Computer Science GCSE Revision Guide

Cornwallis *Academy*

Tomorrow's Future Today

1. Algorithms

An *algorithm* is simply a set of steps that describes how a task is carried out An *algorithm* takes **input** data which it uses to produce **output** data *Every time* you write a computer program, you are creating an *algorithm* – defining a set of steps to perform a task

Basic Constructs

Sequence	A *sequence* is a linear progression where one task or *instruction* is performed sequentially, one after another in the order that they are written.
Selection	Usually seen as an **IF-THEN-ELSE.** A decision is made between two alternative courses of action (sets of *instructions*)
Iteration	**LOOPs** – WHILE, REPEAT-UNTIL, FOR. *Instructions* are repeated until a condition is met.

Pseudocode

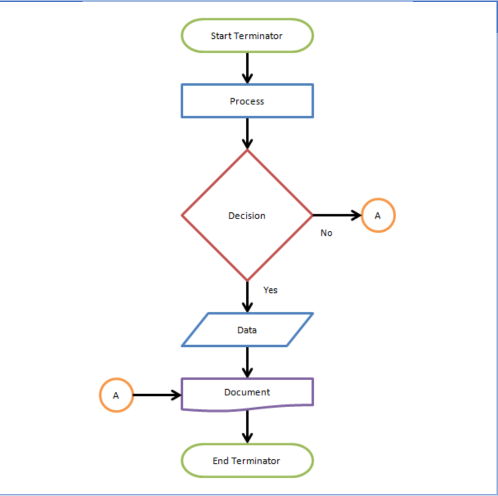
Pseudocode consists of *natural language-like statements* that precisely describe the steps of an algorithm or program. Some common action keywords are:

Input	READ, OBTAIN, GET
Output	PRINT, DISPLAY, SHOW
Compute	COMPUTE, CALCULATE, DETERMINE
Initialise (variables)	SET, INIT
Add one	INCREMENT, BUMP

Flowcharts

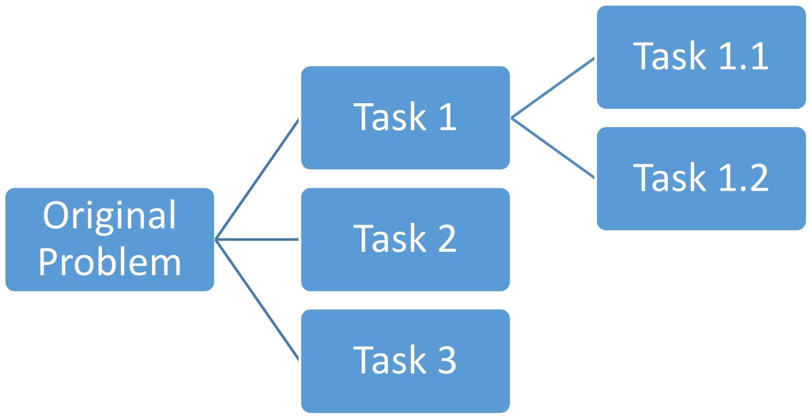
Algorithms (and therefore computer programs) can be expressed as a set of symbols that show the order that actions and calculations are carried out.

Symbol Name	Symbol	Description
Start / Stop (Terminator)	⬭	Used to denote the starting/finishing point of a flowchart
Process	▭	Used to denote an operation that is carried out on some data
Input / Output	▱	Used to show the results of an operation or collect data from sensors or other sources
Manual Input (Keyboard)	⏢	Used to show user interaction, such as pressing a button or entering information
Decision	◇	Used to show a Boolean choice; there should always be 2 paths out



2. Decomposition

Decomposition is a general approach to solving a problem by **breaking it up into smaller tasks** and solving each of the smaller tasks (sub-tasks or sub-problems) **separately.**



3. Developing Code

High-level Programming Languages

A high-level language is a programming language that *resembles a natural language*. Each instruction translates to many machine instructions.

High-level languages are easy to learn and use because they use English-like code.

Instructions in a high-level programming language are called **Statements.**

Computers cannot understand code written in a high-level language on their own. This **source code** must be *translated* into machine code using either an **interpreter** or a **compiler**.

Interpreter – reads one *statement* from the source code, translates it into *machine code* and then executes it.

Compiler – translates the entire source code into a machine code file. This file can then be executed.

What makes good code?

Programs should work and should be efficient. Code should be written to be as readable for others as possible so that it can be updated or amended at a later date, and by other people than the original author.

This can be achieved by grouping similar **functions** together within the source file, and also by:

Comments – descriptive or explanatory text in the source that is ignored by the compiler or interpreter.

Descriptive Variable Names – Using names for variables that describe their function can make code more readable. Consider: t1 vs hours_worked_total as variable names.

Indentation – Indenting code with spaces or tabs to show statements that will run under certain circumstances

Errors in computer programming

There are 3 primary types of errors that programmers may encounter when writing computer programs:

Syntax Errors – occur when a mistake has been made with the **grammar rules** of the programming language; variables may have been used with the wrong case, punctuation added in the wrong place or use of **reserved words** that do not exist. Programs with a syntax error will **fail to compile or execute** as they do not define a valid program.

Runtime Errors – occur when a program instructs a computer to carry out an operation that it cannot do (such as divide by zero). Programs with runtime errors by definition do compile and execute, but may crash or behave in unexpected ways.

Logic Errors – occur when a programmer does not fully understand how the program was supposed to behave, or has made a careless mistake. Programs with logic errors compile and execute but will not behave as expected - the computer does not know what the program is *supposed* to do. Common logical errors are reversed Boolean logic **a s** part of an **if-statement** (testing for less than <, instead of greater than >, for example). Logic errors can be difficult to detect and fix.

Constructs

Programs must have structure in order to be understandable (both by humans writing them and computers running them!). There are 3 primary constructs employed in computer programming:

Sequence – Instructions are executed in the order they are written. Sequences can contain any number of statements but no actions may be skipped in execution.

Selection – Different instructions are executed depending on a particular condition. The most common selection construct is the **if-else** statement.

Repetition – A set of instructions are executed repeatedly, usually until a particular condition is met. More commonly called **Loops**. **For-loops, while-loops** and **repeat-until** loops are common.

Additionally, structure can be given to programs by breaking these up into **subprograms**. A subprogram is a program that can be *called by another program* to perform a particular task or function for the calling program.

Data types and structures

Data can be stored in many different forms. These forms are called *data types*.

Computers use special internal codes to keep track of the different types of data that it processes. **Primitive** data types are predefined types of data which are supported by a programming language:

Character – a single alphabetic character or symbol

Integer – a whole number

Floating-point number – a number with a decimal point, sometimes also called a **real number**

Boolean – logical values, either *True or False.*

The simplest data structure is a **variable**. A variable holds a piece of data that may *vary*, it can be changed by the program.

Arrays are a type of variable which allows the storage of multiple, related values, accessible via an array **index**. In almost all programming languages, arrays are arranged with the first value pointed to by **index zero (0)**.

4. Binary

Computers store all of their information using REALLY tiny little switches.
These switches can be either **on**, or **off**. Something that can only have 2 'states' is said to be **Binary**.
Computers use Binary to represent all of the *data and instructions* that they store.

8 Bit Binary Numbers

128	64	32	16	8	4	2	1

Lets look at the number 27. The way we normally write numbers is called DENARY. We can build a similar table to the one above for denary. You have probably seen one like it at Primary School:

Hundreds	Tens	Units
0	2	7

A computer stores this number in memory using binary, it looks like this:

128	64	32	16	8	4	2	1
0	0	0	1	1	0	1	1

The 'switches' for number 16, 8, 2 and 1 are all turned 'on'. 16+8+2+1 is 27. 00011011 is the number 27 in binary.

Using 8 bits, a computer can represent the positive integers from 0 to 255.

Have a go

Convert the number 33 into binary

128	64	32	16	8	4	2	1

Negative Numbers

In mathematics, representing negative numbers is easy – we can simply add a minus sign to the number. Computers can't do this – they just have ones and zeros. Instead a computer can use one of the bits to represent a numbers sign:

Sign & Magnitude

+/-	64	32	16	8	4	2	1

This is called sign & magnitude representation. If there is a 1 in the left-most bit, the number is negative. Notice that by doing this, the maximum possible value we can represent has been reduced, from 255 to just 127.

Sign & Magnitude representation is not usually the best way to represent negative numbers – it takes a computer more processor cycles to deal with sign bit because it is separate from the number being stored and has to be treated differently when doing calculations.

Two's Complement

Another common way of representing negative numbers is two's complement. Instead of a sign bit, the left-most bit is used to represent the negative number -128:

-128	64	32	16	8	4	2	1

When we need to represent a negative number, we can start at -128 and add on the other numbers to get to the number we need. For example, the number -27:

-128	64	32	16	8	4	2	1
1	1	1	0	0	1	0	1

As before, this can be seen as: -128 + 64 + 32 + 4 + 1 = -27

Addition and Subtraction

Adding and subtracting numbers in binary is very similar to adding denary numbers together in maths:

128	64	32	16	8	4	2	1
0	0	0	0	1	0	0	1
0	0	0	0	0	1	0	1

Rather than simply working out the denary equivalents and adding those together, we can simply add the binary digits according to a set of simple rules. **We always start at the right when adding binary.**

So, to begin with the '1' column:

128	64	32	16	8	4	2	1
0	0	0	1	1	0	0	1
0	0	0	1	1	1	0	1
							0
						1	

1 + 1 is zero, carry the one. When working with Binary, we can only have the value zero or 1. Just like when you add 1 to 9, you need to use a new significant digit (units, tens, hundreds etc). So we carry a 1 into the next column:

128	64	32	16	8	4	2	1
0	0	0	1	1	0	0	1
0	0	0	1	1	1	0	1
						1	0
						1	

0 + 0 + 1 = 1, that's an easy one! Continue with the '4' column:

128	64	32	16	8	4	2	1
0	0	0	1	1	0	0	1
0	0	0	1	1	1	0	1
					1	1	0
						1	

0 + 1 = 1, another easy one. Column '8':

128	64	32	16	8	4	2	1
0	0	0	1	1	0	0	1
0	0	0	1	1	1	0	1
				0	1	1	0
			1			1	

This is the same as the first one, 1 + 1 = 0, carry 1. The next one introduces the last rule:

128	64	32	16	8	4	2	1
0	0	0	1	1	0	0	1
0	0	0	1	1	1	0	1
			1	0	1	1	0
		1	1			1	

1 + 1 + 1 = 1, carry 1. Finally, the '32' column:

128	64	32	16	8	4	2	1
0	0	0	1	1	0	0	1
0	0	0	1	1	1	0	1
		1	1	0	1	1	0
		1	1			1	

0 + 0 + 1 = 1. The last two columns are both 0 + 0 = 0:

128	64	32	16	8	4	2	1
0	0	0	1	1	0	0	1
0	0	0	1	1	1	0	1
0	0	1	1	0	1	1	0
		1	1			1	

At this point, it's a good idea to convert all 3 numbers to denary and check that we didn't make any mistakes.

Subtraction

Subtracting numbers is almost exactly the same, with a single difference: We use *Two's Complement* and simply add a negative number:

-128	64	32	16	8	4	2	1
0	0	0	0	0	0	1	1
1	1	1	1	1	1	1	1

Here we're representing the sum 3 – 1. The bottom number (11111111) is the number -1 in two's complement. To work out the sum, we simply add the numbers together just like before:

-128	64	32	16	8	4	2	1
0	0	0	0	0	0	1	1
1	1	1	1	1	1	1	1
0	0	0	0	0	0	1	0
1	1	1	1	1	1	1	

In each column we carry a one over to the next. When we get to the '-128' column, we run out of places to keep our carry bit, so we just throw it away. This is called ***Overflow.*** In this case, our use of two's complement means that overflow has worked to our advantage. When a two's complement number overflows, it becomes positive, which in the case of 3 – 1, is what we want (2 is positive!).

Overflow

Overflow is where the answer to a calculation exceeds the maximum number that can be represented.

Consider the 8-bit numbers we have been working with. Unsigned, we can represent numbers between 0 and 255. If we performed a calculation that resulted in a number higher than 255, the result would *overflow* and we could have a problem:

128	64	32	16	8	4	2	1
1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	

The last carry bit is lost and the result of our sum (255 + 1) ends up with the result of zero.

When a number overflows, this can cause a problem with any more calculations we do with that number later on, because we're probably using an invalid value.

Hexadecimal

Hexadecimal is a short-hand notation for representing each of the possible 4-bit-binary combinations:

Bin	Hex	Den	Bin	Hex	Den	Bin	Hex	Den	Bin	Hex	Den
0000	0	0	0100	4	4	1000	8	8	1100	C	12
0001	1	1	0101	5	5	1001	9	9	1101	D	13
0010	2	2	0110	6	6	1010	A	10	1110	E	14
0011	3	3	0111	7	7	1011	B	11	1111	F	15

5. Data Compression

Data compression is a class of computing algorithm designed to make files require less storage space.
This can be to reduce storage requirements or to make files faster to download.
There are 2 types of compression algorithm:

Lossless – information is rearranged to take up less space but nothing is lost
Lossy – where some information is discarded in order to make the file smaller

Common Compression Formats

Different types of data require different algorithms to compress them sensibly – consider the difference between and single static image and a video file with many independent frames.

Data Type	Compression 1	Compression 2	Compression 3
Image	JPEG	GIF	PNG
	Lossy – variable ratio for different quality levels. Typical: 1:20	Lossy ~1/3rd original size, but can sometimes be bigger	Lossy, best for line art/drawings.
Music Track	FLAC	MP3	M4A
	Lossless ~ 1/10	Lossy ~ 1/50	Lossy ~ 1/50
Film	MPEG-4	FLV	MPEG-2
	Lossy	Depends – Lossy for video, but lossless for drawing/animation	Lossy
Plain Text	Zip	Rar	gzip
	Lossless	Lossless	Lossless

Run Length Encoding

Run Length Encoding (RLE) is the simplest type of *lossless* compression. It works best with data that has lots of repeated sections, such as a line drawing where most of the colour is the same.

A file encoded with a RLE algorithm consists of a stream of pairs of bytes. The first byte in each pair is called the ***run count***. The second byte is called the ***run value***. When *uncompressing* such a file, the result looks at each byte pair and produces a new file, repeating each run value the number of times specified by run count.

Example

The following is the first 4 byte pairs of a text file that has been encoded using RLE:

05AF 0255 01FF 07A9

When uncompressed, this would produce a file with the contents (expressed in Hexadecimal):

AF AF AF AF AF 55 55 FF A9 A9 A9 A9 A9 A9 A9

Or

1010 1111 1010 1111 1010 1111 1010 1111 1010 1111 0101 0101 0101 0101 1111
1111 1010 1001 1010 1001 1010 1001 1010 1001 1010 1001 1010 1001 1010 1001

As stored on disk as binary.

6. Encryption

Encryption is one half of a process described by ***Cryptographic Algorithms***.
Encryption algorithms are used to transform plain text into something unreadable (*cypher text)*.
Cypher text can then be decrypted, restoring it to its original plain text. Decryption forms the second half of a cryptographic algorithm.

Most security algorithms involve the use of encryption, which allows two parties to communicate without their messages being intercepted by third parties.

Types of encryption

There are many different types of cryptographic algorithm but most belong to one of two categories:

Symmetric

Asymmetric

Symmetric Encryption

Symmetric algorithms are generally faster for computers to execute and require a single key. Both parties are required to know the key in order to encrypt or decrypt messages. This makes it less secure than Asymmetric algorithms since the key must be transmitted between parties somehow.

Asymmetric Encryption

Asymmetric algorithms require 2 keys – a public key, used to encrypt a message to be sent and a private key used to decrypt it on receipt.

Caesar Cipher

The Caesar Cipher is an example of a very simple symmetric encryption algorithm.

A Caesar Cipher replaces each letter in a message with a letter further along in the alphabet using a *shift key*.

For example, with a shift key of 3, a table could be drawn up as follows:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

“My simple message” could then be encrypted as: “Pb vlpsoh phvvdjh” by reading the table top to bottom. The cypher text can then be reversed by reading the table bottom to top.

7. Databases

Businesses and organisations use ***relational databases*** to gather, collate and analyse data

Key Features

Databases use a series of **tables** to store data. A table is a **two-dimensional** representation of data, stored in *rows* and *columns.*

Each table in a database must have a **unique name** so that the *database management system (DBMS)* can find the right table.

Each row stored in a table must have a **unique key** (called a **Primary Key**) in order to make the database usable.

Flat File databases are databases that consist of a single table.

A **Relational Database** consists of a number of tables **linked together.** In order to be useful, a relational database must have 3 basic components:

- A data store
- A method of **creating and retrieving** data
- A method of ensuring data is *logically consistent*

Tables – a table in a relational database is a two-dimensional structure used to store related information.

Records – In databases, records are a complete, single set of information. A record is all the data about one item in a database. A record in a database can also be referred to as a **row**.

Columns – A Column (sometimes called a **field**) within a table contains all the information of a single type, such as all the employees' names, or all of the phone numbers. Columns are usually formatted to accept only **certain types of data** such as integers, booleans or strings.

Queries – A database query is essentially a *question* that you ask the database about the data that it holds.

Primary Keys – Every relational database should contain *one or more columns* that are assigned as a primary key. A primary key **must be unique** for each record contained within a table.

Relationships – Database relationship work by comparing data in key fields, where a field appears in two or more different tables it defines a relationship between the records in those tables where the field has the *same value*. In almost all cases, a primary key from one table is used as a unique identifier for records in another table. This is called a **foreign key**.

Index – Databases use indexes to help organise data in a table and to make it easier (and faster) to find specific records as part of a query. It can be thought to operate in a similar manner to an index in a book.

SQL

SQL, or Structured Query Language is a type of code that is written to perform operations on data in a database.

Each statement begins with a **clause** that defines the type of operation being defined:

INSERT, SELECT, UPDATE and DELETE

Consider a typical query:

```
SELECT forename, surname, phone_number
FROM members
WHERE date_of_birth > '2000-12-31'
ORDER BY surname, forename;
```

The SELECT clause tells us that this query is trying to find some information in a database. Immediately after the SELECT clause is a list of columns that we want to display as the result of our query. The FROM clause tells the database that we want to look in the members table for the information. The WHERE clause acts as a filter, only returning the records that match our criteria: anyone who was born *after* 31/12/2000. Finally, the ORDER BY clause asks the database to return the results sorted alphabetically by the surname, and then by forenames for the results with the same surname.

INSERT – this clause adds new record(s) to a database table:

```
INSERT INTO members (user_id, forename, surname, phone_number, date_of_birth)
VALUES (‘SMDA775’, ‘David’, ‘Smith’, ‘01622 775294’, ‘2001-01-06’);
```

SELECT – As we have seen, select finds data stored in a database

UPDATE - this clause changes information already stored in a database:

```
UPDATE members
SET phone_number=’07557 224558’
WHERE user_id=’SMDA775’;
```

DELETE – removes information stored in a database:

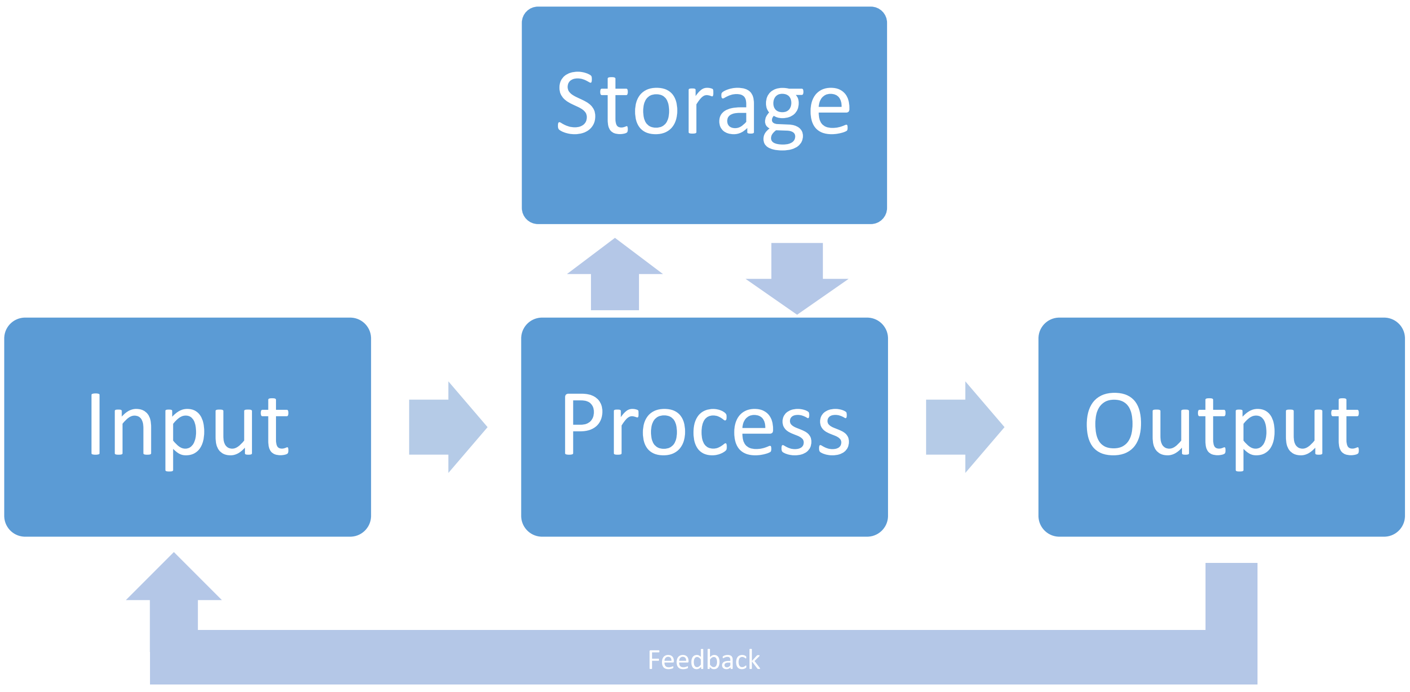
```
DELETE FROM members
WHERE user_id=’SMDA775’;
```

8. Machines & Computational Models

A computer can be described using a simple model, called the ***input-process-output*** model.

Input: This stage represents the flow of data into the process from outside the system
Process: This stage uses data provided by the input to perform calculations and make decisions
Output: The output stage is where the resulting data flows back out of the system

Additionally, most computer systems include a **Storage** stage that allows data to be retained when the computer is switched off.



9. Hardware

Hardware is a name given to a collection of *physical things* that when put together in a certain way form a **system**.
The hardware is the **machine**.
Hardware refers to parts or components of a system that can be *physically touched*.

Key Components

Motherboard – The central printed circuit board that holds all of the crucial hardware components and provides connections (**busses**) between them to enable them to communicate.

CPU – Central Processing Unit. The CPU is responsible for carrying out program instructions and consists of a number of components all packaged together:

Control Unit – *Fetches*, *decodes* & *executes **instructions***

Arithmetic Logic Unit – Performs arithmetic and logical operations on data

Registers – fast, on-chip memory inside the CPU that can either be dedicated to a specific purpose (such as the *program counter*) or general purpose (such as *data registers*)

Logic Gates - Control the flow of information between different components

Memory – There are a number of types of memory in use in a computer system:

ROM – Read Only Memory is special memory used to hold essential programs for interfacing with hardware components. It cannot be overwritten and serves to provide a computer with its initial start up program (BIOS – Basic Input/Output System). This allows a CPU to operate connected storage devices in order to load the Operating System when the computer is first powers on.

RAM - Random Access Memory is used to hold programs and data that are currently in use. RAM is **volatile** storage, meaning that its contents is lost when power to the computer is removed.

Cache – Very high speed memory, usually contained within the CPU package. Used to store information that the processor will need repeatedly (**temporal locality**) or to store information that the CPU is likely to need soon (**spatial locality**). Cache memory is very fast but usually quite small (also part of what makes it fast).

***Virtual Memory** – Some operating systems provide a virtual memory feature that uses space on the hard drive to simulate RAM, it is much slower than a computer's RAM but storing active programs and data in virtual ram can still be faster than launching an application and loading files and documents from hard disk.*

Persistent Storage – Devices such as *hard disk drives* or *solid state drives* are used to store programs and data persistently. Information stored on such devices persist when power is removed, however, it is slower to operate on data stored in this way.

Peripherals

Any device that connects to the computer is referred to as a peripheral. These devices perform either **Input, Output or Storage** functions (or sometimes combinations of all 3).

Output Devices

Monitor Printer Speakers Plotters Braille Embossers

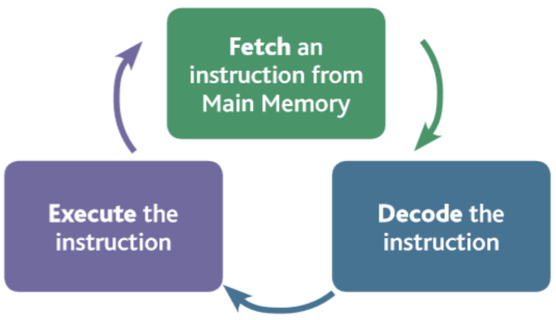
Input Devices

Keyboard Mouse Scanner Camera Microphone Touchscreen Joystick

Storage Devices

External HDD USB Memory Stick SD Card CD/DVD Drive

Fetch-decode-execute cycle



The fetch-decode-execute cycle describes how a Control Unit in a CPU operates:

- First, the program fetches the instruction from the control unit
- Next, the processor copies this information into main memory
- The information is then sent to the memory **buffer register** via a **data bus**
- The instruction is copied into the the **current instruction register** to be decoded and executed
- Finally, the decoder **interprets** the instruction

Microcontrollers

Microcontrollers can be programmed to control devices. They are essentially a small computer, having a CPU, a small amount of RAM and some **input** and **output** devices.

Microcontrollers use **actuators** and **sensors** to function.

Actuator – used to move or control output. Converts energy into motion.

Sensor – measures a physical quantity, such as temperature, distance, sound and converts it into a signal.

10. Logic

The Boolean data type has only two values: **true** or **false**. These values are used to control the flow of the execution of programs. Boolean values are found by comparing other data values. The results of these comparisons may be combined in *Boolean expressions*.

Logic gates

Many electronic circuits have to make decisions. They look at two or more inputs and use these to determine the outputs from the circuit. The process of doing this uses electronic logic, which is based on digital switches called **gates**.

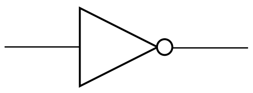
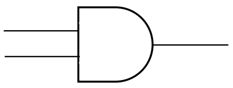
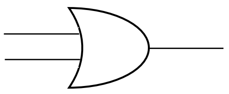
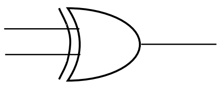
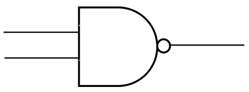
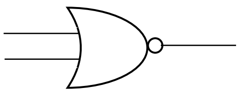
Logic gates allow an electronic system to make a decision based on a number of its inputs. Each input and output of the gates must be one of two states:

true or 1 or 'on'

false or 0 or 'off'

A single digital signal can be *either* on *or* off - for example, a light with one switch can be on or off. However, if there is more than one signal, there are more than two possible states. For example, if two signals are present there are four possible combinations: on/on, on/off, *off*/on and off/off.

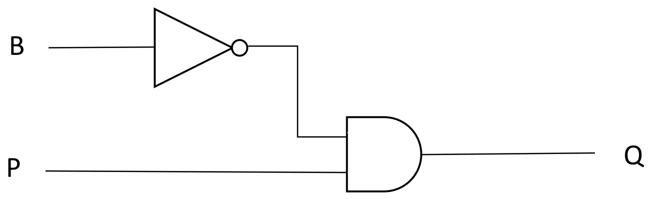
Common Logic Gates

Gate	Symbol	Function
NOT		Accepts 1 input. Changes a true value to false and a false value to true
AND		Accepts 2 inputs. Outputs true if and only if both inputs are true
OR		Accepts 2 inputs. Outputs true if any input is true
XOR		Accepts 2 inputs. Outputs true if one or other input is true (but not if both are true)
NAND		Accepts 2 inputs. Outputs false only if both inputs true, outputs true otherwise.
NOR		Accepts 2 inputs. Outputs true only if both inputs are false.

Truth Tables

Consider a Boolean expression Q = (NOT B) AND P

This expression can be represented as an expression as above, or as a circuit diagram with logic gates:



Or as a truth table:

B	P	NOT B	Q
0	0	1	0
0	1	1	1
1	0	0	0
1	1	0	0

Inputs B and P could be a switch or sensors or even outputs from other logic circuits. Consider a robot with a Brake Switch B and Power Switch B. The motors should only turn (Q=1) if the brakes are off and the power is on (as seen in the second row of the table).

11. Software

Software is the term used for the programs or code that a computer executes on its processor.

Operating System

The operating system (often just called the OS) is a collection of specialised software that manages a computer's hardware resources and provides services to other computer programs.

Some examples of operating systems are:

Microsoft Windows, Apple Macintosh OS X, Linus, iOS & Android

The primary tasks of the operating system are:

Manage computer resources - The operating system manages the CPU, allocation of memory, access to disk drives and connected devices such as printers or scanners.

Interact with the user – The OS provides a way for the user to interact with the computer, allowing them to launch applications and perform tasks. This interaction can be a simple as allowing the user to type commands and view text output, or as complicated as providing graphical elements that the user can click on and move around the screen.

Run applications – The operating system allows programs to run on the computer, managing access to resources such as memory or disk storage.

Application Software

Application software is computer software that causes a computer to perform useful tasks beyond the running of the computer itself.

Some examples of application software are:

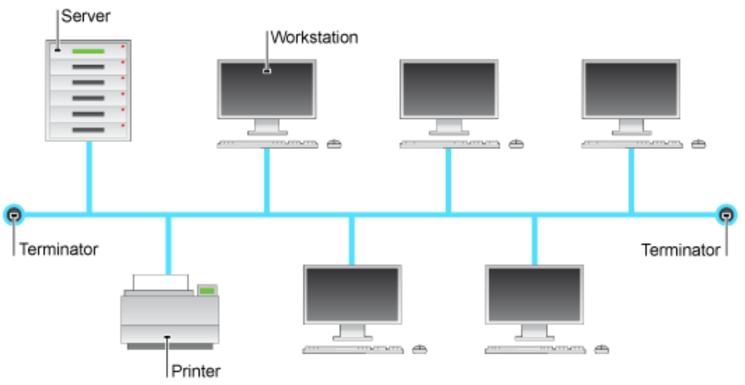
Word Processors	3D Graphics Software	Animation Software	Digital Audio Editors	Graphic Art Editors
Video Editors	Music Sequencers	Presentation Software	Media Content Creators	Games

12. Networks & the Internet

A **computer network** is two or more computers connected together through a communication media.
The purpose of connecting computers together in a network is to exchange information and data or to share the resources of other computers.

Network Topologies

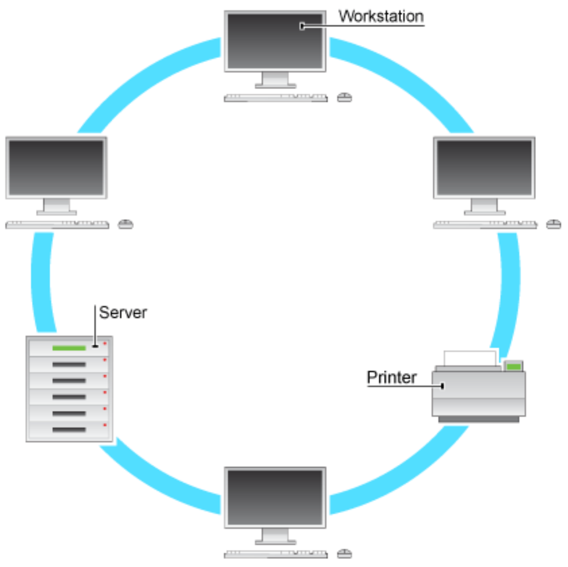
Bus Network



A **bus network** uses a common backbone to connect all of the devices. A *single cable* functions at the backbone and acts as a shared communication medium that each device connects to via a Network Interface Card (NIC).

Bus networks are **easy and cheap** to install as they only need a small quantity of cable. Because all connected devices must share the networking medium, and only one device at a time can actually use it, bus networks are only suitable for a **small number of devices**.

Ring Network

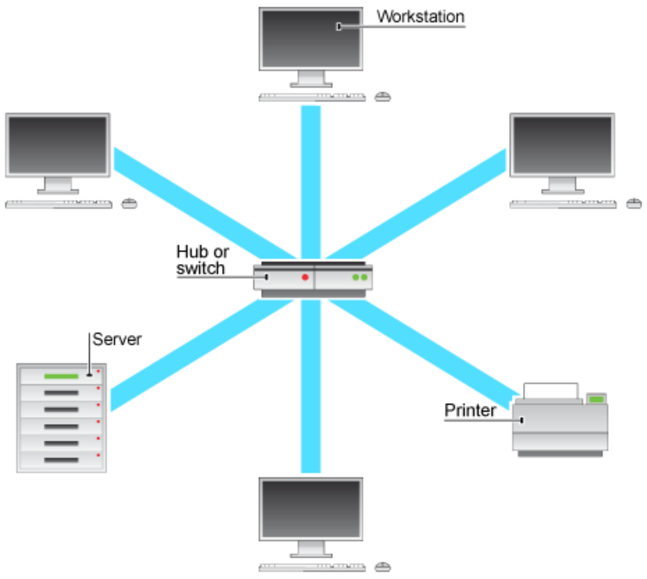


In a ring network, each device is connected to two other devices. All messages are passed in the same direction, either **clockwise** or **anticlockwise**.

Each machine examines incoming data and passes it onto the next machine in the ring if necessary. In the diagram above, in order for the marked workstation to communicate with the server, it must pass its message onto the machine on the right, which passes it to the printer, which passes it to the workstation at the bottom and finally onto the server. The server replies by sending a message to the workstation connected to the left which finally passes it on to the original workstation.

Ring networks are useful in keeping the cost of a network down because a central server is not required to manage communication. This topology can be fragile however, as a break in a single cable can stop the whole network from working.

Star Network



Most modern networks are star networks. A star network requires a central 'hub' device to manage communication between the connected devices. This device could be a **Hub, Switch or Router.** Star networks are robust, meaning devices can be connected and disconnected easily without affecting the operation of the rest of the network. However, they do contain a single point of failure in the hub device, as if this fails, the whole network will fail. Star networks require a great deal of cabling, with each connected device requiring a dedicated connection back to the central hub and so can be expensive to install.

Network Protocols

A protocol is a set of rules that govern how devices will communicate with each other.

The internet uses the TCP/IP model to communicate. The *Internet Protocol* part defines addresses for each machine that consist of 4 bytes, separated by periods: **127.0.0.1** TCP is in charge of the reliable delivery of information, dealing with error checking.

Checksum

Checksum is a simple **error-detection scheme**. Each message is transmitted with a numerical value that represents the number of set bits (1's) in the message. The receiving device can compare this number to the number of set bits it receives and determine whether or not there has been an error in the transmission of the message.

Client-server Model

The client-server model is the structure of a computer network in which many clients request and receive service from a centralised server. Servers wait for requests to arrive from clients and then respond to them.

Handshaking

When two computers connect in a network, they first use a **handshake**. Handshakes determine details such as which protocols will be used for communication or the speed at which communication will happen. All network connections, such as a request from a web browser to a web server, have their own handshaking protocols which must be completed before the requested action can be completed.

13. Emerging Trends

Computer systems are becoming ever more integral to our everyday lives
Microprocessors control many of the products we use each day
We have come to depend on computer systems a great deal
Malfunctions of computer systems can be *catastrophic*, both for organisations and for people.

Issues arising from computer use

File Sharing – Illegal sharing of music, films and software using the internet has become more and more of a problem as internet speeds have increased.

Digital Divide – There are still many people in the world who do not have access to the internet. Rural communities, low-income families, people with disabilities and impoverished nations around the world do not have the same advantages as more privileged people and communities.

Plagiarism – The ease of publishing and accessing information online has led to an explosion of people copying other people's work.

Hacking – Accessing of data without authority or permission has become more of a problem.

Cloud Computing

Advantages

Lower device costs – The user does not need an expensive computer to run cloud computing web-based applications as all the processing is done "in the cloud", instead of on the user's device.

Reduced software costs – Instead of buying expensive software, many tasks can be completed online for much lower cost, or even often for free.

Instant Updates – Cloud-based software is always up-to-date, without having to worry about installing updates or patches.

Unlimited storage capacity – Cloud computing offers virtually limitless storage.

Automatic Backup – Documents and data are stored in many different machines in the cloud, meaning that if your device fails, you can still access your data.

Disadvantages

Requires reliable internet connection – Cloud computing can be problematic without an internet connection.

Can be slower – because the processing is done elsewhere, it can sometimes be slower. This can be compounded by the need to transfer data across the internet, compared to using a locally installed application.

Limited features – Often web-based applications have fewer features than their locally installed counterparts. However, this is changing every day with web-based applications becoming more and more powerful.

Security – Since your data is stored "in the cloud", there is potential for others to access it. You are at the mercy of another organisation's security policies which may not always have your interests at heart.

Quantum Computing

Quantum computing is based upon *quantum physics.*

It takes advantage of the properties of atoms or nuclei that allow them to work together. Rather than storing data as 1's and 0's like conventional computers do, quantum computers use **qubits**, or ***quantum bits***. Qubits can be a 1 or a 0 or even both at the same time!

Advantages

Can be **much more powerful** than conventional computers.

Can be **faster** than conventional computers.

Can be **smaller** than conventional computers.

Disadvantages

It is **hard to control** quantum particles.

Lots of heat is generated by quantum computing.

Quantum computing is **much more expensive** than traditional computing.

It is very **difficult to build** a quantum computer, compared to a conventional computer.

DNA Computing

DNA Computing does not use traditional silicon-based technologies.

Instead, DNA computing uses **DNA, biochemistry and molecular biology**. DNA Computing is also called **Bimolecular computing.**

Advantages

DNA computers are very **light in weight** compared to traditional computers

The amount of **power required is much less** for a DNA computer

In tasks such as complex modelling, DNA computers are **extremely fast** compared to conventional computers.

Disadvantages

Some **simple problems** actually **take longer** for DNA computers.

Errors are more common due to the complexity of DNA strands.

DNA Computers are **very expensive** to build.

It is very **difficult to build** a DNA computer.

Nanotechnology

Nanotechnology, or Nanotech is the manipulation of matter on an **atomic scale**. It is used to make **very small** macro-scale products.

Advantages

Nanotechnology is **much smaller** than traditional silicon components

Nanotechnology is **much faster** than traditional silicon.

Disadvantages

Nanotechnology is very **difficult to build**.

Nanotech is **very expensive,** and currently limited in what it can do.

Artificial Intelligence

Artificial Intelligence, or AI, is the branch of computing concerned with making computers and machines behave and act more like humans.

Presently, there are no computer capable of simulating human behaviour completely.

In the field of games-playing, the best computer programs are capable of beating humans at games like chess or go.

Artificial intelligence is making great progress in areas such as self-driving cars.