PENSBY HIGH SCHOOL

OCR (J277) GCSE

# COMPUTING

## Recap

```
for i in knowledge organiser :
     long_term_memory.append[i]
```

# The specification overview

| Content Overview | Assessment Overview |
|---|---|
| **J277/01: Computer systems**<br>This component will assess:<br>• 1.1 Systems architecture<br>• 1.2 Memory and storage<br>• 1.3 Computer networks, connections and protocols<br>• 1.4 Network security<br>• 1.5 Systems software<br>• 1.6 Ethical, legal, cultural and environmental impacts of digital technology | **Written paper: 1 hour and 30 minutes**<br>**50% of total GCSE**<br>**80 marks**<br>This is a non-calculator paper.<br>All questions are mandatory.<br>This paper consists of multiple choice questions, short response questions and extended response questions. |
| **J277/02: Computational thinking, algorithms and programming**<br>This component will assess:<br>• 2.1 Algorithms<br>• 2.2 Programming fundamentals<br>• 2.3 Producing robust programs<br>• 2.4 Boolean logic<br>• 2.5 Programming languages and Integrated Development Environments | **Written paper: 1 hour and 30 minutes**<br>**50% of total GCSE**<br>**80 marks**<br>This is a non-calculator paper.<br>This paper has two sections: Section A and Section B. Students must answer both sections.<br>All questions are mandatory.<br>In Section B, questions assessing students' ability to write or refine algorithms must be answered using **either** the OCR Exam Reference Language **or** the high-level programming language they are familiar with. |

Practical Programming

All students must be given the opportunity to undertake a programming task(s), either to a specification or
to solve a problem (or problems), during their course of study. Students may draw on some of the content
in both components when engaged in Practical Programming. Please see Sections 2d and 4d for further information.

Full Specification

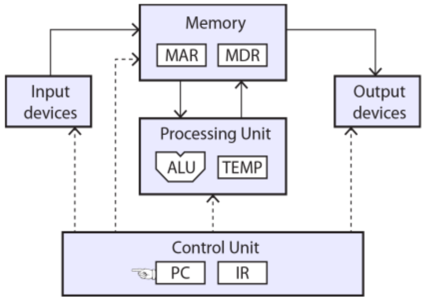
OCR specification-gcse-c

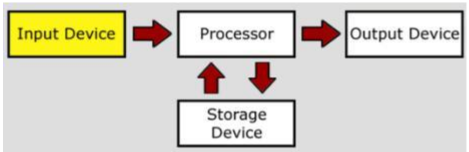
**DQ: Be able to describe the purpose of the CPU. Be able to state the function of the CPU (fetch and execute instructions stored in memory) and understand the fetch decode execute cycle. Be able to describe the common CPU components and their function: ALU (Arithmetic Logic Unit), CU (Control Unit), Cache, Registers. Be able to explain computer systems and explain I/O devices, the system buses (control, address, data) bus, explain the types of peripherals. Explain MDR and MAR.**

**The Von Neumann Architecture** – developed by Jon Von Neumann in 1945 The von Neumann architecture is a design model for a stored-program digital computer that uses a processing unit and a single separate storage structure to hold both instructions and data. A stored-program digital computer is one that keeps its programmed instructions, as well as its data, in read-write, random access memory (RAM)

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

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



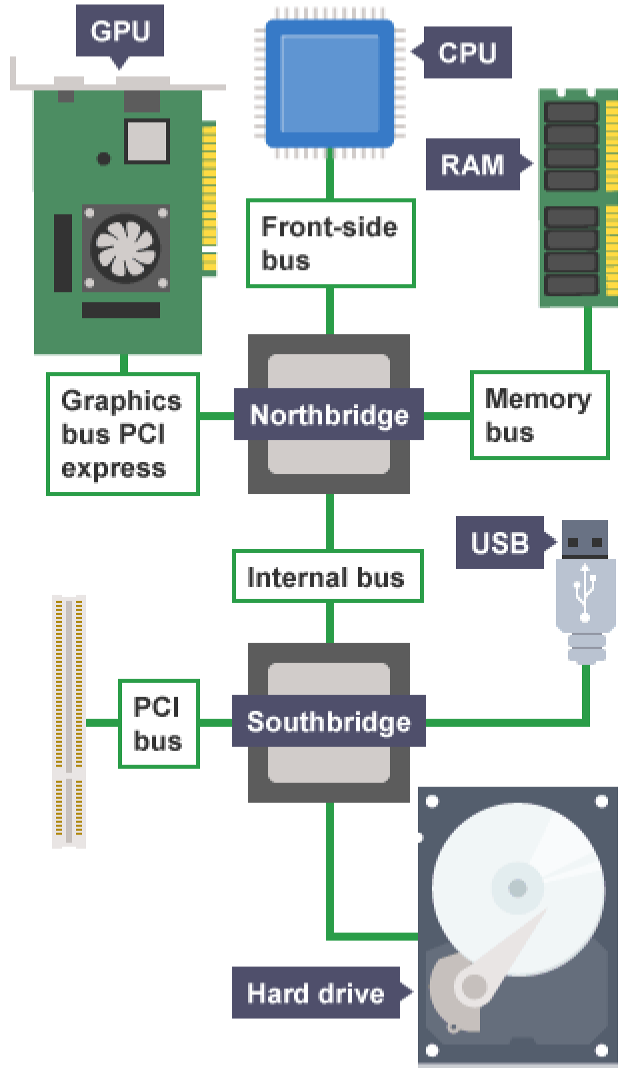

| Topic | Tasks |
|---|---|
| 1. Intro to computer systems. | Introduction to systems architecture. What is a basic computer system? It consists of Input, CPU (Processing), Output and Storage. Label and identify the basic computer system, identifying components.   |
| 2. CPU | The purpose of the CPU – termed the 'brains of the computer'. What is the CPU and what are the characteristics? |
| 3. Architecture | What is the Von Neumann architecture – it is a stored program concept and its design is still based on how general computes are run today. Research the architecture and its components and complete your poster with the components on the Motherboard. |
| 4. Components of the processor | Identify the components ALU, CU and how the program instructions and data move between the main memory and processor using 'Buses'. Include the buses on your Motherboard poster (above). |
| 5. FDE | When a program is to be run, it has to be loaded into memory first. This is called the Fetch – Decode – Execute cycle. Create the FDE in your books and identify all the stages in the cycle and explain what each one does. |
| 6. CPU Performance | Factors affecting CPU performance include clock speed, type and size of memory and computer cache. Explain how clock speed and cache can affect the performance of a computer. |
| 7. Embedded systems | Devices in our homes use CPUs to control functions. These are embedded systems and they have a dedicated function. Find examples of embedded systems and explain their function and why it's embedded. Such as a: dishwasher, GPS, digital watch. |
| 9.Overview/recap/misconceptions | Create a revision poster for the topics.<br>Complete the Seneca Learning topic on Systems architecture. |
| 10. Revision/test | End of unit test |

## What you need to know: (J277)

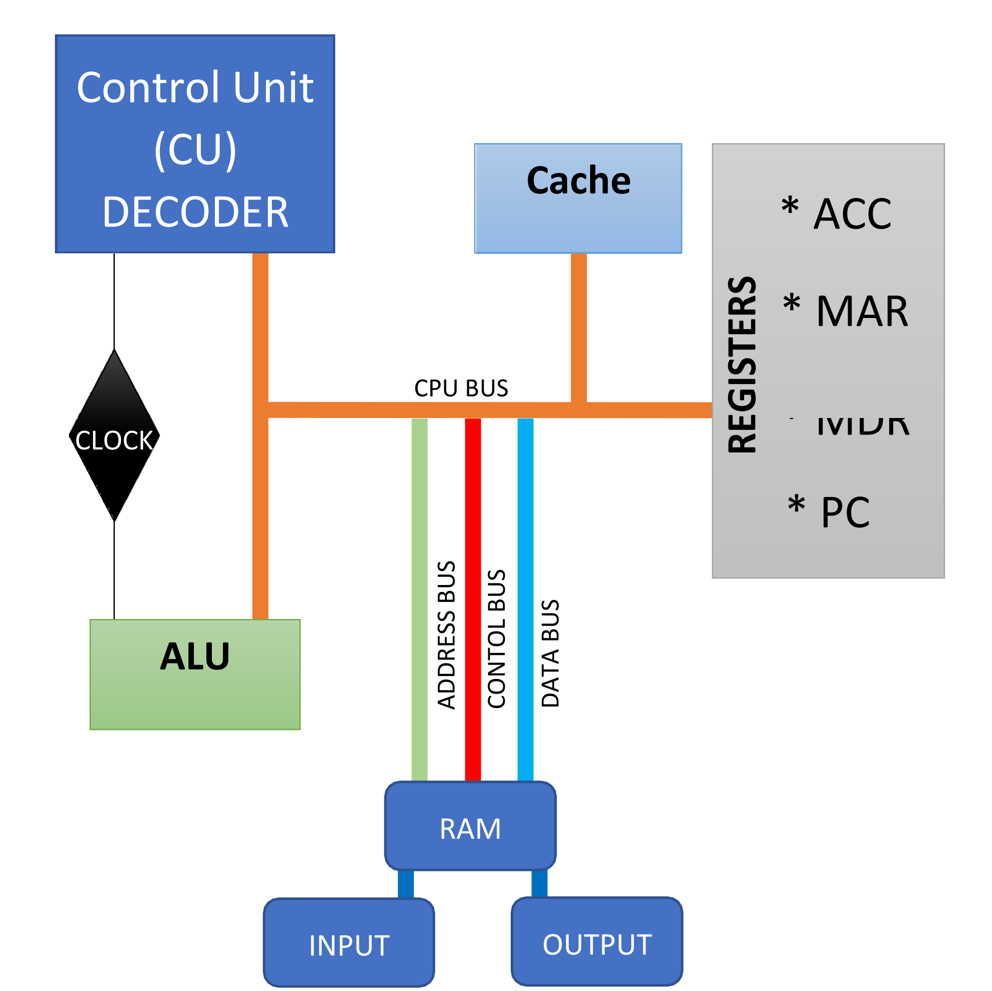
- The purpose of the CPU (fetch, decode, execute cycle)
- Common CPU components and their functions (ALU, CU, Cache, Registers)
- Von Neumann architecture (MAR, MDR, PC (Program counter), ACC (Accumulator)
- CPU Performance – clock speed, cache size, number of cores
- Embedded system – the purpose and characteristics of embedded systems
- Examples of embedded systems

| KEY VOCABULARY | |
|---|---|
| CPU | *Central Processing Unit*. - The "brain" of the computer |
| CU | *Control Unit.* - Part of the CPU that manages the functions of all other parts of the CPU |
| Decoder | Part of the CU which decodes the binary instructions fetched from memory |
| RAM | *Random Access Memory* - The main volatile memory into which programs are loaded from the hard drive |
| MAR | *Memory Address Register* - Small fast memory used to store the RAM address of the next instruction |
| MDR | *Memory Data Register* - Small, fast memory used to store the information collected from the RAM before processing |
| PC | *Program Counter* - Keeps track of the current instruction number of the program |
| Accumulator | Small, fast memory, used to keep track of the data currently being processed |
| ALU | *Arithmetic and Logic Unit* - Does the basic mathematics and comparisons during processing |
| Bus | A physical connection between two elements of a computer system that allows the transfer of data. |
| Cache | Incredibly fast, but very expensive volatile memory using in the CPU |
| Bridge (North / South) | Junctions on a motherboard where the bus connections are controlled and routed. Northbridge deals with core functions, whilst the Southbridge deals with the peripherals, input and output devices and Secondary Storage. |
| von Neumann Architecture | The method used by all modern computers to allow the programming of a machine to be changed depending on the required function. |
| Fetch / Decode / Execute Cycle | Basis of the von Neumann architecture – the repeated process where instructions are fetched from RAM, decoded into tasks and data, then carried out. |
| Clock Speed | The number of FDE cycles that a CPU can carry out per second. Measured in Ghz (1 Ghz = $10^9$ cycles per second or 1,000,000,000hz) |
| Cores | Some processors have multiple CPUs which can work in parallel, sequentially or can multitask. Dual and Quad cores are common in modern PCs |

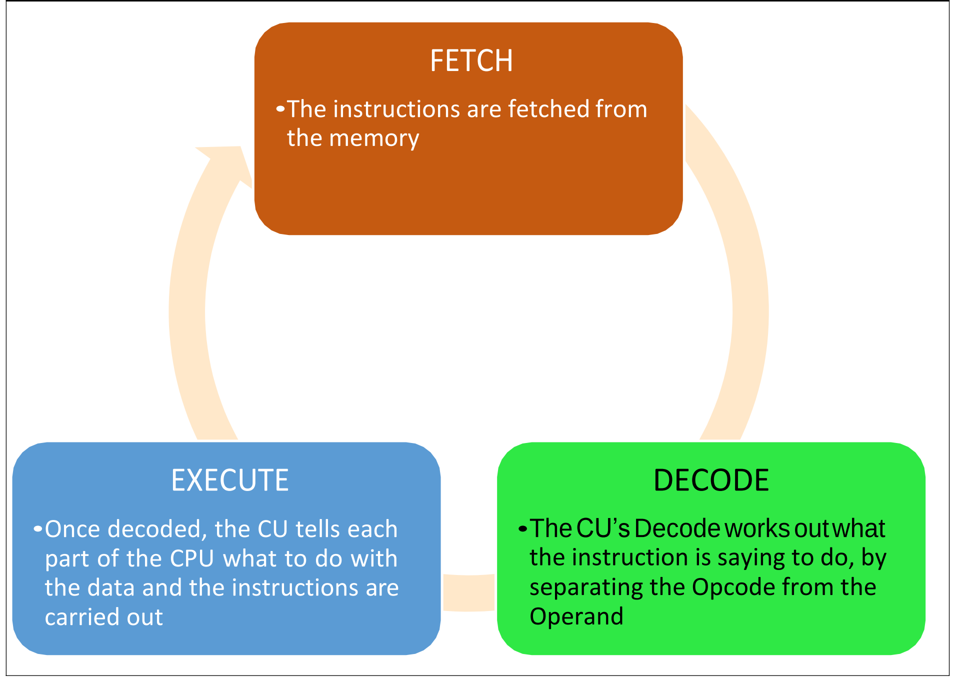
An example of a typical PC's innards.



## BASIC DIAGRAM OF CPU

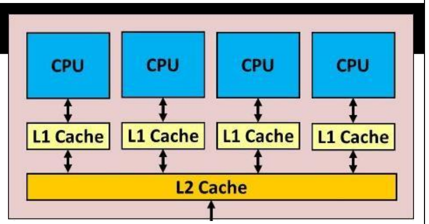



## The FETCH – DECODE – EXECUTE cycle



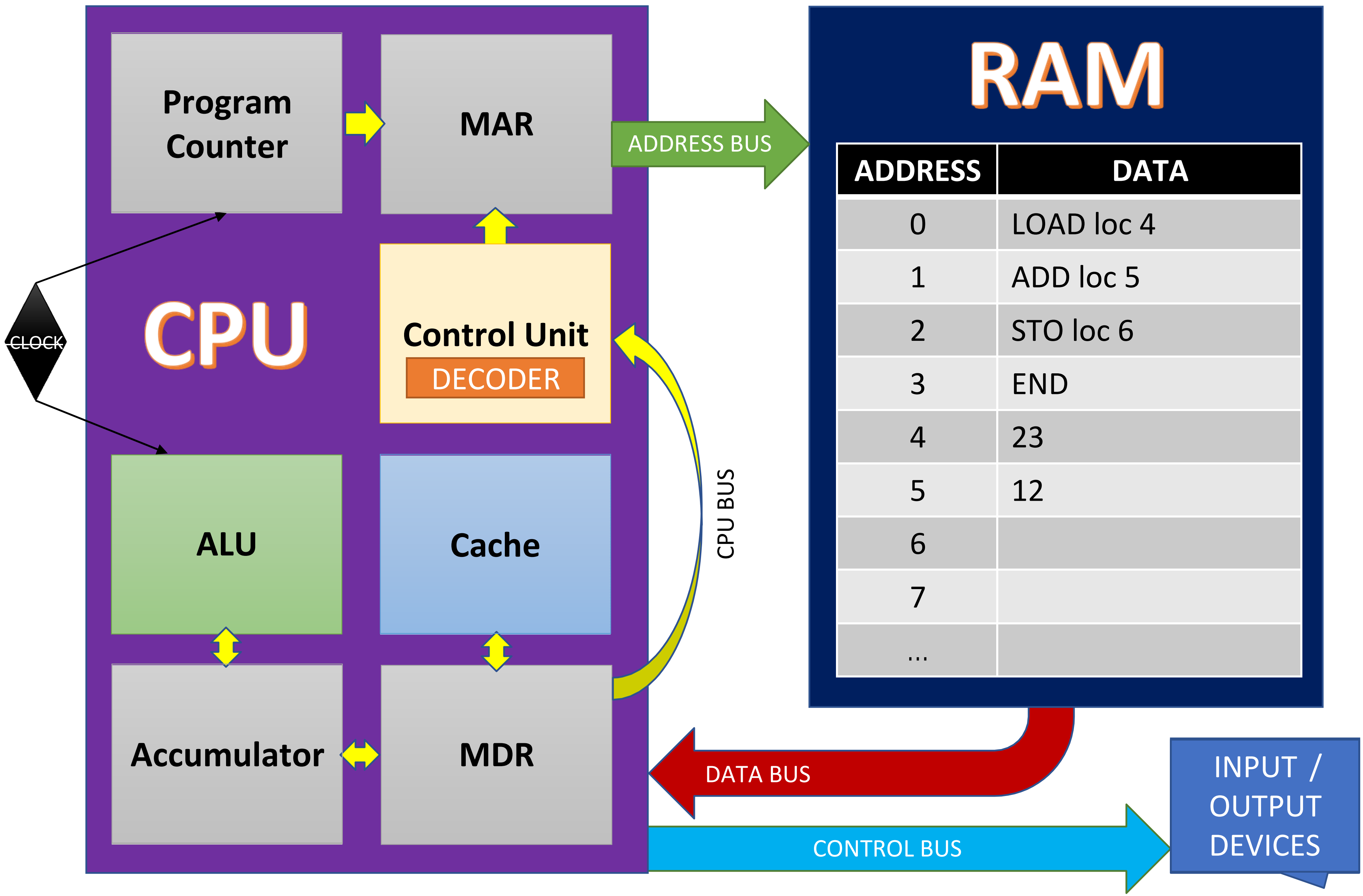

## Multi Core Processing

Some processors have multiple CPU cores on one chip. They all have their own Level 1 cache, but share Level 2 cache, allowing them to collaborate quickly on large tasks.




## KEY VOCABULARY

| | |
|---|---|
| Machine Code | A program, stored in binary, that the CPU undertakes the FDE cycle on. All programs must be in machine code to work |
| Instruction | A single line of machine code, containing the command and data location on which it is to be executed. Stored in binary |
| Opcode | The first part of the instruction, is the command |
| Operand | The second part of the instruction is the data on which to carry out the command. This may be actual data stored in binary form, or a memory location reference of where to find the data |


Program Counter
MAR
CPU
Control Unit
DECODER
CLOCK
ALU
Cache
Accumulator
MDR
CPU BUS
ADDRESS BUS
RAM
ADDRESS
DATA
0
LOAD loc 4
1
ADD loc 5
2
STO loc 6
3
END
4
23
5
12
6
7
...
DATA BUS
CONTROL BUS
INPUT / OUTPUT DEVICES

# Summary Questions

- What does CPU stand for? [1]
- If a CPU has more than one ‘core’ on it, it is called what? [1]
- What is FDE short for? [1]
- What is the name of the type of architecture used to make CPU’s? [1]
- What are the three ‘tasks’ that any computer system performs? [1]
- Where does the CPU fetch the next bit of program or data from? [1]
- What does ‘decode’ an instruction mean? [1]
- What are the three factors that influence the performance of a CPU? [1]
- Explain what an embedded system is? [2]
- Explain why embedded systems are used in devices [2]
- List 4 devices that would use embedded systems. [4]
- What is cache memory? [1]
- Where is cache memory found? [1]
- Write down two things you can say about cache memory [2]
- List 3 properties of an embedded system. [3]
- List 3 components of a CPU. [3]
- Is a CPU hardware or software? [1]
- What is a CPU’s clock speed measured in? [1]
- Which memory outside of the CPU is the CPU connected to? [1]
- What does ‘overclocking’ mean? [1]

An embedded system is a combination of software and hardware that performs a specific task rather than a general-purpose computer that is designed to carry out multiple tasks.

Embedded systems are included as a part of a complete device often with hardware and mechanical parts. As the systems carry out specific tasks, they can be designed to be small and have a low cost. Mass-production of embedded systems can save large amounts of money.

The software written for an embedded system is known as firmware. The instructions are stored in read-only memory or in Flash memory. The software runs with limited computer hardware resources, little memory and no peripherals.

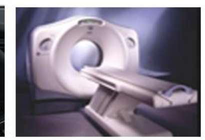
Most embedded systems are reactive - they react to conditions such as temperature, weight, vibration and air quality. These systems detect external conditions and react to them by recording data, turning motors on or off, sounding an alarm or sending a message to another processor.

Reactive embedded systems often control real time events so must be completely reliable. For example, drivers rely of the anti-lock braking system of their car working correctly to avoid accidents on the road.

When an embedded system performs operations at high speed, and if it is very reliable, it can be used for real -time applications. If the size of the embedded system is very small and power consumption very low, then the system can be easily adapted for different situations.

Some examples of embedded systems:

| | |
|---|---|
| Electronics | Mobile phones, games consoles, printers, televisions, digital cameras |
| In the home | Washing machines, microwave ovens, refrigerators, dishwashers, air conditioners |
| Medical equipment | CT Scanners, Electrocardiogram (ECG), MRI Scanners, blood pressure monitors, heartbeat monitors |
| Cars | Electronic fuel injection systems, anti-lock braking systems, air-conditioner controls. |



**INTERESTING FACT**
98% of the microprocessors manufactured go into embedded systems.

| KEY VOCABULARY | |
|---|---|
| Volatile | Memory which requires constant electrical charge. If the power is turned off, then the data is lost |
| Non-volatile | Memory which can retain its data when the power is turned off |
| RAM | *Random Access Memory* |
| ROM | *Read-Only Memory* |
| Cache | Very fast memory, on, or very close to the CPU |
| Virtual Memory | A section of the HDD which can be used as RAM for very memory intensive processes |
| Flash Memory | A type of dynamic (changeable) ROM |
| Boot Process | The instructions needed to start the computer and to initialize the operating system. |
| POST | *Power On Startup Test*<br>A series of checks done on the hardware of the computer to ensure the machine can run. |

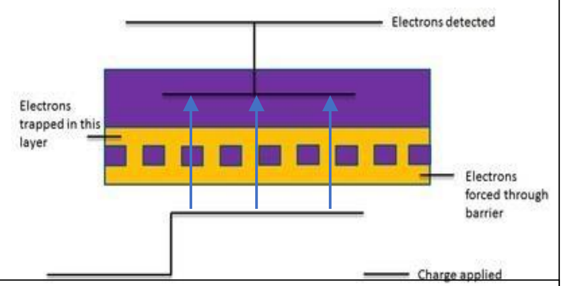
| PRIMARY MEMORY | | | |
|---|---|---|---|
| *TYPE* | *VOLATILE?* | *DYNAMIC?* | *RELATIVE SPEED* |
| Cache | YES | YES | Very Fast |
| RAM | YES | YES | Fast |
| ROM | NO | NO | Slow |
| Flash | NO | YES | Slow |
| Virtual | YES | YES | Very Slow |

## PRIMARY STORAGE - MEMORY

**RAM** is *volatile* memory, which stores data in a single transistor and capacitor. This means it needs a constantly recycled charge to hold its data. If the power is turned off, it cannot refresh the data and it is lost. This is known as *DYNAMIC* memory. The computer uses RAM to store the current program or data being used.

**ROM** is non-volatile. The data is hardcoded onto the chip by the manufacturer, and cannot be overwritten by the user. Because it holds its information even when the power is turned off, this makes ROM ideal for storing the instructions needed to get the computer started up – the *BOOT PROCESS, and POST.*

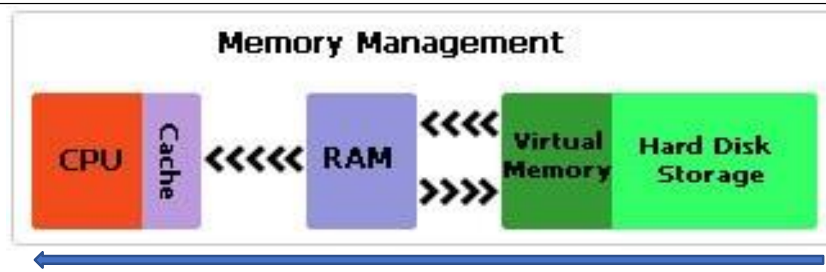
**Flash Memory** is a new(ish) type of ROM chip which holds its data when there is no power making it *non-volatile* but that can be rewritten easily by the user. By using a relatively large electric current, electrons can be *forced* through a barrier and into the *storage layer.* The pattern of electrons can be read as data without affecting the data.




## VIRTUAL MEMORY

To increase the speed and efficiency of RAM, most machines allocate a small portion of the Hard Disk to *VIRTUAL MEMORY.* The contents of the RAM are moved between the slower Virtual Memory and RAM as and when they are needed.

Using / Increasing Virtual Memory does not improve the speed of the computer, but rather using Virtual Memory increases the threshold at which a computer locks, by increasing the usable memory, and preventing deadlock due to filling the available primary memory.




Transfer Speed Increases / Capacity decreases

## *1.2 – Storage*

All basic computing functions are done using Primary Storage – but this is either *volatile RAM* or *static ROM.* To allow storage of a user's information once the power is turned off, *non-volatile, secondary storage* is required.

| KEY VOCABULARY | |
|---|---|
| Secondary Storage | Primary storage is RAM. Secondary storage refers to long term, non-volatile data storage. |
| Non-volatile | Memory which can retain its data when the power is turned off |
| Magnetic | Data is stored by altering the magnetic charge (+ or -) to represent binary information |
| Optical | A reflective layer or dye is marked to either reflect or not reflect a laser beam. The computer reads the reflections as binary data |
| Solid State | Also known as *Flash Memory,* the data is stored by forcing (or flashing) electrons through a barrier into a storage layer. Here it is read as binary information |

| SECONDARY STORAGE | | | | | |
|---|---|---|---|---|---|
| *TYPE* | *CAPACITY* | *COST* | *SPEED* | *Pros* | *Cons* |
| Magnetic | Very High | Low | Fast | Cheap and readily available. Can have very high storage capacity and is reliable | Slow read and write speeds. Moving parts make it susceptible to damage if moved. Data can be wiped if placed near a magnet |
| Optical | Low | Very Low | Slow | Cheap. Can be either Read or Read/Write. | Requires an optical drive to be read. Data corruption occurs over time (10+ yrs) |
| Flash / Solid State | Low | High | Very Fast | Much faster than magnetic drives. No moving parts, so hard to damage by movement. Silent. | Expensive and relatively low capacity. Has limited usable life – about 100,000 rewrites. |

| EXAMPLE FILE SIZES | |
|---|---|
| 1 page text | 100kb |
| 1 photo | 6mb |
| 3 min MP3 | 6mb |
| 3 min audio track (CD) | 50mb |
| DVD film | 4gb |
| HD film | 8-15gb |
| Blu-Ray film | 20-25gb |
| 4k film | 100gb + |

| SECONDARY STORAGE SPECS | | |
|---|---|---|
| *TYPE* | *CAPACITY* | *SPEED* |
| Magnetic HDD | Terabytes | 50-120 MB/s |
| CD | 700 mb | 0.146 MB/s |
| DVD | 4.7 gb | 1.32 MB/s |
| Blu-Ray | 128 gb | 72 MB/s |
| SD Cards | 4-32 gb | 50-120 MB/s |
| USB Drive | Up to 1 tb | 45-90 MB/s |
| Solid State Drive (SSD) | Up to 4 tb but *very* expensive | 200-550 MB/s |

| CONSIDERATIONS WHEN SELECTING SECONDARY STORAGE | |
|---|---|
| Capacity | How much data will it need to hold? |
| Speed | How quickly must the data be written / read? |
| Portability | Does the storage device need to be transported?<br>If yes, then size, shape and weight are important. Will it require other devices to be used (eg. An optical reader). |
| Durability | How *robust* is the device? Can it be moved without fear of damage? Will it be used in a difficult environment? Does it need to be single use or rewritable? |
| Reliability | Does it need to be used over and over again without failing, or will it receive minimal reuse? Will it need to store the information for long periods of time? |
| Cost | Needs to be compared with the above and considered. |

| KEY VOCABULARY | |
|---|---|
| Denary | Base 10 number system. Uses digits 0,1,2,3,4,5,6,7,8,9 |
| Binary | Base 2 number system. Uses digits 0 and 1 only. |
| Hexadecimal (Hex) | Base 16 number system. Uses characters 0-9 and A,B,C,D,E and F |
| BIT | Contraction of BINARY DIGIT – a single value of 0 or 1 |
| Binary Code | Representation of values using multiple bits |
| Character Set | A list of unique values, stored in binary, which represent the letters, numbers and symbols a computer can show/use. |
| ASCII | American Standard Code for Information Interchange.<br>A character set which uses 7 bits to store 128 ($2^7$) characters |
| Extended ASCII | A character set which uses 8 bits to store 256 ($2^8$) characters |
| UNICODE | A characters set which uses 16 bits to store 65,535 characters ($2^{16}$) |
| INTEGER | A whole number (value written to 0 decimal places) |
| FLOAT | A decimal value |
| Conversion | Moving a value from one data type/representation to another, for example Binary to Hex |
| Exponent | Mathematical term which tells you how many time to multiply a BASE by itself. |

**REMEMBER MAXIMUM VALUES!**

Max value which can be represented with 8 bits (1 byte) = **255**

Total number of available values = **256 (255 + 0)**

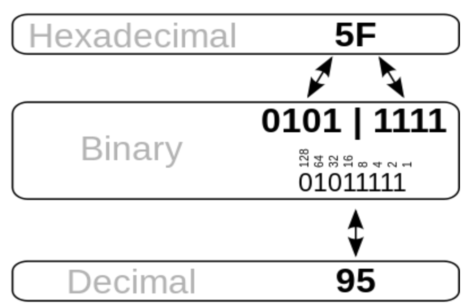
| BINARY PLACE VALUES | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| BASE $^{\text{Exponent}}$ | $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| PLACE VALUE | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

**UNITS OF DATA IN COMPUTER SYSTEMS**

| UNIT | VALUE | SIZE |
|---|---|---|
| bit (b) | 0 or 1 | 1/8 of a byte |
| nibble | 4 bits | ½ a byte (a nibble… get it?!) |
| byte (B) | 8 bits | 1 byte |
| kilobyte (kB) | $1000^1$ bytes | 1,000 bytes |
| megabyte (mB) | $1000^2$ bytes | 1,000,000 bytes |
| gigabyte (gB) | $1000^3$ bytes | 1,000,000,000 bytes |
| terabyte (tB) | $1000^4$ bytes | 1,000,000,000,000 bytes |
| petabyte (pB) | $1000^5$ bytes | 1,000,000,000,000,000 bytes |

**CONVERTING DENARY TO BINARY TO HEX**

| HEXADECIMAL | |
|---|---|
| DENARY | HEX |
| 0-9 | 0-9 |
| 10 | A |
| 11 | *B* |
| 12 | *C* |
| 13 | D |
| 14 | E |
| 15 | F |




There are two methods for converting a HEX value to Denary

OR:

**5F** = (**5**x16) + **F**

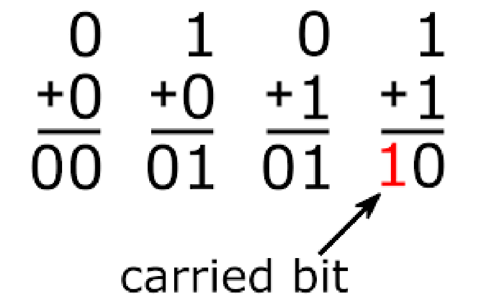
**5F** = 80 + 15

**5F** = 95

## KEY VOCABULARY

| | |
|---|---|
| Overflow Error | Where the denary value cannot be represented with the given number of bits. |
| Binary Shift | The method for multiplying and dividing numbers in binary. Is not necessarily mathematically correct |
| Most Significant Bit | The left-most bit in a binary number – it has the highest value |
| Least Significant Bit | The right-most bit in a binary number – it has the lowest possible value = 0 or 1 |
| Check Digits | Bits used to ensure that the value sent digitally is not corrupted on transfer |
| Lossy Compression | Data is removed from the file to make it smaller. This data is lost and cannot be regained. Suitable where the loss of data is likely not to be noticed. Eg images |
| Lossless Compression | No data is lost, but rather rearranged to ensure a perfect version of the data can be returned. Used where exact reproduction is vital. Eg text documents |
| JPEG / JPG | Joint Photographic Experts Group<br>Compression for images – lossy |
| GIF | Graphics Interchange Format<br>Lossless bitmapped image format for limited colours. |
| PDF | Printable Document Format<br>Open standard for reproducing text and graphic documents without editing permissions – lossless |
| MPEG | Moving Pictures Expert Group<br>Audio-Visual encoding for video Lossy |
| MP3 | Moving Pictures Expert Group Audio Layer 3<br>Digital music files. Lossy compression, but very good and generally only removes sounds that are beyond |

## BINARY ADDITION




When adding 2 large binary numbers, if there is not enough bits to take the *carried bit* then this results in an **OVERFLOW ERROR**

1 1 0 0 1 1 0 1

+ 0 1 0 1 1 1 1 0

1 0 0 1 0 1 0 1 1

This value is not counted, it is *overflow*.

In 8 bits, this sum reads : 203 + 94 = 43!

## BINARY SHIFT

| | |
|---|---|
| Multiplication | Binary shift to the LEFT |
| Division | Binary shift to the RIGHT |

By ***moving the bits*** to either the left of the right, we can double (x2) or halve (%2) the value with each movement.

8 4 2 1

1 0 1 1 =11

A 1 place RIGHT SHIFT (divide by 2)

8 4 2 1

0 1 0 1 1 =5

The bits which are moved outside of the available value places are **LOST.** They cannot be returned by reversing the shift. The same is true at the highest place value

8 4 2 1

1 0 1 1 =11

A single LEFT SHIFT (multiply by 2) would result in an overflow error (when represented with 4 bits.)

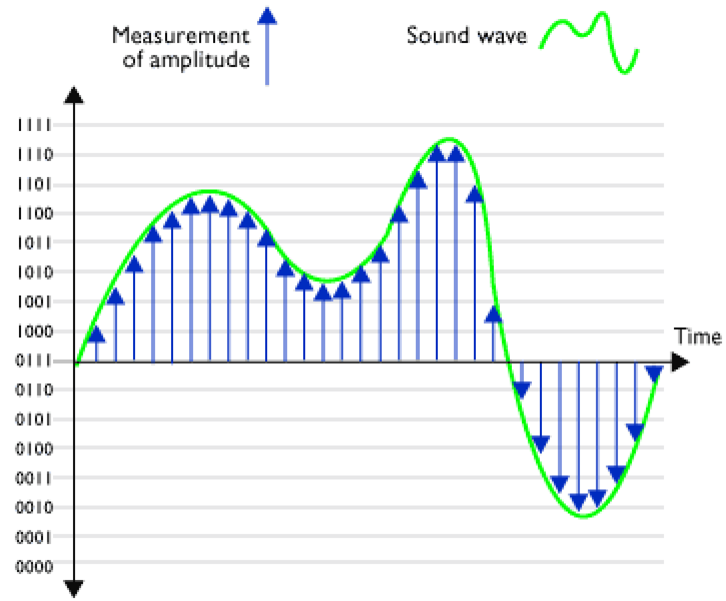
8 4 2 1

1 0 1 1 0 =10

## KEY VOCABULARY

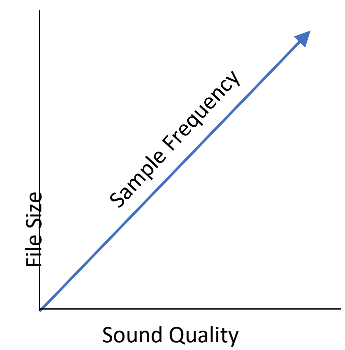
| | |
|---|---|
| Pixel | Smallest element of an image – the dots that make up an image on a screen |
| Bitmap | An image where every pixel is ‘mapped’ in binary to show it’s colour, transparency (Alpha) and brightness (Gamma) Increasing size will lower the quality |
| Vector | An image where the lines are stored as mathematical shapes, so the size can be increased without impacting quality |
| RGB | Red Green Blue – the order of colour data in a pixel |
| Colour Depth (bit depth) | The number of bits used to represent each pixel. Shown in bits per pixel (bpp) |
| Resolution | The number of pixels used per unit eg pixels per inch (ppi) |
| Metadata | Data about the data – in relation to images, it is the data that allows the computer to recreate the image from it’s binary form. |
| Analogue | Continuous changing values – no “smallest interval” |
| Bit Depth | The number of bits used to store the sound |
| Bit Rate | The number of bits used to store 1 second of sound |
| Sample Rate | The number of times the sound is sampled in 1 second; measured in kHz (kilohertz or 1000’s per second) |

## SOUND SAMPLING




Each measurement is assigned a number (byte) according to its amplitude. The end result is a file comprising a string of bytes, eg ...
1001 1110 0001 1010 0111 0100 1111 1101 etc

As the sample rate increases, the quality of the sound goes up – the sound is closer to the analogue original, but the file size also increases. Reduce the sample rate, you reduce quality but also file size.




## BIT DEPTH = NUMBER OF COLOURS

| Bit depth | Available colours |
|---|---|
| 1 bit (Monochrome) | $2^1 = 2$ |
| 2 bits | $2^2 = 4$ |
| 3 bits | $2^3 = 8$ |
| 8 bits | $2^8 = 256$ |
| 16 bits (High Color) | $2^{16} = 65{,}536$ |
| 24 bits (True Color) | $2^{24}$ = 16.7 million |
| 32 bits (Deep Color) | $2^{32}$ = 4.3 billion |

## ESTIMATING FILE SIZES

<u>IMAGES:</u>

**width** X **height** X **colour depth = size**

<u>SOUND:</u>

**Nº of channels** X **sample rate** X **bit depth**

To get the value into mB, you divide by **1,000,000!**

Compression is the process of making a file size smaller. This may be advantageous as it allows more data to be stored on the disk and files may also be transferred more quickly. There are two methods of achieving disk compression; one is software based and the other hardware based.

Software based disk compression is often included as a facility of an operating system and so it is readily available on most computer systems. The disadvantage of this is that it slows down the process of reading and writing to disk.

Hardware disk compression requires specialist hardware, which can be expensive. However, it does not affect the speed of access as much as software based disk compression.

Disk based compression is always lossless.

## Compression types

Compression is the process of making a file size smaller. This may be advantageous as it allows more data to be stored on the disk and files may also be transferred more quickly. There are two primary methods that are used to compress files stored on a computer system; these are *lossy* and *lossless*.

### Lossless compression

Lossless compression uses an algorithm that compresses data into a form that may be decompressed at a later time without any loss of data, returning the file to its exact original form. It is preferred to lossy compression when the loss of any detail, for example in a computer program or a word-processed document, could have a detrimental effect.

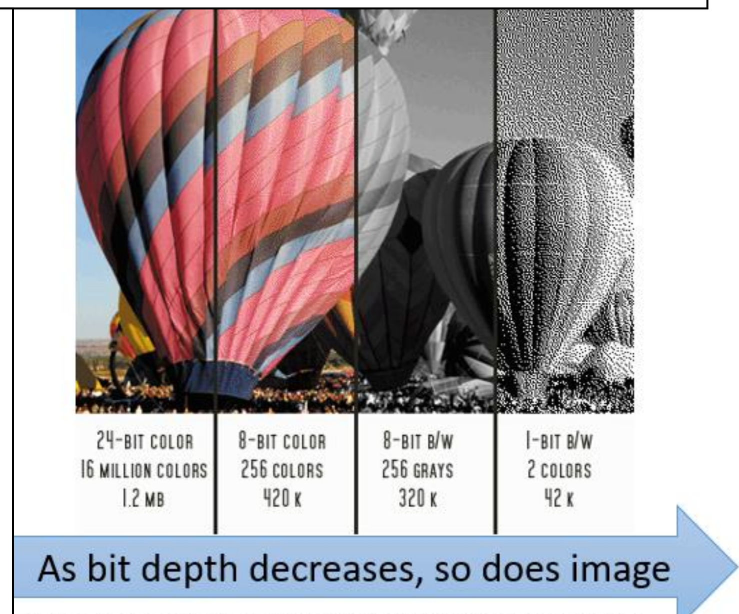
A simplified version of lossless compression on a word-processed document may to be to replace a common string, such as 'the', with a token such as the symbol @. One character takes 1 byte of memory; therefore, the string 'the' would take 3 bytes.

| | | |
|---|---|---|
| **Original uncompressed text** | The word the, is the most frequently used word in the English language. | 71 characters (bytes) |
| **Compressed text** | @ word @, is @ most frequently used word in @ English language. | 63 characters (bytes) |

This is an 11% reduction in the file size!

### Lossy compression

Lossy compression is a technique that compresses the file size by discarding some of the data. The technique aims to reduce the amount of data that needs to be stored.
The following versions of the Air Balloon image show how much of the data can be discarded, and how the quality of the images deteriorate as the data that made up the original is discarded. Typically, a substantial amount of data can be discarded before the result is noticeable to the user. Compression ratio is calculated using the above formula:




Lossy compression is also used to compress multimedia data, such as sound and video, especially in applications that stream media over the Internet.

*1.3 Networks - Wired & Wireless*

| KEY VOCABULARY | |
|---|---|
| Stand Alone | A single machine, not connected to another |
| Network | A collection of machines which can communicate with one another |
| Transparent | The end-user has no need to know the specifics of a network's infrastructure |
| Node | A device on a network (PC or other device) |
| Link | The connections between nodes |
| LAN | Local Area Network (Single location) |
| WAN | Wide Area Network (Multiple connected locations) |
| VPN | Virtual Private Network |
| UTP | Unshielded Twisted Pair – a type of cable |
| Client | The user machines on a network |
| Server | The central 'controller' machine on a network, including main data storage |
| P2P | Peer-2-Peer. A network without a server. |
| WAP | Wireless Access Point |
| NIC | Network Interface Controller |
| Router | Controls the sending of data around a network |
| Hub | A central connection for a small network, which broadcasts all data to all clients |
| Switch | A smart hub for larger networks which only sends the data to the intended client |
| Internet | A worldwide collection of networks |
| WAP | Wireless Access Point |

## WHY NETWORK?

There are many reasons to create networks of computers, and increasingly few reasons not to.

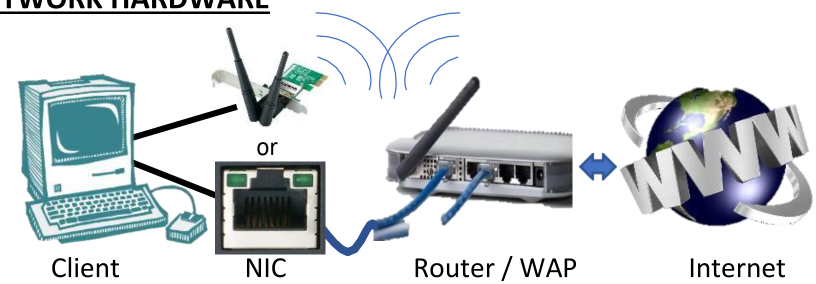
**Positives**

- Communication between users
- Sharing of files
- Sharing of peripheral devices
- Monitoring user activity
- Access control or other security features
- Centralised administration of machines
- Multiple work stations available for users
- Possible to distribute workload for large tasks

**Negatives**

- Higher cost than single machines
- Requires additional hardware
- Requires administration
- Open to attacks
- Client-Server systems are vulnerable to server failure

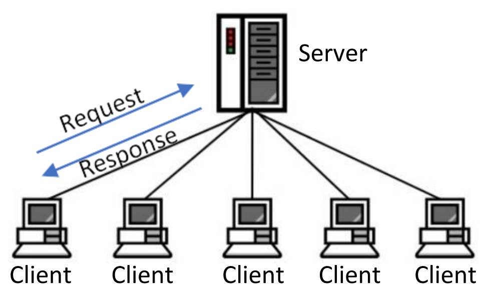
## NETWORK HARDWARE




All clients need an NIC to connect to a ROUTER. This could be a wireless adapter or a network card. The Router in this simple connection can host multiple clients, but more advanced hardware is needed for bigger networks

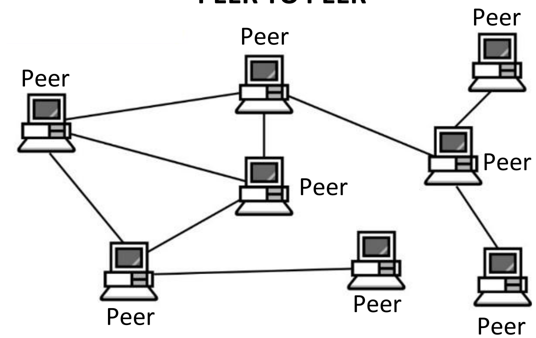
## NETWORK ORGANISATION

**CLIENT - SERVER**




A single high-spec machine is designated the server, which includes the main file storage. Each client then *requests* data from the server which *responds* and fulfills the request.

**PEER TO PEER**




A *distributed* system where each node is equal. Every computer can serve and request data from all others. The system is easy to set up, but slow and difficult to administer.

| KEY VOCABULARY | |
|---|---|
| WAN | Wide Area Network |
| VPN | Virtual Private Network |
| Client | The user machines on a network |
| Server | The central 'controller' machine on a network, including main data storage |
| Internet | A worldwide network of networks |
| DNS | Domain Name Server |
| Hosting | Storing a file on a web-server for access via the internet |
| Cloud | A service which is stored remotely |
| TCP/IP | Transmission Control Protocol / Internet Protocol. These are the standards that allows network nodes to communicate with one another on the internet |
| WWW | World Wide Web - Pages of content |
| email | Electronic mail, sent through the internet |
| URL | Unique Resource Location |

## Virtual Private Networks

VPNs are small collections of devices that act as though physically connected in a LAN, but are actually widely distributed and use a the internet as their network connections.

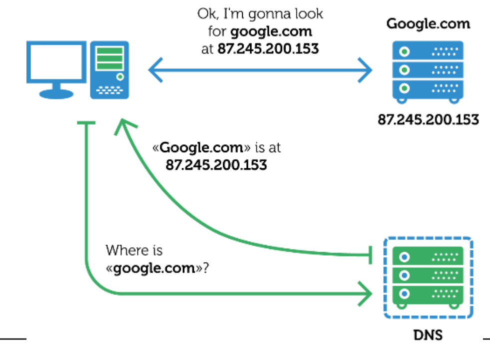
VPNs allow users to store data in a small, private area of the internet, so they can get to it at any time, using an internet connected device.

The benefits are low cost and widely available data, but users must ensure that the data is protected as, without security, their data is available to anyone connected to the internet!

## How DNS works

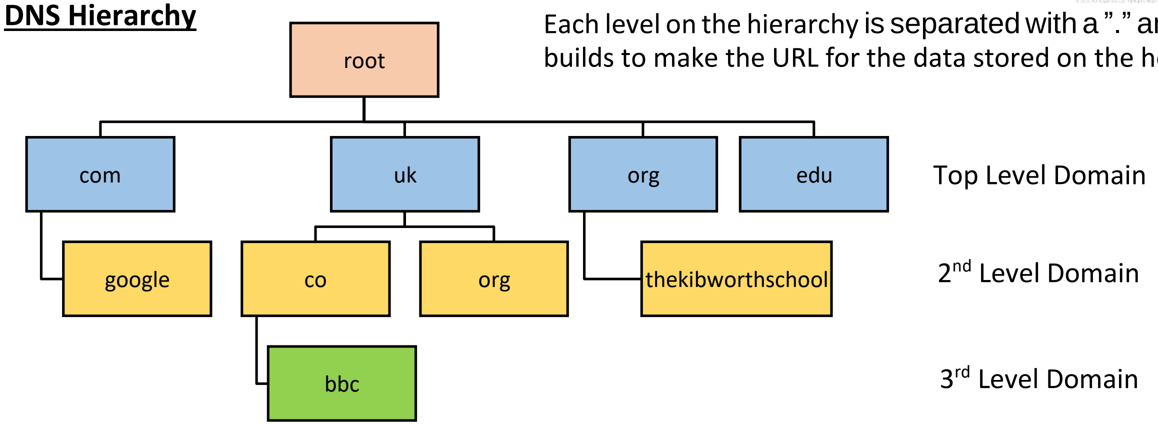
All webpage has an **IP Address** which is a unique reference to find that page. But 87.245.200.153 isn't as easy for users to remember as google.com

When the user types google.com into a web browser's address bar, the client sends a request to the DNS for the current location of google.com. The DNS returns the request, telling the browser to go to 87.245.200.153. The browser now connects to the google server, at the IP address given, and looks for the index.html file to start displaying the webpage.




## DNS Hierarchy

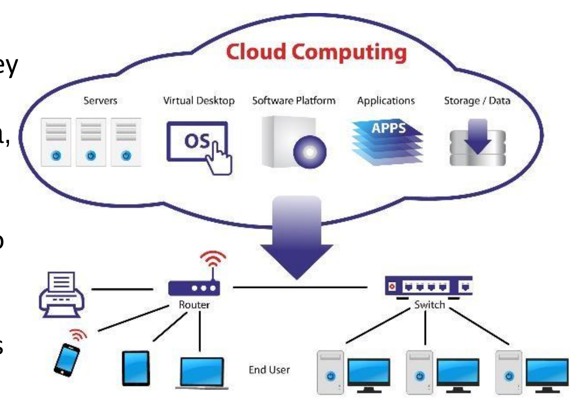
Each level on the hierarchy is separated with a "." and builds to make the URL for the data stored on the host



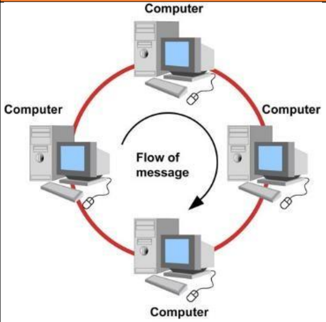
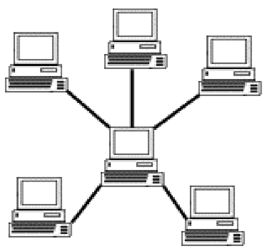
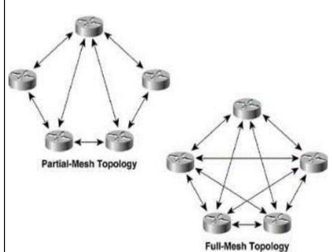
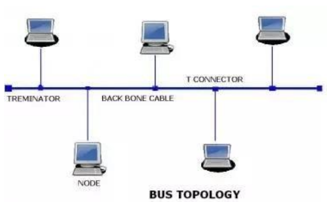

## The Cloud

As our devices are all connected to the internet, they start to become client nodes in a web connected "cloud" network. It's called cloud because your data, services and applications are available everywhere without wires. It's just '*there*' – like a cloud.

PCs like the Google Chromebook utilise the cloud to provide very cheap, very fast hardware, which just connects you to the internet. All the storage, applications and communication is done by services hosted on google's servers.



*1.3 Networks – Topologies*

| Topology means "how a network is laid out and the connections between computers" | | | | |
|---|---|---|---|---|
| NAME | DIAGRAM | DESCRIPTION | ADVANTAGES | DISADVANTAGES |
| Ring |  | Each node is connected to 2 others, and packets tend to travel in 1 direction. | All data flow in 1 direction – greatly reduced chance of collisions.<br><br>No need for network server<br><br>High speed<br><br>Additional nodes can be added without affecting performance | All data passes through every workstation on route<br><br>If 1 node shuts down, then network collapses<br><br>Hardware is more expensive than switches / NICs |
| Star |  | Each node connects to a hub or switch. A central machine acts as **server** whilst the outer nodes are **clients.** | Centralised management through the server<br><br>Easy to add more machines to the network<br><br>If 1 machine fails, the others are unaffected | Potentially higher set up costs, especially in server and switch set ups.<br><br>Central server determines the speed of the network and the number of possible nodes<br><br>If the server fails then the network fails |
| Mesh |  | Every nodes is interconnected with every other, allowing for distributed transmission.<br>Mesh topology can be **FULL MESH** (where every possible connection is made) or **PARTIAL MESH** (at least 2 computers are connected with multiple links) | Multiple devices can transmit data at once, therefore can handle large amounts of data<br><br>A failure of 1 device does not affect the rest of the network<br><br>Adding devices does not impact on data transmission between existing devices | Cost is higher due to increased hardware requirements<br><br>Building and maintaining a mesh network is costly and time consuming<br><br>The chance of redundant connections is very high, which increases the cost, and makes the network cost inefficient |
| Bus |  | Bus or Line topology is a network where all nodes are connected to a single cable (backbone). | Works well with small networks<br><br>Easiest option for connecting nodes with shared peripherals<br><br>Least costly in terms of hardware and cabling | Difficult to fault test because who network crashes when there are errors<br><br>Additional devices slow down the network |

*1.3 Networks – Protocols*

| KEY VOCABULARY | | |
|---|---|---|
| Protocol | The rules and standards that are agreed in order to make it possible for different devices to talk to one another | |
| IP Address | Each node on a network is given a unique 32 bit address (4x8bits) for example 192.168.0.1 There are 4 billion possible combinations. | |
| DHCP | Dynamic Host Configuration Protocol – this protocol allows the network server to control the allocation of IP addresses | |
| MAC Address | Media Access Control<br>Unique addresses hard-coded into the network interface controller. Gives the manufacturer, NIC type and unique identifying number. 48 bits displayed as Hex (eg 01-23-45-67-89-ab-cd-ef) | |
| TCP/IP | Transmission Control Protocol / Internet Protocol | A set of protocols that governs the transfer of data over a network |
| HTTP | Hyper Text Transfer Protocol | Standards for writing webpages to display content for display |
| HTTPS | *Hyper Text Transfer Protocol Secure* | *Client-server protocol for requesting (client) and delivering (server) resources, such as HTML, securely* |
| FTP | *File Transfer Protocol* | *Used to directly send files from one node to another over the internet. Commonly used for uploading files to webservers* |
| POP | Post Office Protocol | Used by email clients to download email from the remote email server and save it onto the users computer. More or less redundant now, and has been replaced by IMAP |
| IMAP | Internet Message Access Protocol | An alternative to POP, allowing more control such as the complete control of remote mailboxes |
| SMTP | Simple Mail Transfer Protocol | An old standard for transmission of email. SMTP can only be used to *push* mail to client machines, whilst both POP and IMAP ae used by clients to *retrieve* mail. |

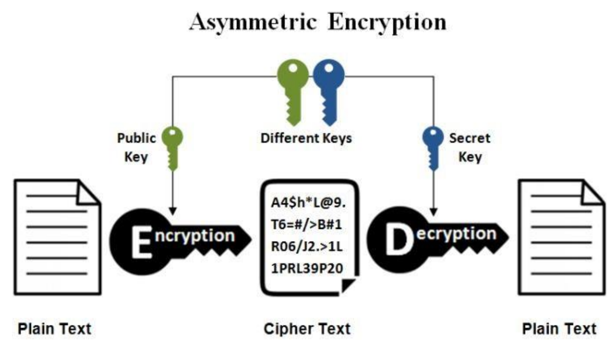
## ENCRYPTION

Encryption is taking a message and changing the letters in such a way that it is not readable. The correct recipient knows how to unscramble the message and can read the text.
Modern encryption is 128bit and secure against brute force attacks

## PUBLIC KEY ENCRYPTION

Public Key Encryption is a method of securely sending data over the internet. The recipient's computer uses an algorithm to produce 2 linked keys: a public key and a private key.

1. Alice (the sender) requests Bob's (the recipient) public key. This is shared.
2. Alice uses Bob's public key to *encrypt* the message she wishes to send
3. The encrypted document is sent over the internet – it is secure.
4. When Bob receives the encrypted document he combines his public key with the secret private key. This allows the message to be decrypted and turned back into plain text



| KEY VOCABULARY | |
|---|---|
| Protocol | The rules and standards that are agreed in order to make it possible for different devices to talk to one another |
| Layering | Rules organised into a distinct order in which they need to be applied |
| Interoperability | The ability for different systems and software to communicate, exchange data and use the information exchanged |
| Encapsulation | Enclosing data inside another data structure to form a single component |
| De-encapsulation | Removing data from inside and encapsulated item. |

## WHY LAYER?

Layering allows problems to be broken down into small chunks, and then smaller solutions created to specific parts of the problem. These small parts interact in an agreed manner, allowing the solution to be built by different teams or companies.

Layering is not unique to computing. In the car industry, a Ford engine might be used with a Jaguar gearbox in a Mazda car. By separating these 'layers', but agreeing on the interface between the layers, each company is free to develop their layer as they see fit, without affecting the other layers. It is also possible to swap one layer out, and replace it with another one – such as swapping an engine for a more powerful one.
This *interoperability* is important as it allows data (in computing) to be passed from one layer to the next.

# TCP/IP Protocol Layers

**LAYER 1: Application**

This layer ensures data is produced in a form that is acceptable to the application that will use it, such as web-browsers or email clients. This is where IMAP or HTML protocols would be used.

**LAYER 2: Transport**

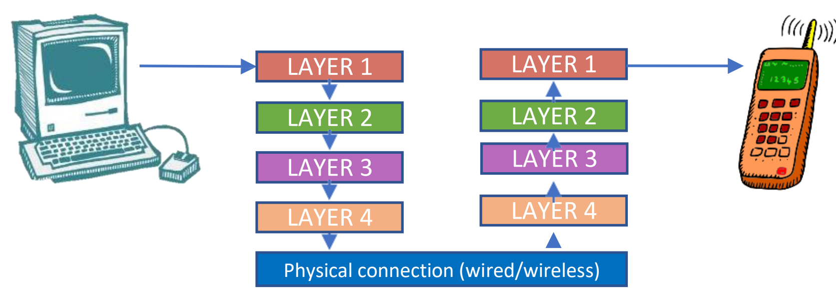
This layer establishes the connection across the network. The transmitting device agrees with the receiving device the speed or data transfer, the size and number of packets and any error checking to be used. This layer uses TCP protocols

**LAYER 3: Internet**

This layer is concerned with transmitting the data across different networks. It identifies the destination and establishes the path the data will take between nodes. It uses IP protocols

**LAYER 4: Network**

This layer deals with the physical transmitting of the data. It converts the data into binary electronic signal that can be understood by the network hardware. It uses protocols such as Ethernet or 802.11 (wireless) so the signal is hardware independent and can use any available compliant physical medium, such as UTP or fibre optic wire.




Data transfer occurs by breaking the file into small *packets*, adding each layer to the packet in order at the sending device, then decoding in reverse order at the receiving device before rebuilding the file.

**Packet switching** is the process that modern networks use to send large data between devices. The data is split into small ***packets*** and numbered. The packets can travel by any route to the destination where the receiving machine reassembles them into the correct order.

*1.4 Network Security - System Security*

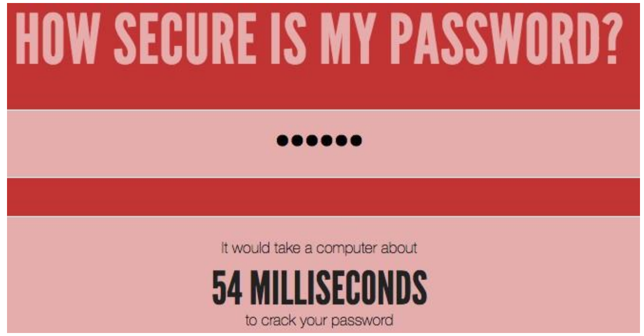
| KEY VOCABULARY - VULNERABILITIES | |
|---|---|
| Hacking | Attempting to bypass a system's security features to gain unauthorised access to a computer |
| Malware | Malware is malicious software, loaded onto a computer with the intention to cause damage or to steal information. Viruses are a type of malware |
| Phishing | Phishing is a common way to try to steal information like passwords. Emails are sent, requesting the user logs into a website, but the site is a fake, and the users details are logged |
| Social engineering | People are the weakest point of any system. If a hacker can convince a user to give over their data, this is the easiest way into a secure system |
| Brute force attack | Using and algorithm to try every possible combination of characters to 'guess' the users password. |
| Data interception | Data interception, or *Man in the Middle attacks* are hacks that use 'packet sniffer' software to look at every piece of data being transmitted in the local area to find ones that meet the hacker's criteria. Often done by creating 'fake' wireless networks to record users details |
| SQL injection | Using SQL statements to trick a database management system (DBMS) into providing large amounts of data to the hacker |
| Denial of Service Attack | Hackers flood a network with huge amounts of fake data and requests in an attempt to overload the system so that it crashes |

| KEY VOCABULARY - PROTECTIONS | |
|---|---|
| Penetration Testing | Employing a *white hat hacker* to try to break into a system to test how good the security is. Any problems in the security can then be fixed before they become vulnerable to real attack |
| Network forensics | Network procedures that capture, record and analyse all network events to discover the source of security attacks |
| Network Policies | Rules which govern how a network may be used – see over page |
| Anti-malware software | Software which analyses files, network traffic and incoming data to look for known malware such as viruses or worms. An infected file is quarantined, and either cleaned or securely deleted to prevent further infection. Needs updating very regularly to ensure that the newest malware is being checked for |
| Firewall | A firewall protects a system by checking all incoming and outgoing network traffic is legitimate |
| User level access | Limiting the access of a user by their requirements to carry out their job. An admin will have more rights than a student, for example. Often even admins do not give themselves full rights to prevent accidents, and will instead have a *super-user* account that will be used only for special high level jobs. |
| encryption | Encoding all data with a secure private, asymmetric key system, so that if data is stolen, it cannot be read or used. |

| TYPES OF MALWARE | |
|---|---|
| Virus | A program designed to infect a computer, then copy itself. Requires human 'help' to spread; usually through infected software being installed or spread through unsecure removable media such as usb-drives |
| Worm | A self-replicating program, which can run itself allowing it to spread very quickly |
| Trojan Horse | A program which disguises itself as legitimate software, and appears to perform one task, but is actually performing another |
| Ransomware | Ransomware secretly encodes a users data and files, then offers to un-encode the files if a large amount of money is paid to the hacker |
| Rootkit | A rootkit allows a hacker to gain full, and often repeated, control of a computer, including the host operating system, which helps the hacker avoid detection |

*1.3 Network Security - System Security: Network Policy*

| COMMON AREAS OF NETWORK POLICY | |
|---|---|
| Acceptable Use | Governs the general use of the computer system and equipment by employees. Usually limited to that which is required to carry out only the tasks that a user is employed to undertake |
| Passwords | Rules to ensure that passwords are strong enough to prevent guessing or brute force attack - often requiring the use of upper and lower case letters, numbers and special characters. Also usually a minimum length is required. Passwords usually have to be changed on a regular basis |
| Email | Governs what may and may not be sent by email |
| Web Access | The configuration of web browsers may limit the types and categories of website that can be accessed |
| Mobile Use | What devices are and are not allowed to be used |
| Remote Access | Govern what can be accessed from outside the system, and what can only be accessed onsite |
| Wireless | Govern how wireless access points (WAPs) are secured, who has access, and under what circumstances |
| Software | Governs who can install software, and which users are able to use that software. May have different levels of access once inside the software |
| Server | Rules about what services are provided by the institution and who may access data stored centrally and for what purposes |
| Back Up | Back up policy determines how frequently back ups are undertaken, and what type of back up (full, incremental, differential). It will also state where the back up media must be stored and for how long. Often a full weekly back up is required to be stored in a fire proof box in an offsite location |
| Incident Response Plan | Details what to do if something goes wrong, or if an attack is discovered. |

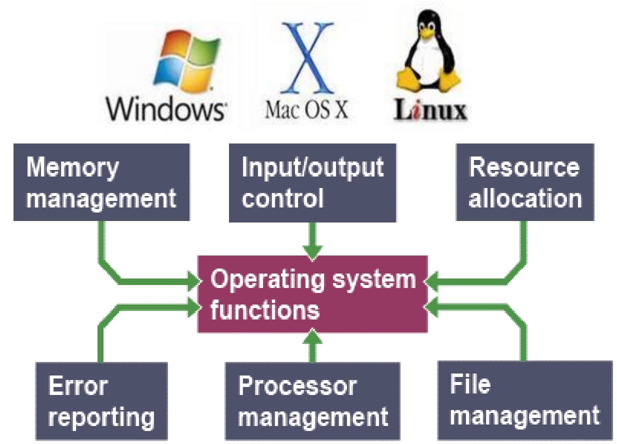


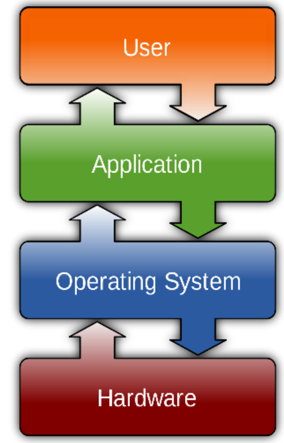

Even modest desktop computers can undertake billions of cycles a second these days. Therefore, without any security features, such as limited password attempts, or asking for only selected characters from a password, a home PC could *brute force crack* commonly used passwords in very, very short periods of time!

PEOPLE ARE ALWAYS THE WEAKEST PART OF A COMPUTER SYSTEM!

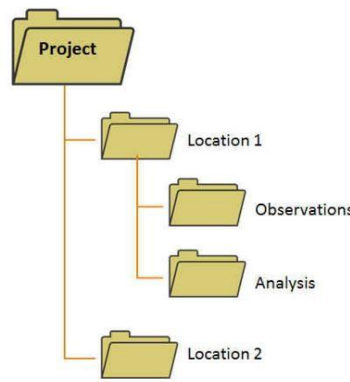
| KEY VOCABULARY | |
|---|---|
| Operating systems (OS) | Collections of programs that tell the computer hardware what to do. |
| User interface | The means of communication between the user and the computer. These are typically either *command line* or *GUI.* |
| Command Line | The most simple form of user interface where users type commands into a prompt |
| Graphic User Interface (GUI) | Most modern computers have a GUI, which uses icons to represent the programs and files. The user runs the programs through a touch-screen or mouse-controlled pointer |
| Voice Command | Increasingly users are able to speak commands to devices such as Google Home and Amazon's Alexa |
| Memory management | The OS controls available memory, moving programs to and from secondary storage to RAM |
| Multitasking | Often users have more than 1 program running at once. In reality, each CPU core can only carryout 1 task at a time, but the OS alternates between the programs to make it appear that multiple tasks are running simultaneously |
| Peripheral management | Computers must communicate with a range of external devices such as printers, monitors and scanners (peripherals). The OS uses *drivers* to correctly pass data to the device and ensure correct function. |
| Drivers | A driver is a piece of software which provides communication between the CPU and a peripherals device |
| User management | Multiple users can have accounts on the same computer, each with their own files, settings and applications, protected with passwords. The OS will ensure that only users who are granted permissions can use files or programs belonging to other users. |
| File management | Computers store files and data in hierarchical folder systems. This is efficient and allows for quick navigation |

**Operating Systems**



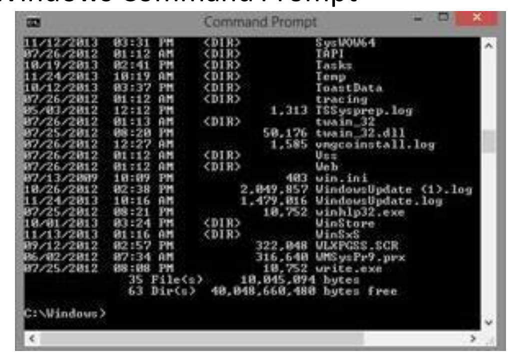




**File Management**




**User Interface**

Windows Command Prompt




Apple Graphical Interface

| KEY VOCABULARY | |
|---|---|
| Utility Software | Utility software supports the OS by performing a limited and specific task. They are used to manage specific actions of the system, or undertake maintenance operations. |
| Encryption software | In order to keep data secure, especially against outside threats, data must be encrypted. Encryption software uses complex algorithms to encode data so it cannot be read without the private access keys. |
| Disk Defragmentation | Over time, through multiple updates and saves, files will become split up and distributed over the platters. It takes longer for the files to be accessed, slowing the machine down. Defragmentation reorganises the files' parts to bring them together. See fig 1. |
| Data Compressions | Allows files to be made smaller by removal of empty space or through compression algorithms (lossy or lossless) – see KO2.6b |
| Back Up | In case of hardware failure or other computer problems, data should be copied to external media so that it can be restored if lost or damaged. |
| Antivirus | Continually scans the system to find, quarantine, and clean any file infected with viruses. |
| Anti-malware | Continually scans to identify any malicious software from being introduced to the system. |

| TYPES OF BACK UP | | | |
|---|---|---|---|
| | *Description* | *Positives* | *Negatives* |
| **Full** | All files and folders are backed up every time | Only requires last back up to restore; quickest to restore | Requires the most space on back up drive; slowest to back up |
| **Incremental** | Only new files or files that have been changed since the last back up are copied | Faster to back up; requires less space; does not store duplicate files | Slowest to restore; needs at least one full back up to start |

**DISK DEFRAGMENTATION:**

Over time, as new files get added, old ones deleted and files increase through use, the parts of files get separated around the HDD. (A to B) This separation causes computer slow-down.

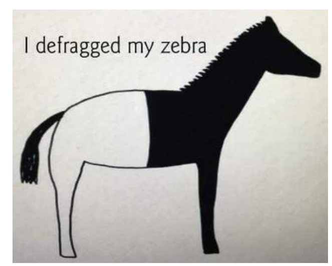
In order to improve performance, disk defrag applications shuffle file parts back into order, and moves all free space to the end of the drive.(C)

This improves data access times and overall system performance.

A

B

C




A computing joke…getit?

| KEY VOCABULARY | |
|---|---|
| Ethical | Relates to *right and wrong* but in a moral sense than a legal issue. For example, there is nothing to stop you legally from using Facebook to stalk an ex-partner, but whether it is *right* to do so, is an ethical issue |
| Legal | There are certain laws set by government that control how computers can be used – see box |
| Cultural | These issues relate to society and how technology can affect religious, or social ideas. If people spend all their time on their phones rather than talking face to face, this is a cultural issue |
| Environmental | How computing impacts on the global and local environments. This might be waste production, or mining to gather resources needed to make phones, or using renewable energy to charge phones, or recycling projects. Companies want to be seen to be 'green'. |
| Privacy | Privacy is a very important issue. A persons right to privacy is very important and there are strong law, alongside ethical guidance that govern how companies can use our data |
| Stakeholder | Anyone that is impacted on, in any way, by a technology. They have a vested interest |
| Open source | Software that is created and shared with the source-code able to be seen. Users are free to make alterations to the source-code to meet their own needs, or to improve the system for everyone |
| Proprietary | Software that is created but the source code is locked. This is often sold and the company wants to protect its intellectual copyright |
| Legislation | Laws that relate to a certain area |

| COMPUTING LEGISLATION | |
|---|---|
| The Data Protection Act (1998) | Sets out how data users who store data about individuals must use that data. It is a set of 8 principles which say how personal data must be collected, used and destroyed. See back of sheet |
| Computer Misuse Act (1990) | Introduced to deal with the increase in computer hacking in the late 1980s when home PCs started to become popular. It aims to protect computer users against willful attacks and theft of information. The Act makes it illegal to:<br>• gain unauthorized access to another person's data<br>• …with the intention of breaking the law further<br>• ….to delete, alter or sabotage by introducing viruses |
| Copyright and Design Patents Act (1988) | Provides the creators of intellectual property (ideas = IP) with proof of ownership, and the exclusive rights to use that idea, and distribute their work. It makes it illegal to copy, modify or distribute IP without permission |
| Freedom of Information Act (2000) | FOI requires public organisations to publish certain data so the public can access it. It also give individuals the right to request to see all data from over 100,000 public bodies.<br>The act covers all electronic information, such as word docs, emails, digital records. Organisations can withhold certain information if releasing it would affect national security |
| Creative Commons Licensing | Creative Commons Licensing (CC) is a way that copyright holders can grant certain privileges to publicly use, share, adapt, alter and redistribute IP without written permission. |

| OPEN SOURCE vs PROPRIETARY SOFTWARE | |
|---|---|
| Open source software is freely available so others can use it. Users can access and modify the source-code and create their own versions. | Proprietary software is not freely available. The compiled code is secured and user must use the software as provided. Any attempt to modify, copy or redistribute the software is a breach of Copyright. |
| EXAMPLES:<br>Linux, Firefox, Android OS | EXAMPLES:<br>Microsoft Office, Adobe Photoshop, OSX |

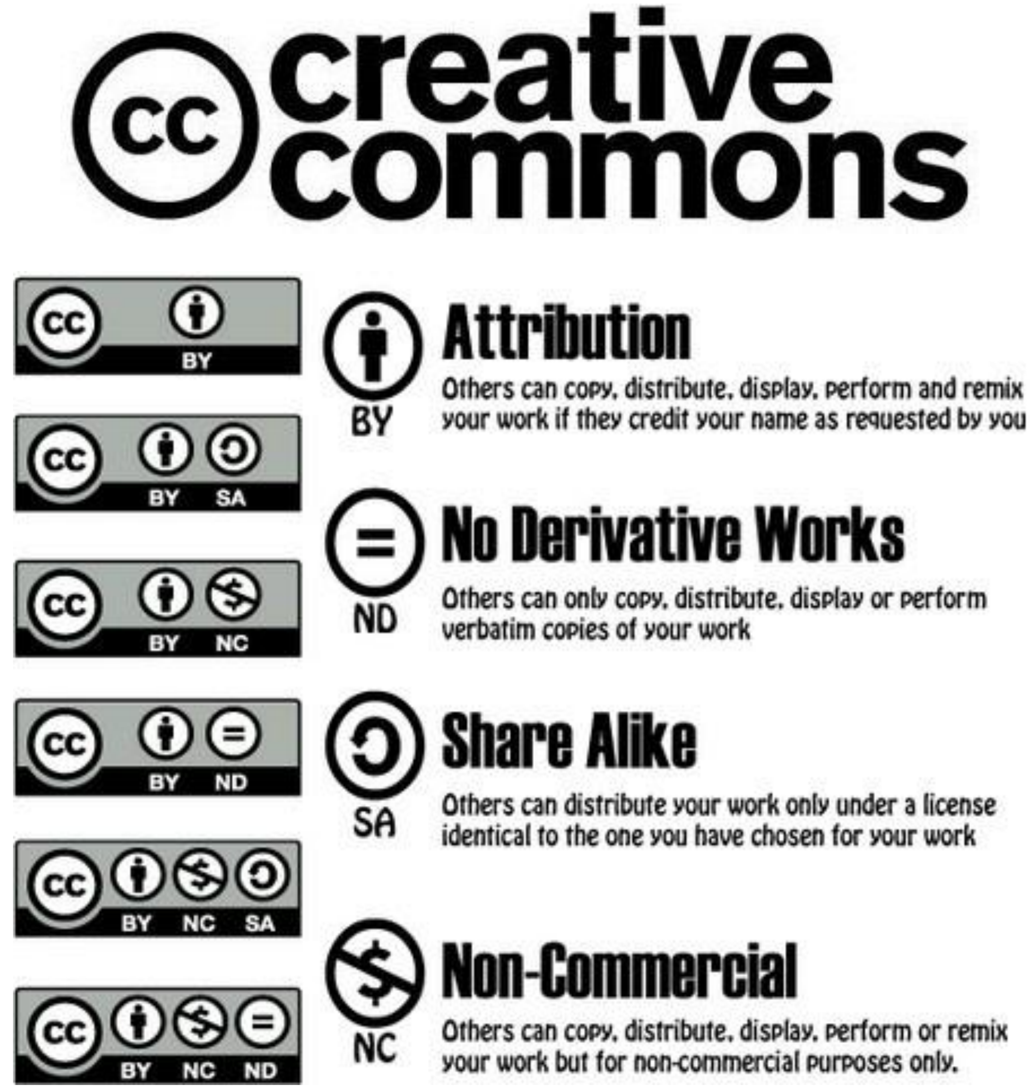
# Data Protection Act (1998)

## What are the eight principles of it?

1. Data must be kept secure;
2. Data stored must be relevant;
3. Data stored must be kept no longer than necessary;
4. Data stored must be kept accurate and up-to-date;
5. Data must be obtained and processed lawfully;
6. Data must be processed within the data subject rights;
7. Data must be obtained and specified for lawful purposes;
8. Data must not be transferred to countries without adequate data protection laws.

## TYPES OF HACKER

**Black Hat** – The Bad Guys. They break into systems to cause chaos and steal data for their own benefits
**White Hat** – Penetration Testing professionals. Often employed by companies to test systems and provide feedback on security
**Grey Hat** – Not trying to cause damage, but aren't trying to help either.
**Red Hat** – Scary people – stop Black Hat hackers by revenge hacking and destroying the hacker's system
**Green Hat** – n00bz trying to learn hacking. Often just download scripts from the internet and run them without understanding the code. Often exploited by Black Hat hackers to do stupid things

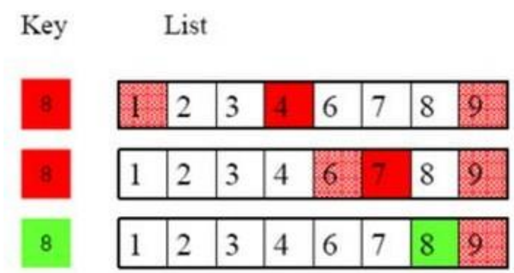


*2.1 Algorithms: Searching and Sorting*

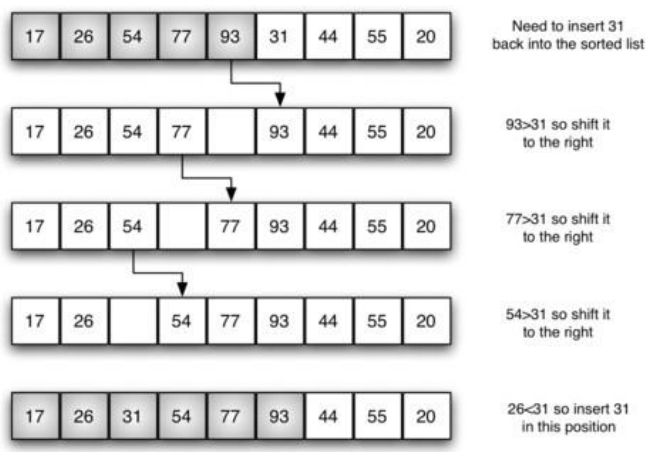
| KEY VOCABULARY | |
|---|---|
| Algorithm | An abstracted program which completes a given task, whatever the data provided |
| Search | Searching is looking through data, making comparisons with a search term, until the algorithm either finds the data, or identifies that it is not present. |
| Sort | Putting given sets of data into specified order – usually ascending (alphabetical) or descending (reverse alphabetical) |
| Linear Search | A type of search where the computer checks every variable, in order, until it finds the search term. Potentially very slow. |
| Binary Search | A search type based on repeatedly halving the searchable data, until the search term is found |
| Bubble Sort | A method of sorting data which looks at pairs of variable, and swaps them around if out of order. This continues until there are no more swaps to be made |
| Merge Sort | Splits the data into increasingly small segments, until single data points are reached, then reassembles the data structure one item at a time. |
| Insertion Sort | Checks through the data until finding the first incorrectly places item. The algorithm then checks all the previous places to see where the data fits, before inserting it into this slot. |

LINEAR SEARCH

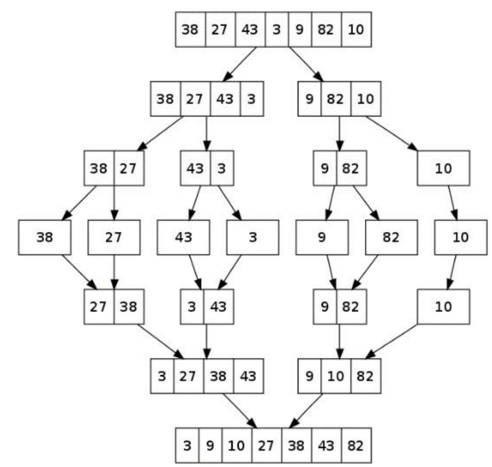

BINARY SEARCH



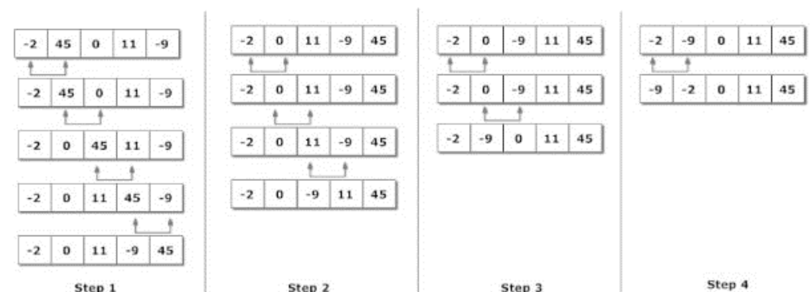

INSERTION SORT




MERGE SORT



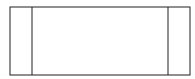
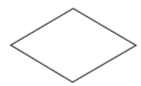
BUBBLE SORT



| KEY VOCABULARY | |
|---|---|
| Algorithm | An abstracted program which completes a given task, whatever the data provided |
| Abstraction | Abstraction is moving a problem out of the specific in order to create a general solution that would work in similar scenarios. Ignoring the gritty details to focus on the problem |
| Decomposition | Breaking a problem down into smaller, computational solvable chunks |
| Pseudo Code | A structured way of planning code, which is 'computational' in style (uses Boolean logic, variables, comparisons and arithmetic for example) but is not tied to a strict high-level language's syntax |
| Flow Diagram | A diagram, made using specific shaped boxes, that mocks up the flow of a program through various stages, processes and decisions. |
| Program Control | Using Boolean logic to guide the computer through a program based on decisions |
| Comparison Operators | The symbols used to look at a variable or piece of data in relation to is similarity to another piece of data or variable |
| Arithmetic Operators | The symbols used to show the mathematics to be carried out on a piece of data |

## Flow charts

Flow charts like pseudocode are informal but the most common flow chart shapes are:

| | | |
|---|---|---|
|  | **Line** | An arrow represents control passing between the connected shapes. |
|  | **Process** | This shape represents something being performed or done. |
|  | **Sub Routine** | This shape represents a subroutine call that will relate to a separate, non-linked flow chart |
|  | **Input/Output** | This shape represents the input or output of something into or out of the flow chart. |
|  | **Decision** | This shape represents a decision (Yes/No or True/False) that results in two lines representing the different possible outcomes. |
|  | **Terminal** | This shape represents the "Start" and "End" of the process. |

Comparison operators

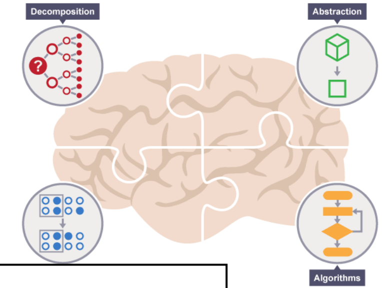
| | |
|---|---|
| == | Equal to |
| != | Not equal to |
| < | Less than |
| <= | Less than or equal to |
| > | Greater than |
| >= | Greater than or equal to |

Arithmetic operators

| | |
|---|---|
| + | Addition e.g. `x=6+5` gives 11 |
| – | Subtraction e.g. `x=6–5` gives 1 |
| * | Multiplication e.g. `x=12*2` gives 24 |
| / | Division e.g. `x=12/2` gives 6 |
| `MOD` | Modulus e.g. `12MOD5` gives 2 |
| `DIV` | Quotient e.g. `17DIV5` gives 3 |
| ^ | Exponentiation e.g. `3^4` gives 81 |

**Computational thinking**

To be able to represent a problem as a set of steps that can be carried out by a computer requires good computational thinking skills. These steps could be presented as algorithms. ***A computer cannot do computational thinking – we do computational thinking so that we can create solutions, to program computers***




**Decomposition**

Programmers use a technique called decomposition to break a large program down into a series of sub problems.

The starting point is to decompose the large problem so that each sub problem is described in the same level of detail and can be solved independently from the other sub problems. The solutions to the sub problems can then be brought together to provide a solution to the whole problem.

One of the advantages of decomposition is that different people can work on different sub problems. However, a disadvantage would be that the solutions to the sub problems might not come together to provide a solution to the whole problem.

If a problem is not decomposed, it is much harder to solve.

Articulate!

- Get into groups of 4, 2 vs 2!
- Each pair gets a pack of articulate cards.
- You have **30 seconds** to try and describe as many items as you can (use the clock feature to time).
- One partner must describe what is on the card without using the name of the item. If the partner answers correctly, the team get a point.
- If the name of the item is used the team get -1 point.
- The team with the most points at the end wins!
  Only one pass is allowed per turn. Use it wisely!

**Pattern Recognition**

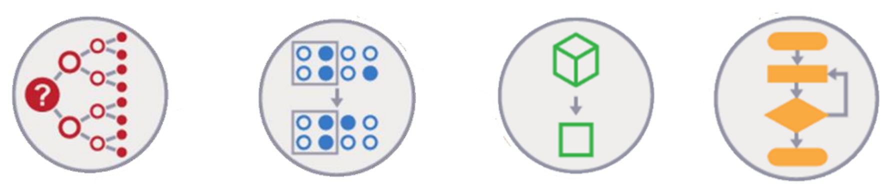
When we decompose a complex problem we often find patterns among the smaller problems we create. The patterns are similarities or characteristics that some of the problems share. Pattern recognition is one of the four cornerstones of Computer Science. It **involves finding the similarities or patterns among small, decomposed problems that can help us solve more complex problems more efficiently**.

The more patterns we can find, the easier and quicker our overall task of problem solving will be.

To find patterns among problems we look for things that are the same (or very similar) for each problem.

1. Consider a typical week. Write a list of the things you do that are the same from day to day.

2. How do you think having a pattern with these consistencies benefits you?



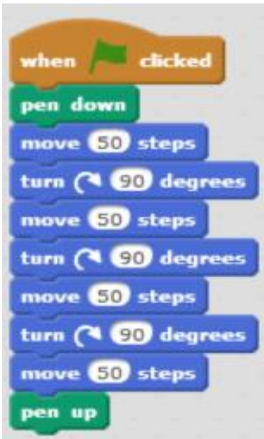
## Abstraction

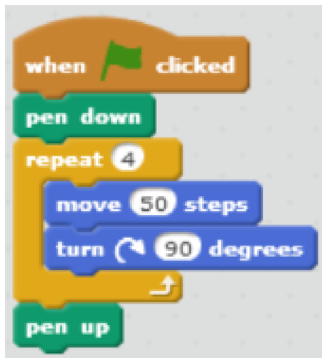
Abstraction is a technique to reduce something to the simplest set of characteristics that are most relevant to solving the problem.

The programmer has to concentrate on the most important aspects of the problem without worrying about fine details.

### A simple example of abstraction

I want to create a program to draw a square in Scratch. The square must have four sides of 50 units with an angle of 90º between each side. The code in Scratch looks like this:




There are a lot of repeated commands in the code. It can be simplified by using a loop for the repeating code.

This code can be defined as a procedure that can be called whenever it is needed.



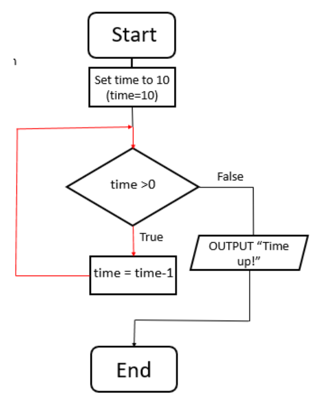
So instead of considering a series of issues or commands the problem has been simplified down to one procedure without all the unnecessary details that have been hidden

## Algorithm Design

**An algorithm is a plan, a set of step-by-step instructions to solve a problem.** If you can tie shoelaces, make a cup of tea, get dressed or prepare a meal then you already know how to follow an algorithm.

In an algorithm, each instruction is identified and the order in which they should be carried out is planned. Algorithms are often used as a starting point for creating a computer program, and they are sometimes written as a **flowchart** or in **pseudocode**.

The aim of the following loop statement is to create a countdown clock. The timer is set to 10 then each loop through it reduces the number by 1 until it gets to 0 and displays "Time up!"



| KEY VOCABULARY | |
|---|---|
| Variable | A piece of stored data, used in a computer program, which can be changed or altered by the program |
| Constant | A piece of stored data which cannot be changed by the program or user |
| Operator | An operator is a mathematical symbol, used to work with data in a program |
| Input | Data, entered into a program, by the user |
| Output | The returned result of an algorithm |
| Algorithm | A set of instructions to carry out a process or problem-solving operation, especially by a computer |
| program control | Selection of code to be executed, based on the results of prior operations in a program, or user input |
| Loop | A piece of repeating code |
| Iteration | A type of **LOOP** which repeats a series of steps with a finite number of variable changes |
| Sentinel | A type of **LOOP** that watches a variable for a logical (T to F, or F to T) and repeats until that change occurs |
| Conditional | A method of controlling the information flow through branching steps – the code checks if something is True, then carries out one set of instructions if it is, and a different set of instructions if it is False. |
| Sequence | A series of coded instructions for a computer to follow, step by step |
| String | A character, or characters, stored as a list, within “ ”. |
| Integer | A whole numbers, stored as its value |
| Real | A decimal number, stored as its value |
| Boolean | True or False. Stored as 1 or 0. |

<table>
<tr><th colspan="2">KEY VOCABULARY</th></tr>
<tr><td>Declaration</td><td>Assigning a value to a variable</td></tr>
<tr><td>Typecasting</td><td>Casting a variable as and integer, Bool, Float or String</td></tr>
<tr><td>Data Arrays</td><td>‘Lists’ of data, stored in an indexable table format<br><br>1 D ARRAY:<br>
<table>
<tr><td>C</td><td>O</td><td>D</td><td>I</td><td>N</td><td>G</td><td>E</td><td>E</td><td>K</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td></tr>
</table>
← single row of elements</td></tr>
<tr><td>2D Arrays</td><td>A data structure which has more than 1 ‘row’ of data.<br>2D arrays use 2 indexes to identify data<br><br>IMPORTANT!!!<br>2D arrays use the Y axis first in the co-ordinates, then the X axis. This is the opposite way around to most other co-ordinates!<br>
<table>
<tr><th></th><th>Column 1</th><th>Column 2</th><th>Column 3</th><th>Column 4</th></tr>
<tr><td>Row 1</td><td>a[0][0]</td><td>a[0][1]</td><td>a[0][2]</td><td>a[0][3]</td></tr>
<tr><td>Row 2</td><td>a[1][0]</td><td>a[1][1]</td><td>a[1][2]</td><td>a[1][3]</td></tr>
<tr><td>Row 3</td><td>a[2][0]</td><td>a[2][1]</td><td>a[2][2]</td><td>a[2][3]</td></tr>
</table>
</td></tr>
</table>

| KEY VOCABULARY | |
|---|---|
| Defensive design | Planning a program from the very beginning to prevent accidental or purposeful misuse |
| Input sanitization | Removing erroneous data from a system prior to processing |
| Data validation | Ensuring all data is in the correct format prior to processing |
| Contingency planning | Having built in checks and outcomes based on what happens when things go wrong |
| Anticipating misuse | Building programs which do not allow a user to deliberately break the system |
| Authentication | Having different levels of user, and preventing everyday users from being able to significantly change a system |
| Maintainability | Building software which is modular to enable sections to be updated and replaced without having to write the whole program again from scratch |
| Code comments | Annotating code so that the person maintaining or working with your code in the future is able to understand your thought process |
| Indentation | Making code more readable by laying it out in a manner that keeps sections of code separate |
| Iterative testing | Step by step testing to ensure that small sections of the code work, before new parts are added and then retested. Important to allow *traceback* to find what caused any errors |
| Terminal testing | Significant testing done once a program is complete under a range of conditions and on multiple hardware – often called *Alpha Testing* |
| Beta Testing | Making a small release of the software to a group of tech-literate enthusiasts to broaden the usage-testing and get lots of feedback prior to full release. |
| syntax error | An error in the typing of the code. Missing punctuation, spacing etc |
| Test data | Data chosen to test the program. Testers use a specific range of data |

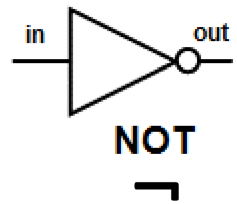
| TESTING DATA | |
|---|---|
| Data Range | The data that will be used to check the code works correctly |
| Valid Data | Obvious data which should definitely pass |
| Valid Extreme | Unusual data – the highest and lowest data – on the very edge of what should pass |
| Invalid Extreme | Data, of correct type, which is on the very edge of what should fail |
| Invalid Data | Data, of the correct type, that should definitely fail |
| Erroneous Data | Data that is the wrong type and should fail |
| Expected Outcome | The data the code should output if it is running correctly |

| ERROR TYPES | |
|---|---|
| Syntax Error | An error in the code – incorrectly typed, missing punctuation etc |
| Logical Error | An error which, although allows the code to run, produces incorrect outcomes |
| EOF Error | The *End of File* has been reached, whilst the computer is waiting for a snippet to be completed. |
| Type Error | Attempting to use data incorrectly – adding 1 to a string etc |
| Name Error | Using a variable before its declaration |
| Indentation Error | Loops or functions are incorrectly indented |

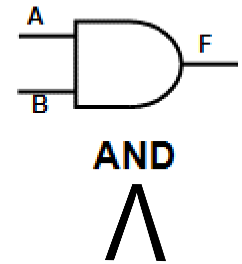
| KEY VOCABULARY | |
|---|---|
| Logic | A system designed to perform a specific task according to strict principles. |
| Logic Gates | The physical switches inside an electronic device which are able to perform the calculations a computer needs to carry out on electronic signals |
| Truth Table | A tabular representation of the possible inputs and outputs from a given logic gate, or collection of gates |
| Boolean | Mathematical *TRUE or FALSE* |
| Operator | A mathematical symbol in computing |
| + | Addition [ 1+2=3 ] |
| - | Subtraction [ 2-1=1 ] |
| / | Division [ 5 / 2=2.5 ] |
| * | Multiplication [ 2 * 2 = 4 ] |
| ^ | Exponentiation, raising a number to the power of… [ 3^3 = 3 * 3 * 3 = 27 ] |
| MOD | Modulus division. To divide a number by another, but only return the *remainder* [ 10 MOD 3 = 1 ] |
| DIV | Integer Division. To divide a number by another, but only return the *number of full sets.* [ 10 DIV 3 = 3 ] |

LOGIC GATES

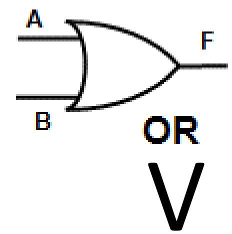
These gates take inputs (usually labelled A, B, C etc, and provide a single output. In this case labelled F, but could be another letter. Each gate is shown with its TRUTH TABLE




| Input | Output |
|---|---|
| I | F |
| 0 | 1 |
| 1 | 0 |



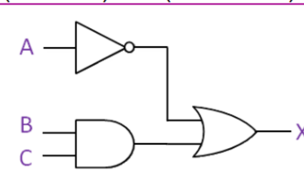

| Inputs | | Output |
|---|---|---|
| A | B | F |
| 0 | 0 | 0 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 1 | 1 |




| Inputs | | Output |
|---|---|---|
| A | B | F |
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 1 |

COMBINED GATES – Logic gates can be combined in any order to provide a range of computational possibilities. Inside a CPU, the physical switches are logic gates, and but combining them in different sequences, computers can undertake incredibly complex mathematics with these very simple tools.

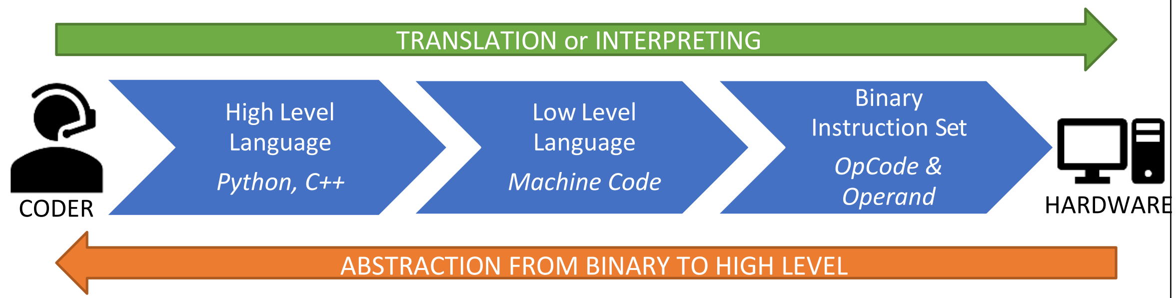
( NOT A ) OR ( B AND C )




| A | B | C | NOT A | B AND C | X = (NOT A) OR (B AND C) |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 |

| KEY VOCABULARY | |
|---|---|
| Low Level Language | A programming language which is closer to binary than English |
| High Level Language | An abstracted programming language which is closer to English than binary |
| Instruction Set | Binary code which tells the computer hardware what to do – OpCode and Operand |
| Machine Code | 1 to 1 instructions coded in mnemonics (STO, ADD, MOD, DIV etc) which must be converted to binary to run |
| Abstraction | Removing a level of detail to allow focus on the problem solving rather than the specifics. *Python, and all other High Level languages are abstracted. You do not need to know the machine code to get something to happen* |
| Translator | A utility to convert High Level Code into binary machine code so it can be executed |
| Interpreter | A utility which translates High Level code on a line by line basis and executes the program as it goes in a special test environment |
| IDE | Integrated Development Environment |
| Text Editor | A place to type code, focused on the content of the file, not the look of the file |
| Error Diagnostics | To test a program and provide feedback to the coder so that errors can be fixed |
| Run Time Environment | Part of an IDE which allows a piece of code to be tested without installation |

## Working the Machine:




For coders to be able to write code quickly, high-level coding language have been made which allow the coder to use *almost* natural language (like English) to solve problems. These *ABSTRACTED LANGAUGES* must be converted into binary code instructions that the CPU can execute in order to work. This conversion of instructions is done in 1 of 2 ways. They are either *interpreted*, one line at a time, and executed immediately, or they are *translated* by converting the entire code file in one go, then attempting to run the program only once the converter has finished *compiling. c*

## Features of an Integrated Development Environment (IDE)

| FEATURE | PURPOSE and BENEFITS |
|---|---|
| Text Editor | An IDE's text editor is where the code is typed. It is not concerned with the look of the code, but usability. Additional features of IDE text editors are: line numbers, code colouring by context, automatic indentation, autocomplete, code-folding, overview 'map', multiple cursors |
| Error Diagnostics | IDEs will give real-time feedback to the coder to show any obvious errors before compiling. These are often with highlighting or line markers. Additionally, any errors which show up during compiling are flagged with helpful guidance to the coder about the error type and the line number |
| Compiler | A utility which attempts to turn the program into a runnable program. This will either be a translator/compiler or an interpreter |
| Run-Time Environment | A 'safe sandbox' where code can be tried out without installing it to the computer. Often ring-fenced from the main machine to prevent accidents. |