switch to daily incremental backups. This way all data is backed up and daily backups are small and less time consuming.

### **Anti-virus**

Anti-virus software scans a computer system, either when activated or automatically at a specified date and time, and identifies potential viruses. Many anti-virus software programs can also delete or fix potential threats once they have been identified.

### **Anti-spyware**

Spyware is installed on a computer without the knowledge of the user. It is usually designed to collect data about the user, often personal information or bank details, which can then be passed on and used in identity theft. Anti-spyware software can find, then delete, the spyware. Some versions can prevent it being installed in the first place.

## **Encryption**

Encryption is the process of encoding data or a message so that it cannot be understood by anyone other than its intended recipient. In computer processing, encryption means that data can be stored and transmitted securely by the sending computer to the receiving computer. The data or message is encrypted using an encryption algorithm. The opposite of encryption is decryption.

An encryption key is a piece of information - usually random characters - used by the software algorithm to encrypt data or a message into a form which is unreadable (encryption) and allow the data or message to be made readable again (decryption).

Here are the top five facts you need to know about encryption.

 **One**: it’s a way of concealing a message so it can be sent secretly.

Encryption is a process of disguising a message or some data so that it cannot be easily understood, especially to prevent unauthorised access. A message for all to see is called plaintext. But once encrypted, it’s known as ciphertext.

 **Two**: encryption is secure when used correctly.

A cryptosystem is the method you use to encrypt and decrypt messages. A Dutchman called Auguste Kerckhoffs came up with this principal - a cryptosystem should be secure, even if everything about the system is public knowledge, which means every detail about your cryptosystem can be public and it will still remain secret. Every detail, except for one important thing.

**Three**: you need a key.

The key tells you how the message has been jumbled up or encrypted. Get the key, unlock the message.

**Four**: Julius Caesar used encryption.

One of the oldest and simplest forms of encrypted writing is called the Caesar cipher. It’s very simple and works by realigning or shifting the alphabet. Each letter is replaced with one further up or down the alphabet.

For example, shifting the alphabet two letters forward would mean that A becomes C, B becomes D, C becomes E and so on. The key to unlock the code is whatever number of letters the alphabet has been shifted. The key in this case would simply be forward or plus two. You can see that the letter Y becomes A and Z becomes B.

The letters at the end of the alphabet have been shifted round to overlap letters at the start of the alphabet. This is known as a wrap around. As there are 26 letters in the alphabet, it means that this type of encryption is very easy to solve. You would only need to try a maximum of 25 combinations before you crack it.

**Five**: asymmetric is more sophisticated than symmetric.

Symmetric encryption, like the Caesar cipher, uses just one key to hide and read a message. It assumes both parties already share a secret key. The problem is that if that key becomes public then all messages can be unlocked.

 A more sophisticated form of encryption is called asymmetric encryption.

To do it, you need a pair of keys: one public and one private. The two keys are uniquely paired from the time they are created. You share your public key with the world at large. Then anyone who wants to send you a message uses that public key to encrypt it. And here’s the trick, only your private key can unlock a message that has been encrypted using your public key. So, as long as you keep your private key a secret then nobody else can read a message that has been encrypted just for you. Most websites use this form of encryption.

Emails, photos, videos, documents, calls, texts, voice messages, instant messages, status updates, cryptocurrency and credit card numbers are all encrypted.

Cryptography works, but only if you keep your secret key secret.

### **Asymmetric encryption - public and private keys**

* Encryption is of little use if unauthorised users know the key. They can use it to decrypt any message that is encrypted with that key.
* One way around this issue is to use an algorithm that generates two keys - a public key and a private key.
* A public key can be given to anyone. Anyone can then use this key to encrypt a message. However, the public key cannot decrypt a message - only the second, private key can do that. So long as the private key is never given out, messages will stay safely encrypted.
* An encryption method that uses public and private keys in this way is known as asymmetric encryption. There are a number of different asymmetric encryption algorithms that are in use.
* Encryption methods that use the same key for encryption and decryption are known as symmetric encryption methods.