GCSE COMPUTING KNOWLEDGE ORGANISER

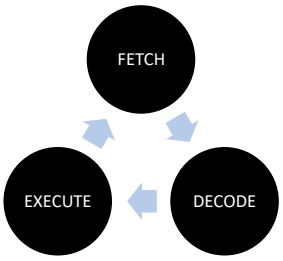
1.1 SYSTEMS ARCHITECTURE

1.1.1 ARCHITECTURE OF THE CPU

The purpose of the CPU:

- ❑ The fetch-execute cycle

- Data and instructions **FETCHED** from main memory
- They are then **DECODED** and **EXECUTED**
- This is carried out in a continuous cycle



REVISION NOTE

You should learn what each register does and its role in the fetch execute cycle

Common CPU components and their function:

- ❑ ALU [Arithmetic and Logic Unit]
- ❑ CU [Control Unit]
- ❑ Cache
- ❑ Registers

- **ALU** performs calculations and logic checks.
- It may take several F-E-Cycles for a calculation to be finished.
- Intermediate results are stored in the **accumulator**
- **Cache** is VERY FAST memory.
- Instructions that are carried out frequently are stored there so that they do not have to be FETCHED [saving time]
- **Registers** = small amounts of high-speed memory contained within the CPU. Registers store data that is needed during the F-E-C

Von Neumann Architecture:

- ❑ MAR (Memory Address Register)
- ❑ MDR (Memory Data Register)
- ❑ Program Counter
- ❑ Accumulator

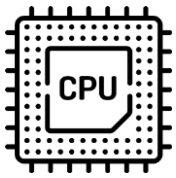
- **John Von Neumann** was a Hungarian mathematician who developed the idea that a computer could be used for many purposes and not just one.
- This was called the **stored program concept**.
- A processor based on **Von Neumann's architecture** would use memory to store data and instructions and would use the **fetch execute cycle** to retrieve and process instructions.
- Von Neumann's architecture makes use of a number of registers…

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	**MDR** -holds the contents found at the address held in the MAR, or data which is to be transferred to primary memory	**PC** - holds the memory address of the next instruction to be fetched from primary memory	**ACCUMULATOR –** holds data while it is being processed and while

1.1.2 CPU PERFORMANCE

How common characteristics of CPUs affect their performance:

- ❑ Clock speed
- ❑ Cache size
- ❑ Number of cores

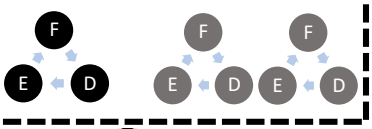


- The **clock** coordinates all the computer's components.
- It sends out a pulse the synchronises each component – the **frequenc**y of the pulses is known as the **clock speed**.
- It is measured in **Hertz**.
- *The higher the frequency, the more instructions can be processed in a given time*

- Each processing unit inside a CPU is called a **CORE**.
- **Each core can carry out the fetch execute cycle**
- *The more cores a CPU has, the more instructions it can process in a given time (i.e. **PARALLEL PROCESSING**)*

CACHE is very fast (and expensive) memory that can store frequently used data or instructions

single core CPU = 1 instruction **VS** dual core CPU = 2 instructions



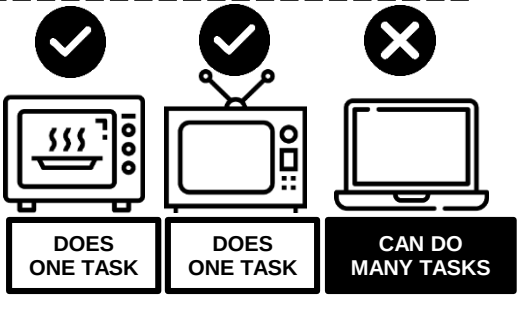
1.1.3 EMBEDDED SYSTEMS

The purpose and characteristics of embedded systems

Example of embedded systems

- **Embedded systems** are small computer systems built inside larger devices or pieces of equipment
- They are designed to do one specific task (rather than range of task)
- Embedded systems have a simple user interface
- In addition, the software used to control or run the system is also very basic

WHICH OF THESE ARE ARE NOT Embedded Systems?

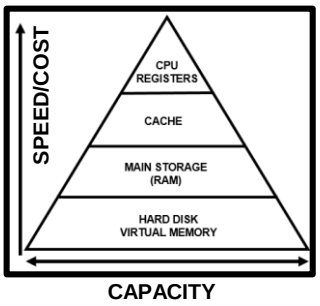


GCSE COMPUTING
KNOWLEDGE ORGANISER

1.2 MEMORY AND STORAGE

1.2.1 PRIMARY STORAGE (MEMORY)

The need for primary storage
The difference between RAM and ROM
The purpose of RAM in a computer system
The purpose of ROM in a computer system
Virtual memory



This shows speed vs capacity of primary/secondary storage

-**PRIMARY STORAGE** is used to store programs and data *currently used* by the computer. When a user needs to run a program, it is loaded from disk to primary storage.
-Another term for primary storage is **RAM** or **Random Access Memory**. It is given this name because data can be stored anywhere within the available memory.
-**RAM** is **volatile** (i.e. any data stored in RAM is lost when the device is powered off)
-**ROM or Read Only Memory** is **non-volatile** (i.e. any data stored in RAM is not lost when the device is powered off) – it is stored **permanently**.
-**ROM** can be used to store the **BIOS** (i.e. the program that boots up and loads the **Operating System** when the computer turned on)

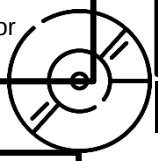
- **VIRTUAL MEMORY** is used when the computer is short of RAM. This involves the hard disk being used as memory instead of RAM.
- *This is not ideal as the speed of a hard disk is MUCH slower then RAM.*

1.2.2 SECONDARY STORAGE)

The need for secondary storage
Common types of storage:

- ❑ Optical
- ❑ Magnetic
- ❑ Solid state

- **Optical Storage** includes CD, DVD and Blu-ray. Data is written to optical storage media using a laser.
- The capacity of this type of media ranges from 640 megabytes (CD) to 50 gigabytes (Blu-ray)
- Since it involves the use of moving parts, access/data transfer speeds are slower than for other types of media.



Examples – CD, DVD, Blu-ray

- **Magnetic Storage** media include hard drives and tape and can have a huge capacity (100's of terabytes)
- Magnetic storage media devices involve the use of moving parts.
- This means that they have comparatively slow data read and write speeds and can be prone to damage.

Examples – hard drive, magnetic tape, external hard drive

- **Solid state** media is also known as electrical or flash storage.
- Solid state has the fastest transfer speed out of all the three types of media, since it features no moving parts
- This also makes it more robust than other forms of storage and in addition they consume less power
- However this form of storage offers lower capacity than other forms of media and is still comparatively expensive.

Examples – Solid State Drive, flash drive, SD Card

Criteria for choosing secondary storage

Criteria	Meaning
Cost	How much does it cost per GB of storage?
Capacity	How much space is there to store files?
Speed	How fast can it read / write data?
Portability	Can it be carried easily or is it a device that is hard or impractical to carry/move?
Durability	How robust is it? Will it break or damage easily?
Reliability	How likely to fail is it? How long will it last?

REVISION NOTE
When recommending a method of secondary storage, always consider the context in which the data will be used

1.2.3 UNITS

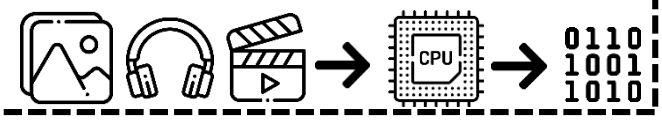
The units of data storage:

- ❑ Bit
- ❑ Nibble (4 bits)
- ❑ Byte (8 bits)
- ❑ Kilobyte (1,000 bytes or 1KB)
- ❑ Megabyte (1,000 KB)
- ❑ Gigabyte (1,000 MB)
- ❑ Terabyte (1,000 GB)
- ❑ Petabyte (1,000 TB)

How data needs to be converted into a binary format to be processed by a computer
Data capacity and calculation of data capacity requirements

Computers are electrical devices; their components are made up of millions of circuits. Each circuit contains switches which can be Either 'on' or 'off'. These can be represented by the values 1 and 0. **This is called binary.**

ALL data is stored and processed in binary form – this includes text, images, sound and video.



1.2.4 DATA STORAGE

Numbers

- ❑ How to convert positive denary whole numbers to binary numbers (up to and including 8 bits) and vice -versa
- ❑ How to add two binary integers together (up to and including 8 buts) and explain overflow errors which may occur
- ❑ How to convert positive denary whole numbers into 2-digit hexadecimal numbers and vice versa
- ❑ How to convert binary integers to their hexadecimal equivalents and vice versa
- ❑ Binary shifts

Characters

- ❑ The use of binary codes to represent characters
- ❑ The term 'character set'
- ❑ The relationship between the number of bits per character in a character set, and the number of characters which can be represented , e.g.:
  - ASCII
  - Unicode

Every character (letters, numbers, symbols) sent to the computer or typed in, is stored as 7-bit binary code*. For example, if the user types in the message below*, **H** is represented by the number '072'. This character set is called **ASCII**

```
Hello, world
>_
```

```
你好，世界
>_
```

UNICODE uses 16 bits to allow an even wider range of characters to be stored, including one used for foreign languages:

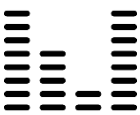
Images

- ❑ How an image is represented as a series of pixels, is represented in binary
- ❑ Metadata
- ❑ The effect of colour depth and resolution on:
  - The quality of the image
  - The size of a sound file

Sound

- ❑ How sound can be sampled and stored in binary form
- ❑ The effect of sample rate, duration and bit depth on;
  - The playback quality
  - The size of a sound file

Sound waves are **ANALOGUE** and must be converted in to **DIGITAL** (0's and 1's) in order to be stored/ processed by computer. This is called **SAMPLING**.



-The height of a sound wave is its **AMPLITUDE**.
-The **SAMPLE RATE** is the number of samples captured per second.
-**SAMPLE RESOLUTION** is the number of bits used to capture the sound

1.2.5 COMPRESSION

The need for compression

- ❑ The need for compression
- ❑ Types of compression;
  - Lossy
  - Lossless

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

To convert 25 to binary:
- ✓ Start with the column nearest 25 (i.e. 16 or 2^4)
- ✓ Put a one in that column
- ✓ Check each remaining column, adding either a 1 or 0 onto the end of our number
- ✓ In this case **16 + 8 + 1 = 25** ($2^4 + 2^3 + 2^0$)
- ✓ So… 25 in binary is**11001**

To convert 101010 into decimal:
- ✓ Work which column the left digit is in (i.e. 32 or 2^5)
- ✓ Work left to right – add up the column value each time there is a 1
- ✓ For example: 32 + 8 + 2 = **42**)

Hex	Dec
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
A	10
B	11
C	12
D	13
E	14
F	15

Converting to/from Hex

To convert decimal to hex:

✓ First convert the number to binary:

i.e. 25 in binary = **00011001**

✓ Then split the number into two groups of 4 bits:

i.e. **0001 1001**

✓ Then convert each group of 4 bits to HEX:

i.e. **0001 = 1**
1001 = 9

Then join (don't add) both digits together:

i.e. 1 + 9 = **19**

To convert hex to decimal, just work through this process backwards

Binary arithmetic 0110 1001 1010

Use binary shift to:

- **Multiply**

```
00110111 x 2=
01101110
[left shift]
```

- **Divide**

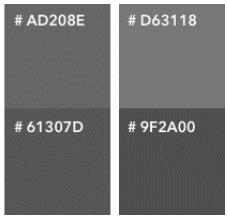
```
00110111/2=
00011011
[right shift]
```

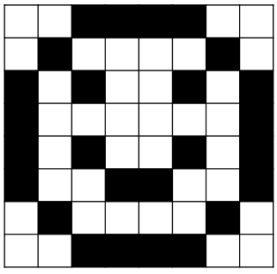
Rules for binary addition		
00 + 00 =	0	No remainder
00 + 01 =	1	No remainder
01 + 01	10	Carry 1
10 + 01	11	Carry 1

If there are insufficient bits to store the answer, this causes **OVERFLOW**

Hexadecimal has many uses in computing**:**
- **assembly language**,
- to store a **MAC Address**
- representing colour codes

Hexadecimal numbers can be represented in fewer digits then in binary making them easier for humans to remember and more economical in terms of storage



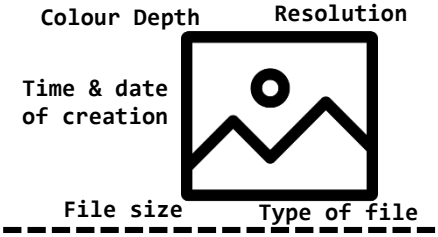


0	0	1	1	1	1	0	0
0	1	0	0	0	0	1	0
1	0	1	0	0	1	0	1
1	0	0	0	0	0	0	1
1	0	1	0	0	1	0	1
1	0	0	1	1	0	0	1
0	1	0	0	0	0	1	0
0	0	1	1	1	1	0	0

00111100
01000010
10100101
10000001
10100101
10011001
01000010
00111100

-Bitmap images are made up of individual **pixels**. The more pixels stored in an image, the higher the detail (**resolution**) will be.
-Each pixel will be represented in binary as a 1 (on) or a 0 (off). These binary digits are combined into binary numbers that can be stored by a computer.
-Colour images need additional binary code to store the colour. The more bits available to store the colour, the wider the possible colour range. This binary value is called **colour depth**.

METADATA = ***'data about data'*** i.e. additional information stored when an image files is saved



COMPRESSION can be applied to any file type and is used to reduce the size of a file. This is useful when files need to be uploaded/downloaded to/from the internet or sent via email.

LOSSY	LOSSLESS
File size is reduced at the expense of quality	File size is reduced with no loss of quality

GCSE COMPUTING
KNOWLEDGE ORGANISER

1.3 COMPUTER NETWORKS CONNECTIONS AND PROTOCOLS

1.3.1 NETWORKS AND TOPOLOGIES

Types of network:
- ❑ LAN (Local Area Network)
- ❑ WAN (Wide Area Network)

Factors that affect the performance of networks

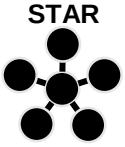
The different computers in a client-server and peer-to-peer network

The hardware needed to connect stand-alone computers into a Local Area Network:
- ❑ Wireless access points
  - ❑ Routers
- ❑ Switches
- ❑ NIC (Network Interface Card)
- ❑ Transmission media

The internet as a worldwide collection of computer networks:
- ❑ DNS (Domain Name Server)
- ❑ Hosting
- ❑ The Cloud
- ❑ Web servers and clients

Star and Mesh network topologies

STAR



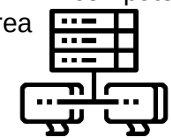
All computers connect to a central hub/ server/switch

LAN/WAN or VLAN - Only one 'node' connected to router

MESH

LAN – computers and devices connected over a single site or small geographical area

WAN – computers and devices connected over a wider area



Peer-to-Peer: Devices connected directly (with no server)

Client-Server: Computers (clients) connected to a central server which could provide services like:
- Shared files
- Internet access
- Share programs
- Shared peripherals (i.e. printers)

-All devices need a **NETWORK INTERFACE CARD** in order to connect to a network.

-This contains a **MAC ADDRESS** – a code which uniquely identifies a device on a network.

5B-8C-2A-11-2F

-ROUTERS connect devices across a WAN (including the internet)

-A **SWITCH** allows devices to connect within a LAN

-Physical networks are possible with **TRANSMISSION MEDIA** such as **ETHERNET CABLES** (i.e. twisted-pair copper, coaxial, fibre-optic)

THE INTERNET is basically a giant **WAN** – it is a *network of networks*. Any computer or device connected to the internet has an IP address which acts as its "address" on the internet

1.3.2 WIRED AND WIRELESS NETWORKS, PROTOCOLS AND LAYERS

THE CLOUD refers to services that allow users to use software or store files on a servers owned and run by a third party. **iCloud** and **Google Drive** are examples of cloud based services and users need an internet connection in order to access these services.

REVISION NOTE

Learn the advantages and disadvantages of the different network types and connectivity methods

Modes of connection:
- ❑ Wired
  - ❑ Ethernet
- ❑ Wireless
  - ❑ Wi-Fi
  - ❑ Bluetooth

Encryption

SYMMETRIC ✗
Risky - Single key used to both encrypt & decrypt the message

ASYMMETRIC ✓
Safer as it uses a PRIVATE & PUBLIC key

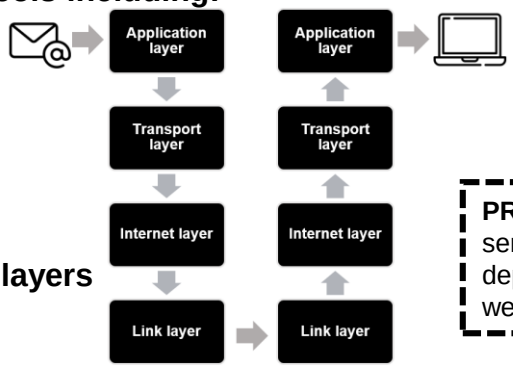
When a user types in a **URL**, this is sent to a **Domain Name Server** on the internet . DNS servers look up the URL and translate this into an IP address

208.80.154.224

IP addressing and MAC addressing standards

Common protocols including:
- ❑ TC/IP
- ❑ HTTP
- ❑ HTTPS
- ❑ FTP
- ❑ POP
- ❑ IMAP
- ❑ SMTP

The concept of layers



Wireless	Up to 100m range	Ideal for connecting personal devices	Does not need a router
Wireless	150-350 ft range	Slower than wired **ETHERNET** connections	Needs a wireless router and uses 2.4 & 5ghz frequencies

PROTOCOLS are rules which allow different devices to send/receive data to/from each other. Different protocols exist depending on (i.e. uploading or downloading data, displaying a webpage, sending/receiving an email

GCSE COMPUTING KNOWLEDGE ORGANISER

1.4 NETWORK SECURITY

1.4.1 THREATS TO COMPUTER SYSTEMS AND NETWORKS

Forms of Attack:

- ❑ Malware
- ❑ Social engineering, e.g. phishing, people as the 'weak point'
- ❑ Brute-force attacks
- ❑ Denial of service attacks
- ❑ Data interception and theft
- ❑ The concept of SQL injection

- **MALWARE** is software which can cause damage to a computer. A computer or system could become infected by a **VIRUS**, **WORM** or **TROJAN**
- Malware is often hidden inside other programs (usually ones that have been illegally)

SOCIAL ENGINEERING involves exploiting human weaknesses in order to gain entry to computer system. This can be done in a number of ways

BRUTE FORCE ATTACKS involve a hacker attempting to guess a users password using trial-and-error. They may use a computer program to do this, since it could try millions of combinations very quickly.

-PHISHING emails are sent by criminals and are designed to steal money or login details.
- They contain links or attachments which, if clicked on or downloaded, allow the criminal to access what they want

FROM: **webflix@xyz123.com**
TO: **a.dent@webmail.com**

WEBFLIX

Deer Customer

Payment Issues

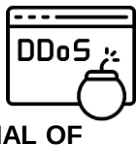
Were having problems taking payment using the [illegible] provided. Please click **here** to enter your current bank details

Yours sincerely

Webflix Customer Support

HOW CAN YOU SPOT A PHISHING EMAIL?

-Spelling mistakes
-Suspicious origin email address
- Impersonal (i.e. no name used)
- Asks for personal information
- Contains links or attachments



DoS (DENIAL OF SERVICE) attacks are designed to "crash" a network or website. Criminals do this by bombarding it with so much 'traffic' that it cannot function properly.

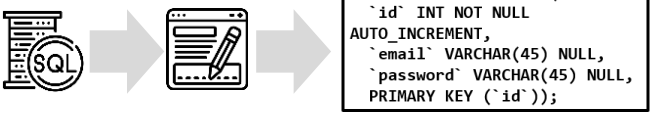
REVISION NOTE

Learn the differences between the different forms of malware and make sure that you are clear about how they spread and what effects they might have

Other methods of **DATA INTERCEPTION and THEFT** could be non – technical; for example, **SHOULDERING** (looking over someone's shoulder when they enter data) or finding private information (like login details) on discarded documents)

SQL INJECTION can be used to hack poorly coded websites. A hacker could use a database language called **SQL** to gain entry to a websites database (for example, on an online shopping site) by typing SQL code into a web form.

1.4.2 IDENTIFYING AND PREVENTING VULNERABILITIES




```
CREATE TABLE `users` (
  `id` INT NOT NULL
AUTO_INCREMENT,
  `email` VARCHAR(45) NULL,
  `password` VARCHAR(45) NULL,
  PRIMARY KEY (`id`));
```

Common prevention methods:

- ❑ Penetration testing
- ❑ Anti-malware software
- ❑ Firewalls
- ❑ User access levels
- ❑ Passwords
- ❑ Encryption
- ❑ Physical security

Companies can hire employ hackers to try and find weaknesses in their systems. This is called **PENETRATION TESTING**

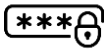
PHYSICAL SECURITY includes methods such as use of CCTV, security guards and locked doors.

FIREWALLS can be either hardware or software – they are designed to intercept data packets before they are received from or sent to the internet

ANTI-MALWARE SOFTWARE can scan files and programs for viruses. They need to be kept up to date as new malware is developed all the time.

Network administrators can set different levels of **USER ACCESS LEVELS** – for example, some users may be able to install software, while others may only be able to view files

Users should be made to set secure **PASSWORDS** – for example, containing combinations of numbers, letters and characters. Users could also protect their files using **ENCRYPTION**



GCSE COMPUTING
KNOWLEDGE ORGANISER

1.5 SYSTEMS SOFTWARE

1.5.1 OPERATING SYSTEMS

The purpose and functionality of operating systems:

- ❑ User interface
- ❑ Memory management and multitasking
- ❑ Peripheral management and drivers
- ❑ User management
- ❑ File management

OPERATING SYSTEMS act as an interface between the user and the computer hardware.

Operating systems have two types of interface;
COMMAND LINE INTERFACE (which uses text based commands)
GRAPHICAL USER INTERFACE (which uses icons and pointers)

```
Volume in drive C is MS-DOS-6
Volume Serial Number is 1E49-15E2

dir /s Windows
C:\>_
```

In order for a computer to make use of additional hardware (such as mice, keyboards and printers) **PERIPHERAL MANGEMENT** is needed. The OS will manage these devices, which require small programs called **DRIVERS** in order to function.

	COMMAND LINE INTERFACE	GUI
Ease of use	✗	✓
Flexibility	✓	✗
Heavy use of system resources	✗	✓

FILE MANAGEMENT allows users to search for, create, delete and organise files and folders. It also allows access rights to files and folders to be changed (i.e. to read only)

USER MANAGEMENT means that multiple user accounts can be added to one machine. Users will have different access rights.

One of the most important roles of the OS is **MEMORY MANAGEMENT** and **MULTITASKING.** Multitasking allows multiple files and programs to be resident in memory at one time. This allows users to switch rapidly between different programs.

A computer's memory is organised into "blocks". The OS moves programs and files in and out of main memory as and when they are needed.

MAIN MEMORY

PROGRAM 1	PROGRAM 2	FILE 1	PROGRAM 3	FILE 2

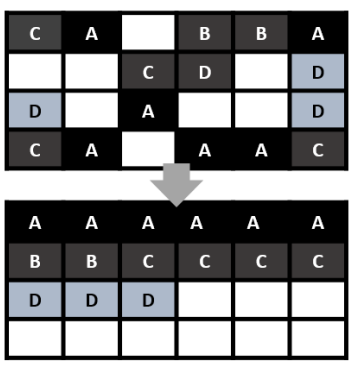
1.5.2 UTILITY SOFTWARE

The purpose and functionality of utility software
Utility system software:

- ❑ Encryption software
- ❑ Defragmentation
- ❑ Data compression

UTILITY SOFTWARE is software used to keep a computer running with optimum efficiency. Many OS contain built-in utility programs and these can also be purchased as third-party software.

DATA COMPRESSION utilities can be used to reduce the size of a file. This is useful when sending files electronically. Files compressed with this type of software need to have their contents extracted (using the same utility) before they can be used again.



Files and programs are stored in "blocks" on a hard drive. Drives become fragmented as files and programs are removed and added over time.

DEFRAGMENTATION groups files/programs and free space together – this decreases the time that the disk has to spend load/saving.

REVISION NOTE
SSD's do NOT need to be defragmented since electrical storage works differently to magnetic storage

ENCRYPTION SOFTWARE uses **ALGORITHMS** to turn **PLAINTEXT** files into **CIPHERTEXT.** This means that the contents of an encrypted file cannot be read without the use of the **KEY** that was used to encrypt it.

GCSE COMPUTING
KNOWLEDGE ORGANISER

1.6 ETHICAL LEGAL AND ENVIRONMENTAL IMPACTS OF DIGITAL TECHHNOLOGY

1.6.1 ETHICAL, LEGAL, AND ENVIRONMENTAL IMPACT

Impacts of digital technology on wider society including:

- ❑ Ethical issues
- ❑ Legal issues
- ❑ Cultural issues
- ❑ Environmental issues
- ❑ Privacy issues

ETHICS = *Our principles, the things that influence our choices and behaviour*
CULTURE = *Our way of life, including customs and beliefs*

REVISION NOTE
You need to be able to talk about each of these issues IN DETAIL, considering both advantages & disadvantages of each

HEALTHCARE
✓ Use of AI to diagnose illness
✗ Poorer nations cannot afford technology

SOCIAL MEDIA
✓ Great for keeping in touch
✗ Can be used for bullying/trolling

DRIVERLESS CARS
✓ Can auto-navigate and do not get tired on long journeys
✗ Will computers cause accidents?

ENVIRONMENT
✓ Networking removes the need for travel
✗ Waste and pollution from manufacturing tech

THE RATINGS CULTURE
✓ Easy to find, give and share opinions
✗ People can become obsessed with how they are perceived

PRIVACY ISSUES
- Do you own your own data?
- Should governments and law enforcement have complete access?
- What are COOKIES? Should you allow websites to access them?

Legislation relevant to Computer Science:

- ❑ The Data Protection Act 2018
- ❑ Computer Misuse Act 1990
- ❑ Copyright Designs and Patents Act 1988
- ❑ Software licences (i.e. open source and proprietary)

NEWS
✓ Lots of up-to-date news sources
✗ Which are real, which are "fake news"?

AI/ROBOTICS
✓ Can do repetitive tasks well
✗ Will it replace humans?

THE 8 PRINCIPLES OF THE DATA PROTECTION ACT 2018

- ✓ Fair, lawful & transparent processing
- ✓ Purpose limitation
- ✓ Data minimisation
- ✓ Accuracy
- ✓ Data retention periods
- ✓ Data security
- ✓ Accountability

REVISION NOTE
You must learn the basic principles of each act of law and be able to recognise and identify the circumstances when it would apply

COMPUTER MISUSE ACT 1990

There are three main principles of the Computer Misuse Act. It is an offence to:

1. access computer material without permission,
2. *e.g. looking at someone else's files*
2. access computer material without permission and with intent to commit criminal offences, *e.g. hacking into your bank's computer and increasing the money in your own account*
3. alter computer data without permission, *e.g. writing a virus to destroy someone else's data*

THE COPYRIGHT DESIGNS AND PATENTS ACT 1988

An act of law designed to provide protection for creators of books, software music and video, against illegal copying, piracy and distribution.

COPYRIGHT – material cannot be used/distributed without permission

CREATIVE COMMONS – material can be used without permission (though credit may need to be given)

SOFTWARE LICENSES

	Open Source	Proprietary	Freeware
Is licensed	✓	✓	✓
Can be edited by users	✓	✗	✗
License is free	✓	✗	✓
Users can make and sell their own version	✓	✗	✗

GCSE COMPUTING KNOWLEDGE ORGANISER

2.1 COMPUTATIONAL THINKING ALGORITHMS AND PROGRAMMING

2.1.1 COMPUTATIONAL THINKING

Principles of computational thinking:

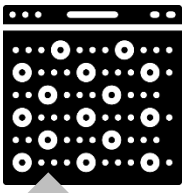
- ❑ Abstraction
- ❑ Decomposition
- ❑ Algorithmic thinking

"The process of approaching problems systematically and creating solutions that can be carried out by a computer" ✓

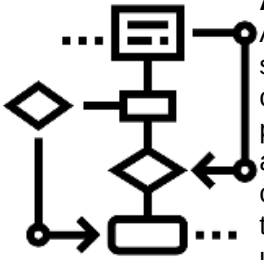
"Thinking like a computer" ✗

REVISION NOTE

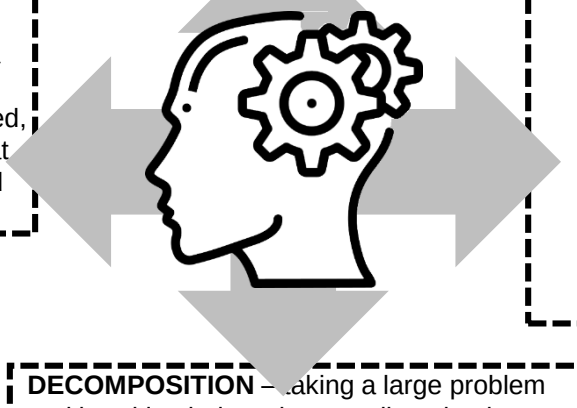
Pattern recognition is one of the four methods of computational thinking – but it is not studied at GCSE level



PATTERN RECOGNITION involves looking for similarities or patterns in different aspects of the problem.



ALGORITHMIC THINKING An algorithm is a series of steps necessary to complete a task or solve a problem. Once an algorithm has been planned, code can be written so that the problem can be solved using a computer.



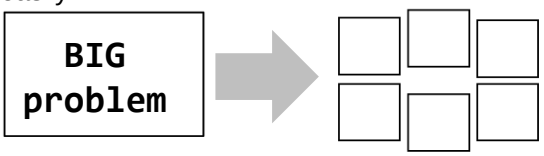
ABSTRACTION – taking only the important and relevant data about the problem and discarding the unnecessary data.

2.1.2 DESIGNING, CREATING AND REFINING ALGORITHMS

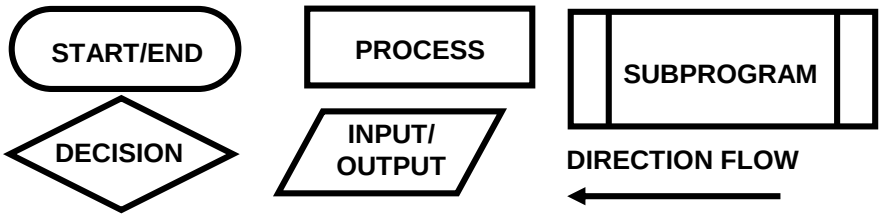
Identify the inputs, processes, and outputs for a problem
Structure diagrams
Create, interpret, correct, complete, and refine algorithms using:

- ❑ Pseudocode
- ❑ Flowcharts
- ❑ Reference language/high-level programming language
- ❑ Identify common errors
- ❑ Trace tables

DECOMPOSITION – taking a large problem and breaking it down into smaller, simpler problems. These are can be tackled more easily



FLOWCHARTS can be used to represent algorithms using the symbols shown here. Algorithms can also be represented using **PSEUDOCODE** or **REFERENCE LANGUAGE**




```
y = 0
z = 0
FOR x in range (4):
        y = x * 2
        z = z + y
```

TRACE TABLES are used to test and identify the outcome of algorithms

x	y	z
0	0	0
1	2	2
2	4	6
3	6	12
4	8	20

High Level Language	Pseudocode	Reference Language
Specific syntax must be used	No formal syntax – can take any form	More formal structure than pseudocode
Used to write code	Used to present an algorithm so that a human can understand it	Used to present an algorithm to closely resemble code
`FOR loop in range(10): PRINT(loop)`	`Loop 10 times Print loop position End loop`	`FOR loop = 1 to 10 PRINT(loop) NEXT loop`

2.1.3 SEARCHING AND SORTING ALGORITHMS

Standard searching algorithms:

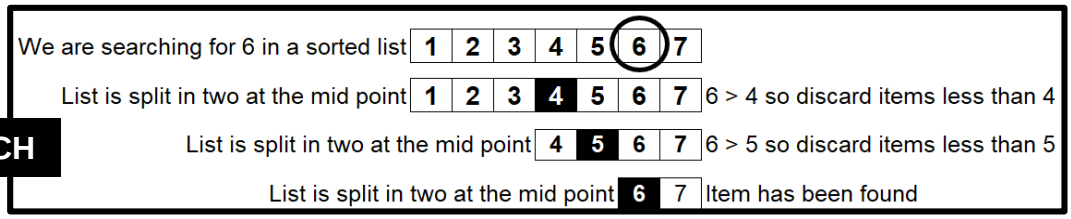
- ❑ Binary search
- ❑ Linear search

A **BINARY SEARCH** requires data to be sorted in order before it can be searched. A **LINEAR SEARCH** does not –the algorithm will look at every item in list until it either locates the data or reaches the end of the list. The binary search is the more efficient of the two

```
INPUT item to be searched for
found = False
numbers = [4,2,6,1,5,3]
REPEAT
   Compare item with current item in list
   IF current item is the item searched for then
            found = True
UNTIL end of list OR found = True
IF found = True
   PRINT ("Item found")
ELSE
   PRINT ("Item not found")
```

LINEAR SEARCH

BINARY SEARCH



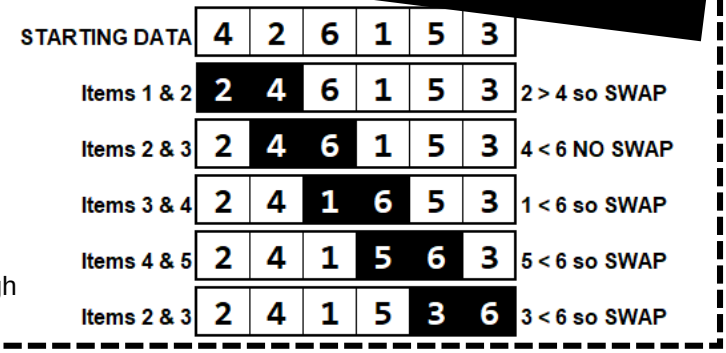
Standard sorting algorithms:

- ❑ Bubble sort
- ❑ Merge sort
- ❑ Insertion sort

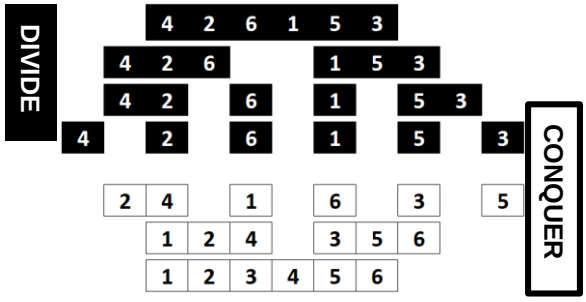
REVISION NOTE

You need to be familiar with searching sorting algorithms but there is no need for you to be able to code them

-A **BUBBLE SORT** is an algorithm for sorting data.
-The algorithm works by going through a list of unordered data and evaluating the data in pairs.
-If two data items are in the wrong order they are exchanged.
-The algorithm then moves to the next pair.
-When the algorithm reaches the end of the data, the process will be repeated until all data has been sorted correctly. This might take **SEVERAL PASSES** through the data.



-A **MERGE SORT** is a **DIVIDE AND CONQUER** algorithm;
-First of all, the items of data in a list are divided in half until each item is in a **SUBLIST** of one item.(This is the **DIVIDE** stage)
-The algorithm will then merge each sublist, after comparing and sorting them as appropriate.
-When all of the data has been merged back into a single list it will be in the correct order. (This is the **CONQUER** stage)
- Merge sorts are more efficient than bubble or insertion sorts.



-An **INSERTION SORT** is more efficient than a bubble sort.
-The insertion sort works in a similar way to sorting a hand of cards.
-The algorithm works by comparing the current data item with the other items in the list
- If the data item is in the wrong place, it is shifted to left until it is in the correct place.
- This continues until all the items of data are in the correct place.

GCSE COMPUTING
KNOWLEDGE ORGANISER

2.2 PROGRAMMING FUNDAMENTALS

2.2.1 PROGRAMMING FUNDAMENTALS

The use of variables, constants, operators, inputs, outputs and assignments
The use of the three basic programming constructs used to control the flow of a program:

- ❑ Sequence
- ❑ Selection
- ❑ Iteration (count- and condition-controlled loops)

The common arithmetic operators
The common Boolean operators AND, OR and NOT

A **VARIABLE** is a memory location used to store data. Programmers can label a variable using an **IDENTIFIER**. The contents of the memory location (and the value of the variable) can be changed by the programmer.

Giving a value to a variable is called **ASSIGNING**. Variables be can be assigned a value directly by the programmer or **INPUT** by the user when running the program.

A print statement can be used to **OUTPUT** data – a print statement can be used to display specific text or the contents of a variable.

```
name = input ("Please enter your name")
print ("Hello,"name)
```

CONSTANTS are similar in principle to variables, but their value does not change throughout the program

```
const Pi = 3.142
```

There are three main "constructs" used in high level language programming – **SEQUENCING**, **SELECTION** and **ITERATION. SEQUENCING** involves a block of code that executes line after line (in sequence) :

```
print("Good morning")
name = input("What is your name?")
print("Hello",name)
age = int(input("How old are you?"))
print(age,"is a very good age!")
```

SELECTION involves the use of **IF** statements to evaluate the contents of a variable - program will execute different code depending on the value of the variable

```
question = input("Do you enjoy
programming?"))
if answer == "yes":
    print("Awesome!")
```

ITERATION is used to repeat (loop) a block of code. This is a more efficient way of programming then to add the same code multiple times. There are two types of iteration; a **count controlled loop** runs a block of code a SET number of times;

```
for count in range (1,10):
    print("I have counted to", count)
```

a **condition controlled loop** runs a block of code until a specific condition is met – for example, a program could ask for a password until it is entered correctly.

```
correct = False
while correct == False
      password = input("Enter your password")
      if password == correctpassword:
            correct = True
```

MATHEMATICAL OPERATORS allow calculations to be performed using variables and constants

+	Addition
-	Subtraction
/	Division
*	Multiplication
DIV	Integer division
MOD	Modulus (remainder)
^	Exponent

BOOLEAN OPERATORS are used when making logical comparisons (i.e. when using IF statements)

NOT	Addition
AND	Subtraction
OR	Division
!=	Not equal to
==	Equal
<	Less than
>	Greater than
<=	Less or = to
>=	Greater than or = to

2.2.2 DATA TYPES

The use of data types:

- ❑ Integer
- ❑ Real
- ❑ Boolean
- ❑ Character and string
- ❑ Casting

Constants and variables can be stored as a range of **DATA TYPES**. It is also possible to use **CASTING** to convert data from one type to another:

```
NumberString = "42"
Number = int(NumberString)
pi = 3.141
pi = int(pi)
print(pi)
3 >_
```

DATA TYPE	EXPLANATION	EXAMPLE
Integer	Whole number	HIGH_SCORE = 100000 RANK = 10
Float/ Real	A "fractional" number	PI = 3.141 TEMPERATURE = 21.5
Character	A single character (letter, number, symbol)	INITIAL = "J" GRADE = "A"
String	Zero or more characters	NAME = "Arthur Dent" PASSWORD = "FISH42*"
Boolean	Can be either TRUE or FALSE	PERMISSION = True CORRECT = False

2.2.3 ADDITIONAL PROGRAMMING TECHNIQUES

The use of basic string manipulation
The use of basic file handling operations:
- ❑ Open
- ❑ Read
- ❑ Write
- ❑ Close

The use of records to store data
The use of SQL to search for data
The use of arrays (or equivalent) when solving problems, including both one-dimensional and two-dimensional arrays
How to use sub programs (functions and procedures) to produce structured code
Random number generation

Data can be imported to/exported from programs using **FILES**. This means that a program can keep its data, even when it is closed and reopened. A range of **FILE HANDLING OPERATIONS** are possible..

`open`	Prepares the file ready for use
`close`	Close access to the file when it is no longer needed
`read`	Retrieve data from the file
`write`	Overwrite the file with new data
`append`	Save new data onto the end of the file

STRING MANIPULATION
Many programming languages (including Python) have built-in functions allow programmers to manipulate strings.

Description	Example	Result
Length	`length = len(name)`	`17`
Convert to upper case	`capitals = name.upper()`	`ZAPHOD BEEBLEBROX`
Convert to lower case	`small = name.lower()`	`zaphod beeblebrox`
Return a substring	`name.substring(0,2)`	`Zap`

There are a wide number of ways in which strings can be manipulated – a few are examples are given in the table for this example: **`name = "Zaphod Beeblebrox"`**

DATABASES are used to organise and structure data. In a database, data is stored in on a table – each row holds a **RECORD** and each column (**FIELD**) refers to different aspect of the data. **SQL** (**STRUCTURED QUERY LANGUAGE**) is a language used to build, edit and interrogate databases.

```
SELECT Pupil_ID, FirstName,Surname
FROM Pupil
WHERE grade > 7
```

REVISION NOTE
Python does not use ARRAYS – make sure you are clear about how they differ from LISTS

PUPIL

Pupil_ID	First Name	Surname	Mentor	Mark	Grade
1012	Ford	Prefect	HG5	80	8
0981	Tricia	McMillan	HG7	95	9
1422	Arthur	Dent	HG1	55	6

```
1012 Ford Prefect
0981 Tricia McMillan
```

	LIST	ARRAY	2 Dimensional Array
Data Structure	✓	✓	✓
Can contain mixed data types	✓	✗	✗
Size can be changed after it has been defined	✓	✗	✗
Arranges data in row and columns	✗	✗	✓

While variables store individual pieces of data **ARRAYS** are data structures which store related items of data.

```
PupilName = {"Ford",
"Tricia", "Arthur"}
NameAndMark = {"Ford", 80,
"Tricia", 95, "Arthur", 6}
ClassTests [20,10]
```

SUBPROGRAMS are "programs within programs" and perform a specific function within a larger program. Using subprograms allows larger programs to be broken down into smaller parts making them easier to design, test and understand.

Description	PROCEDURE	FUNCTION
Example of a subprogram	✓	✓
Needs to be called from the main program	✓	✓
Can have parameters passed into it	✓	✓
Can return values back out to the main program	✗	✓

Programming languages have built-in functions that can be used to generate "RANDOM" numbers.

```
dice_roll = random (1,6)
```

GCSE COMPUTING
KNOWLEDGE ORGANISER

2.3 PRODUCING ROBUST PROGRAMS

2.3.1 DEFENSIVE DESIGN

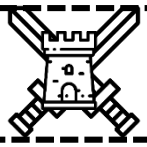
Defensive design considerations:

- ❑ Anticipating misuse
- ❑ Authentication

Input validation

Maintainability:

- ❑ Use of sub programs
- ❑ Naming conventions
- ❑ Indentation
- ❑ Commenting



DEFENSIVE DESIGN means writing a program anticipating that users might either accidentally or deliberately cause it to fail.

AUTHENTICATION is used to ensure that only authorised users are able to access data in a system. This can involve methods such as **captcha** or biometric technology like finger print or face recognition.

VALIDATION is a set of rules that a program can use to ensure user input is restricted to acceptable values and does not crash the program.

MAINTAINABILITY is a way of producing code in such a way that it is easier to fix bugs and flaws because it is easier for others to read and understand. A range of techniques is available to the programmer in order to ensure maintainability.

SUB PROGRAMS make programs easier to understand. They also make errors easier to correct, since they can be isolated without affecting the main program

```
MAIN
   MENU()
   PLAYER_TURN()
   COMPUTER_TURN()
GAME OVER
```

Range check	Checks that a number/date is within a given range
Type check	Checks that the data is of an appropriate type (i.e. text, integer)
Length check	Check that the length of text is neither too short or long
Presence check	Checks that data has been entered
Format check	Checks that the structure of the data is correct – i.e. that a date is written in the correct way

```
for l in range(1,10):
      v =int(input("Enter your guess"))
      if x = n:
            print("Well done!")
            break
      else:
            print("Bad luck")
            print("Out of goes!")
```

```
for loop in range(1,10):
      guess=int(input("Enter your guess"))
      if guess = number:
            print("Well done!")
            break
      else:
            print("Bad luck")
print("Out of goes!")
```

NAMING CONVENTIONS are important because sensibly named constants and variables make a program easier to understand to programmers who are not familiar with it.

```
for loop in range(1,10):
guess=int(input("Enter your guess"))
if guess = number:
print("Well done!")
break
else:
print("Bad luck")
print("Out of goes!")
```

```
for loop in range(1,10):
      guess=int(input("Enter your guess"))
      if guess = number:
            print("Well done!")
            break
      else:
            print("Bad luck")
print("Out of goes!")
```

INDENTATION helps to identify blocks of code where iteration or selection are taking place. Many programming languages automatically indent code. Without indentation it can be hard to identify where there are blocks of code that do not belong to main 'sequence' of the program.

COMMENTING is used to annotate a program listing in order to explain what the code means. This is useful when more than one developer is working on a program and may not be familiar with aspects of the code.

```
#create loop to run until correct password is
entered
#program checks password on file against user
input
```

2.3.2 TESTING

The purpose of testing
Types of testing:
- ❑ Iterative
- ❑ Final/terminal

Identify syntax and logic errors
Selecting and using suitable test data:
- ❑ Normal
- ❑ Boundary
- ❑ Invalid
- ❑ Erroneous

Refining algorithms

THE PURPOSE OF TESTING is not only to ensure that the program works , but to ensure that it completes the tasks that it was designed to do. Testing identifies any bugs that are in the program.

ITERATIVE	TERMINAL
Carried out while the program is in development	Carried out at the end of the development process
Uses **TRACE TABLES** and a **TEST PLAN**	Checks against original plan to make sure all parts work and that it works as intended

- SYNTAX ERRORS are mistakes that prevent the program from running.
- All programming languages have rules - syntax - (for example, about the use of capital letters) and syntax errors occur when these rules are broken.

Capital 'P'

Missing "

```
Print("Good morning")
Name = input("What is your name?)
print("Nice to meet you" name)
```

Missing ,

REVISION NOTE
You need to be able to identify errors in given code and explain how to correct them

-Programs containing **LOGIC ERRORS** will run. However, they do not produce the output that the programmer intended
- This is because the program does not contain syntax errors, but instead has mistakes in the logic that make it behave unexpectedly.
- Logic errors are often harder to spot (and fix) then syntax errors

```
#program to calculate area of a square

side = int(input("Please enter the length)
area = side + 4
print("The area is",area)
```

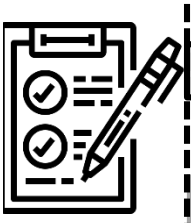
In this example, the program will run but it will not calculate the area of a square because the programmer has used this line…

```
area = side + 4
```

instead of this line…

```
area = side * 4
```

Before a program can be tested, the programmer needs to create a **TEST PLAN**. The programmer will have to identify **TEST DATA** that can be used in the program to see if it generates errors.



```
number = int(input("Please enter a number from 1 – 10))
```

Type of test data	EXPLANATION	EXAMPLE
NORMAL	Data that the program that is designed to handle	1,2,3,4,5,6,7,8,9,10
BOUNDARY /EXTREME	Data on the very edges of what is / isn't acceptable	1,10
INVALID	Data the is outside the limits set by the program	0,11
ERRONEOUS	Data that is unsuitable for the purpose -	A,B,C,#,!

Test Number	Test purpose	Test data	Expected result	Actual result

Typical plan layout

Testing allows the programmer to **REFINE ALGORITHMS.** Once the programmer has tested the program and identified the mistakes, they will need to return to the program to correct or improve the code.

2.4 BOOLEAN LOGIC

2.4.1 BOOLEAN LOGIC

Simple logic diagrams using the operators "AND", "OR" AND "NOT"
Truth tables
Combining Boolean operators using "AND", "OR" and "NOT"
Applying logical operators in truth tables to solve problems

Computers are made up of circuits containing millions of switches. As electrical switches have two possible values (**ON** or **OFF**) , these values can be represented using binary values **1** or **0**.

Each circuit contains logic gates and **BOOLEAN LOGIC** is used to evaluate the results of different combinations of 1's and 0's.

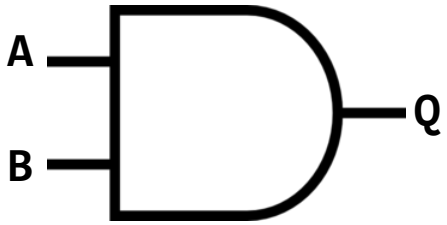
There are a number of different logic gates which produce different results when they receive inputs (1's and 0's.)

The possible values for each gate can be represented using a **TRUTH TABLE.**

An **AND** gate has two possible inputs - **'A'** and '**B'**

'Q' are the possible outputs

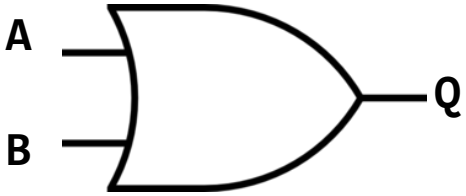
AND gate



A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

An **OR** gate has two possible inputs – **'A'** and **'B'**

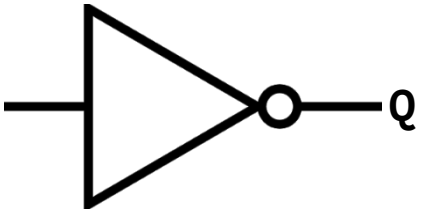
OR gate



A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

A **NOT** gate has a single input – **'A'**

NOT gate

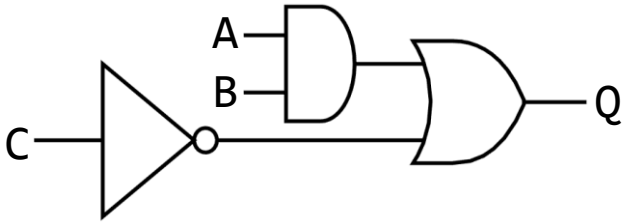


A	Q
0	1
1	0

REVISION NOTE

There are a number of additional gates used in Boolean logic, but only knowledge of AND, OR and NOT is required at GCSE level.

Logic gates can be combined to create complete circuits. These can also be represented using truth tables. The circuit below is made up of three gates:



This an also be represented as a Boolean expression:

(A AND B) OR (NOT C)

A	B	C	Q
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

REVISION NOTE

You need to be able draw a truth table for a given circuit. You also need to be able to represent a circuit as a Boolean expression

GCSE COMPUTING KNOWLEDGE ORGANISER

2.5 PROGRAMMING LANGUAGES AND INTEGRATED DEVELOPMENT ENVIRONMENTS

2.5.1 LANGUAGES

Characteristics and purpose of different levels of programming language:

- ❑ High-level languages
- ❑ Low-level languages

The purpose of translators

The characteristics of a compiler and an interpreter

HIGH LEVEL LANGUAGES have different purposes - for example, games are often written in **JAVA** while **PYTHON** is used for scripting, **LOW LEVEL LANGUAGES** are used for writing device drivers and programs that interact with the hardware.

REVISION NOTE

You are not expected to be able to program in a low level language, but it is important that you are aware of the differences between low and high level languages and how they are used

	Language	Syntax	Translation	Hardware dependent?	Example

LOW LEVEL	Machine Code	Data and instructions made up of 1's and 0's	Does need to be translated	YES (unique to each processor type)	`11000101 11100101`
					`11001101 11010101`
					`01010111 11001000`

	Assembly Language	Mnemonics/ symbols	One statement translates to one machine code instruction	YES (unique to each processor type)	`MOV1 #5B #6A` `LDA1 #6A`
HIGH LEVEL	Python, JAVA, C++, Visual Basic	Resembles human language	One statement translates into many machine code instructions	NO – transferrable and usable on any computer	`print("Hello, world")`

All programs are executed in machine code – this means that any program now written in machine code needs to be translated into this form. Software called **TRANSLATORS** is used to convert High Level Languages or Assembly Language into machine code. There are two types of translator – **COMPILERS** and **INTERPRETERS.** **SOURCE CODE** is the language that the program was written in. When this is compiled into **OBJECT CODE** it creates an **EXECUTABLE** file that can run on any computer without the use of a compiler.

	COMPILER	INTERPRETER
How does translation take place?	Compiles High Level Language programs into machine code when the program is complete	Translates the program as it is being written – translation will only take place on correct code
Produces object code?	✓	✗

REVISION NOTE

Assemblers are another form of translator which do not need to be covered at GCSE

2.5.2 THE INTEGRATED DEVELOPMENT ENVIRONMENT

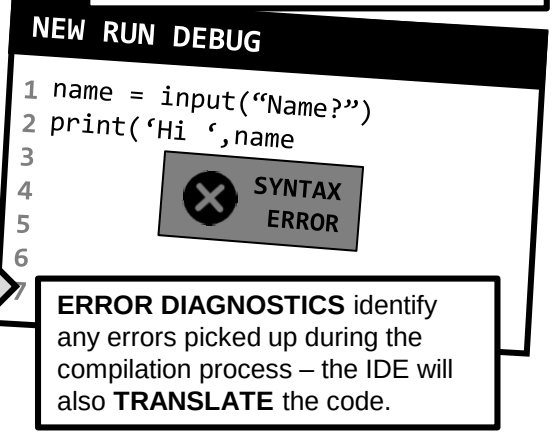
Common tools and facilities available in an Integrated Development Environment (IDE):

- ❑ Editors
- ❑ Error diagnostics
- ❑ Run-time environment
- ❑ Translators

IDE's (INTEGRATED DEVELOPMENT EVIRONMENTS) allow programmers to **WRITE**, **EDIT**, **EXECUTE** and **TRANSLATE** their code

The **RUN –TIME ENVIRONMENT** shows what happens when the code is executed

AN EXAMPLE IDE



The **EDITOR** allows the programmer to enter/edit code and may provide tools like auto-indenting, colour coding variables and commands, and adding line numbers.

ERROR DIAGNOSTICS identify any errors picked up during the compilation process – the IDE will also **TRANSLATE** the code.