AQA Computer Science A Level Notes

4.1 - Fundamentals of Programming

Much of this section requires one to understand how to complete a number of tasks, such as iteration, which would be specific to the language of use, and are rather trivial.

ALGORITHM: *Set of rules or sequence of steps specifying how to solve a problem. The specification indicates that the steps indicated by the algorithm must terminate, but many disagree with this. An Algorithm generally contains an* ***INPUT*** *step, a* ***PROCESSING*** *step and an* ***OUTPUT*** *step.*

Data Types: *Different types of data are stored differently in the computer's memory; these are:*

- ***Integers:*** *Number written without a fractional component, is a member of* $\mathbb{Z}$
- ***Real (floats):*** *Number written with a fractional part.*
- ***Boolean:*** *Variable with two possible values, 'true' and 'false'*
- ***Character:*** *A single number, letter, etc.*
- ***String:*** *Array of characters*
- ***Date / Time:*** *Storage of the date and time*
- ***Arrays:*** *Collection of a number of similar variables – often in rows and columns in a table like structure*
- ***Records (Struct):*** *Collection of fields, often with a number of different variables.*

Program Constructs: *There are three basic programming constructs:* ***Sequence, Selection and Iteration.***

- ***Sequence*** *is one statement after the other.*
- ***Selection*** *statements are used to find which statement should be performed next, making use of conditional operators.*
- ***Iteration:*** *Two types of iteration, definite and indefinite iteration – with definite iteration involving a clear assignment at the beginning of the loop of how many loop cycles would occur for. Meanwhile the indefinite loop would often make use of conditions, (such as in a while loop), where the loop would occur while the condition is being met.*

Using Variables: *The process involves three steps,* ***declaration,*** *followed by* ***assignment*** *and then* ***use.***

Identifier Names: *These names are used for everything from the names of variables to the names of classes and functions. It is helpful to important to use useful identifier names as it makes it easier for the program to be easily readable by you or indeed by others.*

Integer vs Float Division: *Integer division returns the answer without the remainder, whereas float division will return the answer to the division in a decimal form. In order to get just the remainder, modular division can be used.*

Truncation: *Process by which the number of digits after the decimal point are limited.*

Boolean Operations: *Boolean operations are effectively passing variables through logic gates, such as the AND, OR, and NOT gate.*

Variables vs Constants: *Variables are identifiers which are given to a specific memory location where the value will change over the course of the program. Constants are instead values which will remain constant throughout the running of the program – in a long program, they offer peace-of-mind to a programmer, that they could not change this value.*

Exception Handling: *Exception handling is a mechanism by which the program will cater for any possible issues that occur through the running of the program, such as accessing non-existent variables, etc.*

Subroutines: *A subroutine is a block of code which performs a specific task within a program. There are two main types of subroutines, functions and procedures. There are a number of built in functions, for example in C#, functions include Console.ReadLine(). Some functions return a result, while some are void, therefore not returning a result to the main program. A function is called, always assigning the value of the return value to a variable. A procedure is called by writing the identifier without setting the value of the return value to a variable. In the same way that subroutines can return values to the main program, the main program can be passed as parameters to the subroutine.*

Structured Programming: *When a program is short and simple, there is no need to break it up into subroutines. However, when the program is long, it is often useful to break the program into a number of subtasks. This offers a number of advantages.*

- *Easier to read for the programmer and someone else reading the program.*
- *Easier to edit the program –* ***maintainability****.*
- *Easier to write an algorithm – where the problem can be divided into a number of subtasks.*
- *Reduced complexity.*

Hierarchy Charts can be showed to show the structure of the program, including how it flows, including subroutines and subroutines inside these subroutines. However, the program does not show the detailed program structures in every module, therefore it doesn't have the required complexity. These can be shown in a structure chart.

- 4.1.1.15 - Functions of a Call Stack
  - **Majorly used to store information about the active subroutines while a program is running**
    - HOLDING RETURN ADDRESSES
      - Call stack keeps track of the address of the instruction that control should return to when the subroutine ends.
      - Therefore, must be a stack.
    - HOLDING PARAMETERS
      - Parameters required for a subroutine may be held on the call stack
    - LOCAL VARIABLES

            - Much more efficient to store local variables on the call stack than using the heap.
    - **The Stack Frame**
        - Call stack consists of stack frames - each frame consists of a call to a subroutine.
            - Including return addresses, holding parameters, local variables.

- 4.1.1.16 - Recursive Algorithms
    - Recursive if it calls itself
        - Has a base case
        - Must call base case after a finite number of calls.
    - Uses the call stack - with each recursive step taking up a stack frame.
    - **Recursive tree traversal**
        - In-order
            - In-order left node
            - Visit root node
            - In-order right node
        - Pre-order
            - Post-order left
            - Visit root
            - Post-order right

4.1.2.3 - Object Oriented Programming

In object oriented programming, the world is seen as a collection of objects, which could refer to absolutely anything.

OOP is composed of a number of interacting objects, each of which is responsible for its own data and the operations on that data.

Code creates these objects and allows the objects to communicate with each other by sending messages and receiving answers.

Why Object Oriented Programming

- Forces designers to go through an extensive planning phase.
- Encapsulation: the source code for an object can be written, tested and maintained independently of the code for other objects.
- Once an object is created, knowledge of how its methods are implemented in order for someone to know how to use it.
- Reusability
- Software maintenance.

Objects

Each object will have its own attributes and a state

Each object has behaviours - functions which can be performed by the object.

Classes

Class is a blueprint or template for an object - defines the attributes and methods of objects in that class.

As a general rule, instance variables or attributes are declared private and most methods public, so that other classes may use methods belonging to another class but may see or change their attributes.
This is a principle of information hiding.
Constructors are items used to create objects in the class.
Getters and setters also exist.

Instantiation
Creating an instance of a class is known as instantiation - creating a reference type variable.
Using getters and setters, the items in the class can be changed.

Encapsulation
An object encapsulates both its state and its behaviours and methods.
Related to the concept of encapsulation is **information hiding** , where details of how a class can be used can be ignored when utilising this class.

Inheritance
Classes can inherit data and behaviour from a parent class in the same way that children can inherit characteristics from their parents.
A child class in OOP is a subclass and parent class is a superclass.
Can therefore create a inheritance diagram.
Inheritance should be used using the 'is a' rule…

Association, Aggregation and Composition
Association is a 'has a' relationship

Association Aggregation or simply aggregation is a more special type of association - can occur when a class is a collection or container of other classes but the contained classes do not have a strong lifecycle dependency on the container
(player does not cease to exist if the team is disbanded)

Composition is a stronger form of aggregation - if the container is destroyed, then every instance of the contained class is destroyed - i.e. if a hotel is destroyed, then every room is destroyed.

Polymorphism
Polymorphism refers to the ability to process objects differently depending on their class - using overriding things which exist.

COMPOSITION OVER INHERITANCE
Less rigid relationship between two objects. Therefore, generally people prefer to go for composition over inheritance.
Things are often not 'inheriting' objects - instead they are utilising parts of them.

PROGRAM TO AN INTERFACE

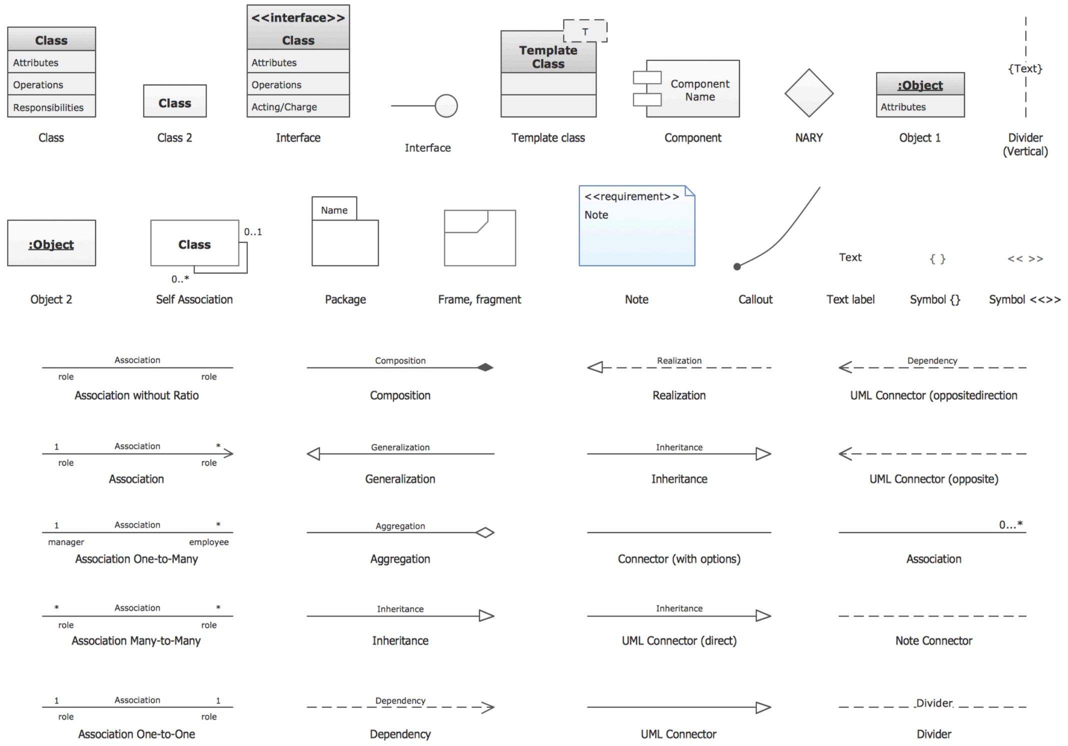
An interface is a collection of abstract methods that a group of unrelated classes may be implemented. Although the methods are specified in the interface, they will only be implemented by a class that implements the interface.

This is generally a good idea as it can take immediately from the design and therefore can ensure that someone definitely implements all the required features of the class.

ENCAPSULATE WHAT VARIES

Used in order to reduce maintenance and testing effort.
If the concept is encapsulated in a single module - and needs to be changed - then only that module has to be changed while the rest of the system can be left entirely alone.
During design, if things are going to change, then they should be made using encapsulation rather than inheritance. Therefore, if they are changed, there are no knock-on effects in other classes which inherit from them.

CLASS DIAGRAMS KEY



4.2 – Fundamentals of Data Structures

- Static vs Dynamic Data Structures
  - Dynamic Data Structure refers to a collection of data in memory that has the ability to grow or shrink in size
    - Does this with the aid of a **heap**
    - Queue must often require being given a maximum size to prevent memory overflow.

- Static Data Structure
    - Static array is fixed in size.
    - However, takes up this size forever.

- Abstract Data Types
  - *Queues, stacks, trees and graphs*
  - Logical description of how the data is stored, and the operations that can be performed on it, and not the way this is done.
  - (Example of **data abstraction**, therefore creating an **encapsulation** around the data. Therefore **information hiding**).
- **QUEUE**
  - First In First Out (FIFO)
  - New elements added to the end of the queue.
  - Elements retrieved from the front of the queue.
  - Uses
    - Outputs waiting to be printed.
    - Characters typed on a keyboard - keyboard buffer.
  - OPERATIONS
    - enQueue
    - deQueue
    - isEmpty
    - isFull
  - Structure of a queue
    - Pointer to the front and the rear of the queue.
  - **Implementing a queue**
    - 1st way: As items leave a queue, all the other items move up one space so that the front of the queue is always the front element of the structure.
    - 2nd way: Pointer to front and rear of the queue - integer value of the size of the array as well as a variable giving the max number of items.
  - **Circular Queue**
    - Gets rid of the problem of the second way - that eventually pointer to last item of data structure - by it eventually pointing to the first event
  - **Priority Queue**
    - Acts like a queue in that the order in which things are dequeued from the front of the queue.
    - However, the logical order of the items within the queue is determined by their priority with the highest priority at the front of the queue.
    - Implemented by checking the priority of each item in the queue, starting at the rear and moving it along one item at the time, until an item with the same or lower priority.
- **STACK**
  - Last In, First Out (LIFO)
  - Items added to the top, and items removed from the tom.
  - Can be implemented using either a **static** or **dynamic** data structure.
    - Static can be implemented using two variables: one for length of stack and one for location for the top of stack.

- Operations:
    - Push
        - Add item to top of stack
    - Pop
        - Remove item from top of stack and returns it
    - Peek
        - Returns item from top of stack (doesn't remove it)
    - IsEmpty
    - IsFull
- **Overflow** and **Underflow**
    - Stack will always have a maximum size, because memory cannot grow indefinitely.
    - If implemented using an array, full stack can be tested for by examining the value of the stack pointer.
    - Pushing another item would cause an overflow.
    - Poping an item would cause an underflow if the stack pointer is -1

- **LIST**
    - Abstract data type - effectively an array with an undefined number of terms.
    - It is possible to use a static array to store the items of a list.
    - Functions
        - Insert
        - Remove
    - **Ordered List**
        - Some languages, like Python, have a built in dynamic list structure which uses a linked list - therefore hides all the associated function.
        - As nodes are added, new memory locations can be dynamically pulled from the heap
            - **HEAP:** A pool of memory locations which can be allocated or deallocated as required.
        - The pointers in different items can then be updated.

- **HASH TABLES AND DICTIONARIES**
    - **Hashing**
        - Large collections of data need to be accessible very quickly without looking through all the records.
        - This can be done by holding an **index** of the physical address on the file where the data is held.
        - Hashing algorithm applied on the key field of each item.
        - **COMMON ALGORITHM**
            - Divide the key by the number of available addresses and take the remainder as the address.
        - **Collisions**
            - When two keys generate the same hash
        - **Synonyms**
            - Two things which generate the same hash

        - **Hash Table**
            - Collection of items stored so they can be accessed quickly.
            - **Collisions are stored in the next available free slot.**
            - **Searching for an item**
                - Apply the hashing algorithm
                - Examine the resulting cell in the list
                - If there - return the item
                - If empty - item is not in the table
                - If used - keep moving forward until it is found, or a cell is empty.
        - **Hashing Algorithm**
            - FOLDING METHOD
                - Divides an item into equal parts and adds the parts to give the hash value.
            - STRINGS
                - Adding up the ASCII values for each letter
        - **Collision Resolution**
            - Finding an empty slot when a collision has occurred - simply looks for the next empty cell
                - Plus 3 rehash is another possibility
    - **DICTIONARIES**
        - Effectively a hash table
        - Consists of associated pairs of items, where each pair consists of a key and a value.
        - When the user supplies the key, the associated value is returned.
        - OPERATIONS ON DICTIONARIES
            - Create new empty dictionary
            - Add a new key:value pair
            - Delete a key:value
            - Amend a key:value
            - Return a value associated with key k
            - Return true or false depending on whether key is in the dictionary
            - Return length of the dictionary
        - Pairs not held in any particular sequence.
- **GRAPHS**
    - **Set of vertices or nodes connected by edges or arcs.**
    - Edges may be one-way or two way.
    - In an undirected graph, all edges are bidirectional. If all edges in a graph are all one-way, the graph is set to be a directed graph.
    - The edges may be weighted to show there is a cost to go from one vertex to another.
    - **Implementing a graph**
        - ADJANCENCY MATRIX
            - Value stored about each connection between nodes.
            - Adjacency matrix is symmetric if the graph is undirected.
        - Adjacency List
            - More space-efficient in the case of a sparsely connected graph.

        - List of all the connections between the node.
        - List can be implemented as a list of dictionaries, with the key in each dictionary being the node and the value.
  - **Applications of Graphs**
    - Computer networks
    - Roads between towns
    - States in a FSM
    - **PageRank Algorithm**
        - Makes a graph out of the internet and defines the connections as a graph - weighted based on the importance of the page, and the number of clicks.
- **TREES**
  - **Connected, undirected graph with no cycles.**
  - If a tree starts with a single node, it is a rooted tree (with the root node)
  - DEFINITIONS
    - Node: Nodes contain the tree data
    - Edge: An edge contains two nodes. Every node except the root is connected by exactly one edge from another node in the level above it.
    - Root: Only node with no incoming edges.
    - Child: The set of nodes that have incoming edges from the same node.
    - Parent: A node is a parent of all the nodes it connects to with outgoing edges.
    - Subtree: The set of nodes and edges comprised of a parent all descendants of the parent - may also be a leaf.
    - Leaf node: Node with no children.
  - **BINARY TREE**
    - Rooted tree in which each node has a maximum of two children
    - Binary search tree holds items in such a way that the tree can be searched quickly and easily for a particular item.
        - **IMPLEMENTING A BINARY SEARCH TREE**
            - Each node consists of:
                - Data Item
                - Pointer to the left item
                - Pointer to the right item
            - Alternatively could be held in a 2D list, or three separate lists of array, one for each of the pointers and one for the data items.
- **VECTORS**
  - Can be represented as:
    - List of numbers
    - Function
    - Way or representing a point in space
  - Often implemented as a list or a dictionary.
  - Adding and subtracted vectors - adding elements together.
  - Multiplying vectors by scalars - simply multiplying by a value.
  - **DOT PRODUCT OF TWO VECTORS**
    - One form of multiplication - scalar product - combining each element.

        - A.B = |A||B|cosß
        - Can be used to find the parity bit
    - **Galois Field of Two Elements**
        - **GF2(+) = XOR**
        - **GF2(*)=AND**

4.3 – Fundamentals of Algorithms

- **4.3.2 - Tree traversal**
    - **Pre-order traversal**
        - Go around the tree anticlockwise - when you are to the left of a node, it should be outputted.
        - **COPYING A TREE**
    - **In-order traversal**
        - Go around the tree and as you pass underneath a node, output the data.
        - **BINARY SEARCH TREE**
    - **Post-order traversal**
        - As you pass to the right of a node, output the data.
        - **4.3.3 - INFIX TO REVERSE POLISH NOTATION**
- **4.3.3 - Reverse Polish Notation**
    - (Postfix)
    - Eliminates the needs for brackets in subexpressions
    - Produces expressions in a form suitable for evaluation using a stack
    - Used in interpreters
    - Precedence rules
        - =
        - (
        - + -)
        - * /
        - ^
        - Unary minus
    - **Infix =** (6 * 7) + 4
    - **Postfix =** 67*4+
    - Binary Expression Tree
        - In order traversal will give in infix form
        - Post order traversal will give in reverse polish form.
- **4.3.4 - Searching and Sorting**
    - LINEAR SEARCH
        - Go through all the items in the list and check if something there - clearly O(n)
    - BINARY SEARCH
        - **If items are sorted**
        - Split the lists and then search through each halves in turn.
        - O(log n)
    - **4.3.5 -** SORTING ALGORITHMS
        - MergeSort - O(nlog n)
        - BubbleSort - O(n^2)
    - **Space Complexity -** Amount of memory that an algorithm requires.

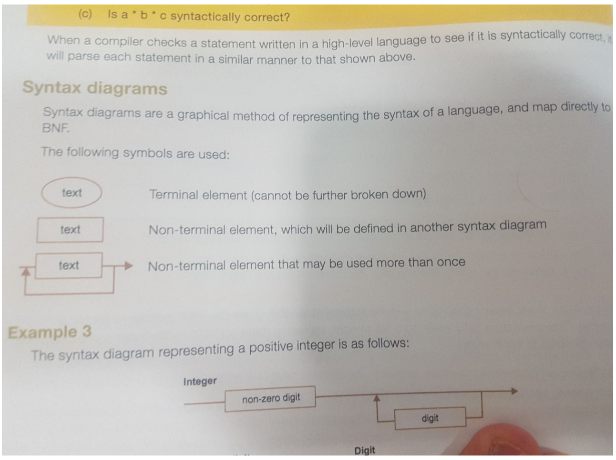
- **4.3.1 - Graph Traversal Algorithms**
    - **Depth First:**
        - Uses a stack - goes deep going as deep as it can before visiting all other nodes.
        - APPLICATIONS
            - Going through a maze
    - **Breadth First:**
        - Uses a queue - goes to all nodes before then checking that it can go any deeper, adding them to a queue of points to visit.
        - APPLICATIONS
            - Djikstra is kinda effectively a breadth first search
            - All dem BIO questions blud.
            - Web Crawlers

- **4.3.6 - Optimisation Algorithms**
    - **Djikstra**
        - 4
        - Goes through a graph and has an edge weight for each connection
        - Structured Programming - stop using GOTOs.
        - Uses a priority queue rather than a FIFO queue.

4.4 – Theories of Computation

- **4.4.2.1 - FSMs (with or without inputs - Mealy Machines)**
    - FSM with no output is a Finite State Automaton
    - FSM is an abstract representation or model of computation used in designing computer systems and logic circuits.
    - FSM with output = Mealy Machine
    - State Transition Diagram
        - States are
            - Start State
            - End State /
            - Accept State
    - **Mealy Machine**
        - Machine has outputs that are determined both by its current state and the current input.
        - For each input, only one transition is possible.
        - **Applications**
            - Cipher Machines (modelling them)
- **4.4.2.2 - Maths for regex**
    - Set is an unordered collections of values or symbols in which each value or symbol occurs at most once.
    - Can be defined in three ways
        - **Common Sets**
            - Empty set
            - Natural Numbers (N)
            - Integers (Z)
            - Rational Numbers (Q)

        - Real Numbers (R)
    - **Finite and Infinite Sets**
        - Finite = sets whose elements can be counted off by natural numbers until a particular number
        - Infinite set
            - Can be countable and uncountable
            - N is countable - can be counted off against the set of natural numbers
            - R is uncountable
        - Cardinality of set = n of elements in the set
        - **Countable Set is** a set which can be counted against a subset of the natural numbers.
    - **Defining a set by set comprehension**
        - B = {n^2 | n E N ^ n < 5}
        - | = such that
        - ^ = and
    - **Defining a set using the compact representation**
        - A = {0^n 1^n | n}
  - Set arithmetic
    - Cartesian product of two sets
        - Set of all ordered pairs where one member is a member of the first set, and the other is a member of the other set.
    - Union
        - Contains everything in both sets.
    - Intersection
        - Contains everything which exists in both sets.
    - Difference
        - Removes the members from the element to be subtracted from.
- **4.4.2.3 - Regular Expressions**
  - **| = alternatives**
  - **? = zero or one**
  - *** = zero or more**
  - **+ = one or more**
  - A regular language can also be defined as any language that a finite state machine will accept.
  - Regular expressions and FSMs are equivalent ways of defining a regular language
    - Regular languages are defined as anything which can be represented by a regular expression.
    - Also a regular language is defined as anything which can be represented by an FSM.
- **4.4.3 - Backus-Naur Form**
  - It is a met-language, which can be used to check syntax - allowing things to be described more succintly using BNF.
  - Made of terms of the form:
    - LHS ::= RHS (where ::= is 'is defined by')
        - ::= is a metasymbol
    - <point> is a metacomponent and a syntactic variable

        - Eg. <digit> ::= 0|1|2|3|4…
    - Can be composed using other BNF functions
    - Also often uses recursion
    - SYNTAX DIAGRAMS
        - 

(c) Is a * b * c syntactically correct?

When a compiler checks a statement written in a high-level language to see if it is syntactically correct, it will parse each statement in a similar manner to that shown above.

Syntax diagrams

Syntax diagrams are a graphical method of representing the syntax of a language, and map directly to BNF.

The following symbols are used:

text — Terminal element (cannot be further broken down)

text — Non-terminal element, which will be defined in another syntax diagram

text — Non-terminal element that may be used more than once

Example 3

The syntax diagram representing a positive integer is as follows:

Integer

non-zero digit

digit

Digit

-
- **4.4.4 - CLASSIFICATION OF ALGORITHMS (time complexity and space complexity)**
    - **4.4.4.2 - O Notation**
        - **Time Complexity:** The definition of how much time an algorithm takes to solve a problem
        - Functions
            - Linear: f(x) = ax + b
            - Polynomial: ..
            - Logarithmic: …
            - Factorial: …
        - Orders
            - O(1) - Constant time
            - O(n) - linear time
            - O(n^x) - polynomial time
            - O(2^n) - exponential time
            - O(log n) - Logarithmic time
            - O(n!) - exponential time
    - **4.4.4.4 - Limits of Computation**
        - **Insoluble practical problems**
            - Problems can be theoretically soluble but will take millions of year to solve
            - Therefore practically insoluble
            - Eg. Passwords
        - **Travelling Salesman Problem**
            - For a simple method, requires a brute-force method, therefore testing every combination of the route that would visit every node.
            - Therefore, insoluble for large number of data.
            - **Computationally Difficult - requires heuristics to be easily doable.**
        - 4.4.4.5 - Tractable vs intractable
            - NP = Intractable

        - P = Tractable
    - Heuristics
        - Solution which has a high probability of being correct - therefore things like A* instead of Djikstra
        - Other thins like that are usable for Travelling Salesman Problem
        - Eg. A*, virus scanners use heuristics
    - 4.4.4.6 - Computable vs Non-Computable
        - Things which can't be solved algorithmically are non-computable.
        - Turing proved some problems are simply non-computable.
    - **4.4.4.7 - THE HALTING PROBLEM**
        - Problem of, whether a given input, a program will finish running or continue for ever.
        - Turing proved that no machine can solve this for all possible inputs.
        - **Shows there are some problems which cannot be solved by computer.**
- **4.4.5 - Turing Machines**
    - A turing machine can be views as a computer with a single fixed program with
        - A finite set of states in a state transition diagram
        - A finite alphabet of symbols
        - An infinite tape with marked off symbols
        - Sensing read-write head that can travel along the tape, one square at the time.
    - **States include:**
        - **Start state**
        - **Halting state (any state with no outgoing transitions)**
    - Example is:
        - Binary alphabet of 1, 0 or blank
        - And a machine can increment by one
    - **Transition Functions**
        - Transition rule can be represented by a function d (delta)
        - **D(Current State, Input Symbol) = (Next State, Output Symbol, Movement)**
    - **Universal Turing Machine**
        - A turing machine can theoretically represent any computation
        - The UTM can be used to compute any computable sequence - effectively by compiling something which is written at the beginning of the tape.
        - Anything which is computable can be computed by a turing machine. Otherwise, it is not computable.
        - Led to the idea of the stored program computer - program and data stored in memory

4.5 – Fundamentals of Data Representation

Number Systems

N: Natural Numbers – positive integer or nonnegative integer (some people choose to include zero while others do not)

Z: Integers – numbers which can be represented without a fractional or decimal part. The natural numbers are a subset of the integers.
Q: Rational Numbers – numbers which can be represented as a fraction. **Irrational numbers** such as p are numbers which cannot be represented as a fraction
R: Real Numbers – set including all natural, rational and irrational numbers.
Ordinal Numbers: Indicate the numerical position of objects – 1^{st}, 2^{nd} etc.

Number Bases

Different number bases show how many possible numbers could be represented by a single character.
Denary: Base 10 – makes use of '0' to '9'
Binary: Base 2 – makes use of '0' and '1'
Hexadecimal: Base 16 – makes use of '0' to '9' and 'A' to 'F'
Conversion

1. Denary to Binary
    a. Find the largest 2^n that would fit into the number and then subtract this from the original number. From here, look at all possible 2^n-x where n-x >= 0, putting a 0 if it would not fit in what is left of the number or a 1 if it would, then subtracting this from the number.
2. Binary to Denary
    a. Using the multipliers of each column, add up the values to find the original number, for example:

0	1	1	1
x8	x4	x2	x1
0	4	2	1
Therefore 0111 = 0 + 4 + 2 + 1 = **7**			

3. Binary to Hexadecimal
    a. Look at each nibble (four bits) of the number separately, converting each into hex, (largely through converting through denary) then combining each hex equivalent of each nibble to form one complete hexadecimal number.
4. Hexadecimal to Binary
    a. Convert each hexadecimal digit directly into a nibble, converting through denary, then recombining the value afterwards.
5. Denary to Hexadecimal
    a. Follow the same process as for denary to binary just for 16 as opposed to 2.
6. Hexadecimal to Denary
    a. Using the multipliers of each digit, add up the values to find the original number.

Hexadecimal Advantages:

7. Used to represent binary since it can represent a byte in only two digits rather than the eight required if binary were to be used.
8. Easy for technicians and computer user to remember hexadecimal digits.
9. Easy to convert to and from raw values on the computer, since binary can be easily converted to hex.

Binary

Terminology

- **Bit** = fundamental piece of information – a single 1 or 0
- **Byte** = set of eight bits
- **Nibble** = set of four bits
- **Signed Integers** are when the binary can represent both positive and negative numbers, where integers are **unsigned** when that can only represent nonnegative integers.

Unit Nomenclature

Name	Symbol	Power
Kibi	Ki	2^{10}
Mebi	Mi	2^{20}
Gibi	Gi	2^{30}
Tebi	Ti	2^{40}
Pebi	Pi	2^{50}
Exbi	Ei	2^{60}
Zebi	Zi	2^{70}
Yobi	Yi	2^{80}
Kilo	K	10^{3}
Mega	M	10^{6}
Giga	G	10^{9}
Terra	T	10^{12}
Peta	P	10^{15}
Exa	E	10^{18}
Zetta	Z	10^{21}
Yotta	Y	10^{24}

ASCII Table

The standard manner for representing the characters on a keyboard is called ASCII (American Standard Code for Information Interchange). ASCII originally only used 7 bits, but there was an increase into an 8 bit version to attempt to include more characters such as those in foreign languages. Subsequently, there was the production of Unicode (UTF-16 and UTF-32) which have all the possible characters that could be needed. However, this ensured that the first 128 characters were the same as those in ASCII to ensure there was complete compatibility with this.

Important Characters:

- **' ' = 32**
- **'0' = 48**
- **'A' = 65**
- **'a' = 97**

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Representing Negative Numbers

Two's Complement: Two's complement is a system where the most significant bit of the sum indicates whether the number is positive or negative, while the rest of the system determine the value of it. The range that can be given by a two's complement number is hence:

$$-(2^{n-1}) \text{ to } 2^{n-1} - 1 \text{ (where n is the number of bits used)}$$

Therefore, for an eight bit two's complement number, any number between -128 and 127 can be represented.

Converting Denary to Two's Complement

- Denary = -10
- Binary of 10 = 0000 1010
- **FLIP the bits =** 1111 0101
- Add one = 1111 0110

Converting Two's Complement to Denary

- Two's complement = 1110 0101
- **FLIP the bits =** 0001 1010
- **Add one =** 0001 1011
- Therefore, positive denary = 1 + 2 + 8 + 16 = 27
- Therefore, original was -27

OR

1	**1**	**1**	**0**	**0**	**1**	**0**	**1**
-128	64	32	16	8	4	2	1
-128	64	32			4		1
-128 + 64 + 32 + 4 + 1 = -27							

Representing Fractions

The binary system continues from 2^0 to 2^{-1} (0.5) etc, hence allowing us to represent decimals. However, it is important to note that the system is less useful than the decimal system as many decimals would require infinite bits to represent them, such as 0.2 and 0.3. To convert from decimal to fixed point binary is similar to convert from denary to binary, where one attempts to find the largest 2^n that would fit in the decimal before moving further to lower n, until the entire decimal has been represented in the binary using 2^n + 2^m...

Another way of representing fractions is making use of **floating point numbers**

Floating Point Numbers

Rounding Error: An important problem with both floating point and fixed point numbers is that they cannot necessarily be represented in binary. Some numbers, such as 0.1, simply cannot be represented as a string of binary numbers.

- Absolute error can be calculated as the difference between the number to be represented and actual representation.
- Relative error is the absolute error divided by the number, and may be expressed as a percentage.

Floating Point Binary Numbers:
Made up of three parts, the sign bit, the mantissa and the exponent. To convert the floating point number to a decimal, the number must be translated to the correct binary, by shifting it the requisite number of places, before converting it to denary.

Normalisation: Normalisation is the process of moving the binary point of a floating point number to provide the maximum level of precision for a given number of bits.

A positive number starts off with a 01, while negatives should start with a 10 - the mantissa of a negative number in normalised form always lies between -1/2 and -1.

FIXED POINT vs FLOATING POINT
Floating point allows for greater range of numbers using the same number of bits,
Fixed point depends entirely on the number of places before the binary point and the number of places after the binary point. However, it is much simpler and faster to process.

UNDERFLOW AND OVERFLOW
Underflow occurs when a number is too small to be represented in the allocated number of bits - eg. For example, a small number is divided by another small number, underflow will occur and the answer will be represented by 0.

Overflow occurs when the result of a calculation is too large to be stored in the number of bits allocated.

Error Checking

Parity Bits: This was developed especially when there was an extra bit, when 7 bit ASCII code was used. This extra bit could help to find if there was a single change. There are two types of parity bits, **even parity** and **odd parity.** This works by ensuring that the number of

1s in the byte are even or odd respectively by using the 8th bit as a 0 or 1 as required. This ensures that if there is a change in the rest of the byte (or indeed in the parity bit), the receiver on attempting to verify it, would find that there was a problem, and hence request retransmission. However, if there are two (or even number) of problems, this would not find that.

Majority Voting: Majority voting is a system which would require the same bit to be sent three times. On the receiving end, these are evaluated for each bit and it chooses the majority receipt (0 or 1) to be the correct one that had been sent, assuming that the majority of transmissions would be correct.

Checksums: Checksums work by adding up the value of all of the bytes separately and sending this value in addition to everything else. Therefore, it is likely (though not guaranteed) that if there are any changes in any of the bytes it would change the value of the checksum, hence registering a problem on the receiving end. This would then allow the receiving computer to request retransmission of the information.

Checkdigit: A checkdigit is similar to a checksum and includes an additional digit at the end of the string of the other numbers. An example of a use of checkdigits is in the ISBN (International Standard Book Number) and EAN (European Article Number) which is specific to each book. They make use of the modulo 10 system.

ISBN	**9**	**7**	**8**	**0**	**9**	**5**	**6**	**1**	**4**	**3**	**0**	**5**	**1**
Weight	1	3	1	3	1	3	1	3	1	3	1	3	
Multiplication	9	21	8	0	9	15	6	3	4	9	0	15	
Addition	Numbers are added together												99
Remainder	The answer is divided by 10 with the remainder being found (99 % 10)												9
Subtraction	The answer is subtracted from 10												1

Representing Graphics

Bitmapped Images

A bitmap (or raster) contains many picture elements or pixels that make up the whole image. A pixel is the smallest identifiable area of an image. Each pixel is attributed a binary value which represents a single colour.

The **Resolution** of the image can be expressed by width in pixels x height in pixels. It is sometimes also expressed as the number of pixels per inch, PPI, and refers to the density of the pixels. For printing, there is a similar measure of DPI (dots per inch) which refers to the printing quality of a printer.

The **Color Depth** of the image refers to the number of bits assigned to each pixels taken to describe the colour of the pixel. The more bits which are used, the greater accuracy of colour that will occur. The current standard is 3 bytes per pixel, allowing for one byte for red, one for green and one for blue. Sometimes an extra byte is used as an alpha channel to control transparency.

The **Metadata** specifies the properties of the image, including the colour depth, width and height of the image.

Vector Graphics

Vector graphics allows for the storage of an image in terms of geometric shapes or objects such as lines, curve, arcs and polygons. A vector file stores the necessary details about each shape to redraw the object when the file loads, including its position, and for example for a circle, radius, fill colour, line colour, line style and line width. These properties are stored in a drawing list which specifies how to redraw the image.

	Vector Graphics	**Bitmapped Graphics**
Scaling	Vector graphics scale perfectly, regardless of how large or small the image should be.	Bitmapped images do not scale well, simply increasing the sizes of the pixels leading to blurring at large scaling.
File Size	For simple images, with lots of geometric shapes, vector graphics are much more efficient, taking up much less space on disk. However, for complex shapes with lots of changing colours, and few geometric shapes, which could be abstracted from the image, the file size is very large, often with a specific list item (square) required for every single pixel, being far larger than the binary equivalent.	Bitmapped images are worse for simple shapes but more efficient for complex images, especially with images with continuous areas of changing colours.

Manipulation	Vector graphics are easy to manipulate in changing the position and properties of the objects however it is very hard to change small specifics, since the only changes that are allowed are to the properties of the object.	While large scale manipulation is harder, fine changes are much more simple, with the photo being able to manipulated to the pixel.

Representing Sound

Sample Resolution: The resolution of a sound is increased by a greater **audio bit depth** which describes the number of bits which would be used to describe the amplitude of the sound at a given point in time. Each point at the sample rate must be represented by a binary value and hence the more bits used to describe this amplitude means a greater accuracy compared to the original analog sound.

Sample Rate: The sampling rate is the rate at which one records the amplitude of the sound. The more often the sample is taken, the smoother the playback is, with fewer large changes in the amplitude between samples. However, the greater the sampling rate, the more space is required for the information about the song, hence leading to the song taking up more space on the hard disk.

Nyquist's Theorem: Henry Nyquist in 1928 found that in order to produce an accurate recording, the sample rate should be double the maximum frequency of the original signal, with the theory being proved by Claude Shannon. Therefore, music is generally sampled at 44,100 Hz (the max audible frequency for humans is around 20,000 Hz).

ADC:

- In order to convert analog music into digital music, a microphone must record samples at a regular interval (the sample rate). Each sample is then **quantised**, where the wave height is measured and given an integer value, which can be represented in binary, using a specific audio bit depth that is being used.
- To output a sound, the binary values for a sample point are translated back into analogue signals or voltage levels and sent to an amplifier.

MIDI

A Musical Instrument Digital Interface is a list of instructions that allow a device to synthesise a sound based on digital samples and samples of sounds created from different sources of instruments. Therefore, the file is generally much smaller (1,000 times) than conventional recordings). Hence, they are often used for phone ringtones.

Event Messages: Since MIDI files generate music used a timed sequence of instructions, they can also send event messaged to other instruments to synchronise tempo or control pitch and volume changes.

Metadata: MIDI files contain the information for a computer to recreate it accurately including the duration of the note, the instrument, volume and timbre.

Data Compression

Data compression techniques attempt to reduce the size of files on the disk. This is highly important previously when the cost of storage was very high, however, it is still quite important when the size of the file is important as the file must be transmitted over the internet. Compression can be **lossy** when the quality of the file after the compression is reduced and **lossless** when the lossless compression retains all information to ensure the original file can be replicated exactly.

Lossy Compression

This works by removing non-essential information. In images, this often works by reducing the colour depth ad in reducing the pixel size. In music files, frequencies which could not otherwise be heard are removed, while quiet sounds which are played at the same time as loud ones are also removed, meaning the final file is around 10% of the original size. When we talk over a telephone, the system makes use of lossy compression.

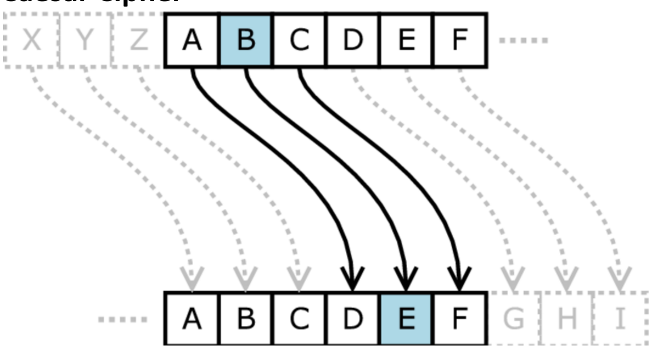
Lossless Compression

Lossless compression looks at patterns in the file to reduce the size of the data. Using the patterns and a set of instructions on how to use them, the compressed file can easily be returned to the original.
Run Length Encoding (RLE): Instead of recording every item in a repeating sequence, RLE find these sequences and records how many times and where these are used, therefore saving the space required to repeat these many times.
Dictionary-Based Compression: The system is useful when sending text files. Words are looked up into a common dictionary, finding their place in the dictionary. Therefore, this number can be transferred instead of the actual word. For example, ‘pelican’ which would otherwise require 7 bytes to be sent can be sent in two bytes.

Encryption

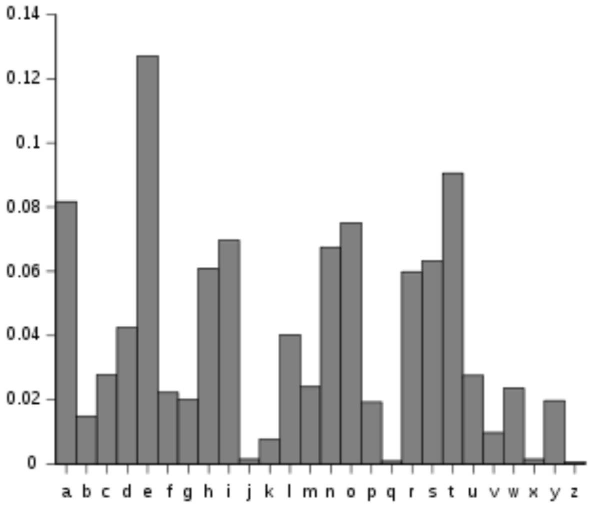
Encryption is the transformation of data from one form to another to prevent an unauthorised third party from being able to understand it.

Caesar Cipher



One of the earliest form of ciphers, as used famously by Julius Caesar, to ensure secrecy in messages was a shift cipher where each letter of the message was moved a certain number of characters up or down the alphabet, such that a shift encryption with a 3-shift of CAT is FDX. This is in essence a very simple cipher, and relied mostly upon the reader not spending the time to break the number of possibilities, clearly 26, the number of possible shifts that could occur, each one producing a different result.

Over time, a more sophisticated form of breaking shift ciphers and in fact a number of other ciphers, was developed by the Arab polymath, Al Kindi, in his paper, ‘A Manuscript on Deciphering Cryptographic Messages’, by investigating the text of the Qu’ran.



The process relies upon the fact that in certain languages the frequency of certain letters is characteristic. Thus, by investigating the frequency of letters in the encrypted text, one can easily see the code, as the graph would likely be shifted to the side, such that the peaks and troughs would be moved a certain amount. Therefore, one can easily find the likely shifts, reducing the number of attempts required to find the exact shift used.

Vernam Cipher
The Vernam Cipher is the only cipher to still be proven as unbreakable (by Claude Shannon). It relies on the use of the one-time pad.

One Tine Pad
According to many, Claude Shannon is the father of modern cryptography whose concepts are still used today, when designing new ciphers. Shannon’s work addressed the ‘problems of cryptography’. He largely separated the types of cryptography into two categories, one where the cipher was designed to protect against hackers who have infinite resources, now known as unconditional secrecy and a second, where the cipher protects against hackers with finite amount of resources. He also began to define the idea of ‘Perfect Secrecy’ the cipher text conveys ‘no information about the content of the plaintext’. The manner in which this can occur is using the One-Time Pad, where frequency analysis and other cryptanalytic techniques would have no effect upon the encrypted text. The One-Time Pad is similar to the Polyalphabetic and Substitution Cipher except that the code (key) is entirely random (as opposed to pseudorandom) and as long as the text to be encoded, such that there is no repetition of the entire code, which leaves the polyalphabetic code vulnerable. Shannon proved that perfect secrecy was only possible if this was true. It makes use of the bitwise XOR operator, which is carried out on the key and the plaintext for each bit.

Plaintext: M	Key	XOR
1	0	1
0	1	1
0	0	0
1	1	0
1	0	1
0	1	1
1	1	0

Though the One-Time Pad is by far the most sophisticated type of cipher that we have encountered so far, there are a few issues that it introduces. Firstly, the length of the code must be at least as long if not longer than the message that the people wish to encrypt, thus taking up a lot of length. This makes it very difficult to remember and use effectively. Additionally, the question becomes how to ensure that both parties have the same code, to decrypt and encrypt the message, since much of the communication is not in person and rely upon inherently insecure systems such as the internet. The mechanisms of fixing these issues were fundamentally fixed in RSA and the Diffie-Helman Key Exchange System.

4.6 – Fundamentals of Computer Systems

Hardware & Software

Hardware is the term used to describe the physical parts of a computer and its input, output and storage devices.

Software comprises of all of the programs that are written to make computers function. The software operates on the hardware and cannot exist without it. On the other hand, the hardware can exist without software.

Classification of Software

Software is broadly classified into **system software** and **application software. Application Software is software completing a task which would need to be done whether or not the person had a computer. On the other hand, System Software would be unrequired if the person does not have a computer.**

System Software

System software is the software needed to run the computer's hardware and application programs, including the operating system, utility programs, libraries and programming language translators.

Operating System: An operating system is a set of programs that lies between application software and the computer hardware. It serves many different functions including: resource management – managing all the computer hardware including the CPU, memory, disk drives, IO devices and provision of a user interface to enable users to perform all the tasks required.

Utility Programs: Utility software is a system software designed to optimise the performance of the computer or perform tasks such as backing up files, restoring corrupted files from backup, etc. Examples include a Disk Defragmenter is a program which will reorganise the hard disk so that files which have been split up into blocks and stored all over the disk. The defragmenter recombines the files into sequential blocks. A virus checker is another example of a utility program which checks your hard drive to ensure that there aren't any viruses.

Libraries: Library programs are ready-compiled which can be run when needed and which are grouped together in software libraries. Most compiled languages have their own libraries of pre-written functions which can be invoked in a defined manner from within the user's program.

Translators: Programming Language Translators fall into three categories: compilers, interpreters and assemblers.

Application Software

Application Software can be categorised as General Purpose, Special Purpose or Custom-Written.

General Purpose: Software such as word-processor, spreadsheet software which can be used for multiple purposes.

Special-Purpose: Software which completes a single specific task or set of tasks to serve a niche. Examples include payroll and accounts software. Software can be bought as **'off-the-shelf**' or **'bespoke**' software. While bespoke software is likely to more correctly and properly satisfy the requirements of the user, it is likely to be far more expensive, as well as requiring the user to well define the exact requirements of the software which can be challenging.

Role of an Operating System

An operating system is a program or set of programs that manages the operations of the computer for the user. It acts as a bridge between the user and the computer's hardware, since a user cannot communicate directly with hardware.

Functions of an OS

1. Creation of a Virtual Machine
    a. The OS disguises the complexities of managing and communicating with its hardware from the user via a simple interface. Through this interface, a user can complete all the actions they want to do.
    b. *A Virtual Machine is any instance when software is used to take on the function of the machine, including executing intermediate code or running an operating system within another to emulate different hardware.*
2. Memory Management
    a. The RAM of a computer is not large enough to store all running programs simultaneously so that the hard disk is used as virtual memory. Since the virtual memory is much slower to access than the RAM, it is important for the OS to ensure the programs that are being used are stored on the RAM, while the others are stored in the Virtual Memory.
3. Processor Scheduling
    a. When multiple tasks are being completed at the same time, the OS is responsible for allocating processor time to each one as they compete for the CPU. While one application is busy using the CPU for processing, the OS can queue up the next process required by another application to make the most efficient use of the processor.
    b. It also ensures that computers with lager processors can complete multiple tasks at the same time (by completing small parts of each larger task in sequence).
    c. The scheduler is the module responsible for ensuring that the processor time is used as efficiently as possible.
        i. **Objectives**
            1. Maximise throughput
            2. Ensure all users on a multi-user system receive their requirements
            3. Provide acceptable response time to all users
            4. Ensures that hardware resources are kept as busy as possible.

4. Backing Store Management
    a. This stores the areas of the disk where the files are stored, and where the free space is, containing an index of what is in every part of the disk.
5. Peripheral Management
    a. This manages all the Input / Output devices connected to the computer – including the keyboard, mouse, printer.
6. Interrupt Handling
    a. An interrupt is a signal from a peripheral or software program that causes the operating system to stop processing its current list of instructions and carry out the request as defined in the interrupt.
7. Provision of a User Interface
    a. This includes many forms of interface from a command line interface, such as in MS-DOS, to a Graphical User Interface such as in Microsoft Windows.

OS for Embedded Systems

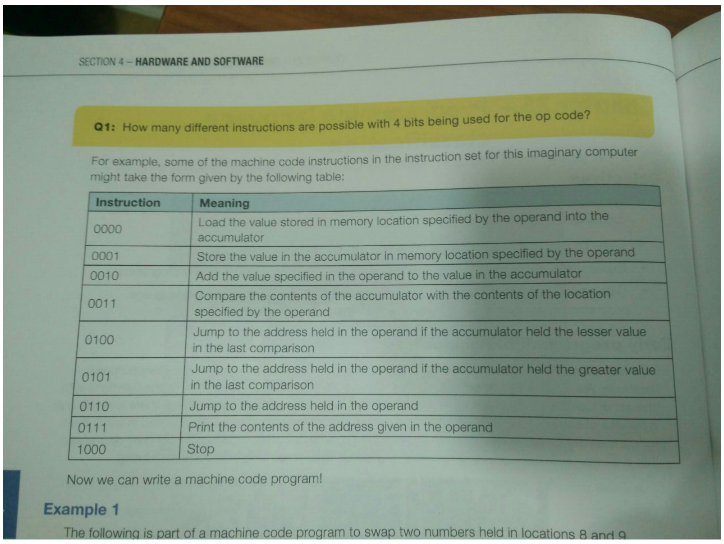
OSs are also required for all kinds of systems from a washing machine, with differences in everything from the hardware of the system to the requirements of the system.

Programming Language Classification

Development: The first computers were made in Bletchley Park during the Second World War, when they attempted to crack the Enigma Code. The earliest computers had a very limited memory (with each memory cell being made of a vacuum tube) of around 16 bits, a special memory location in which all calculations were carried out and a control unit that decoded instructions. In order to program a computer, one has to enter valid machine code (in binary).

Machine Code

In Machine Code, an instruction makes use of an operation code (opcode) in the first few bits and the operand in the rest of the memory cell, including the data to be operated on or the address where the data would be held.


SECTION 4 – **HARDWARE AND SOFTWARE**

Q1: How many different instructions are possible with 4 bits being used for the op code?

For example, some of the machine code instructions in the instruction set for this imaginary computer might take the form given by the following table:

Instruction	Meaning
0000	Load the value stored in memory location specified by the operand into the accumulator
0001	Store the value in the accumulator in memory location specified by the operand
0010	Add the value specified in the operand to the value in the accumulator
0011	Compare the contents of the accumulator with the contents of the location specified by the operand
0100	Jump to the address held in the operand if the accumulator held the lesser value in the last comparison
0101	Jump to the address held in the operand if the accumulator held the greater value in the last comparison
0110	Jump to the address held in the operand
0111	Print the contents of the address given in the operand
1000	Stop

Now we can write a machine code program!

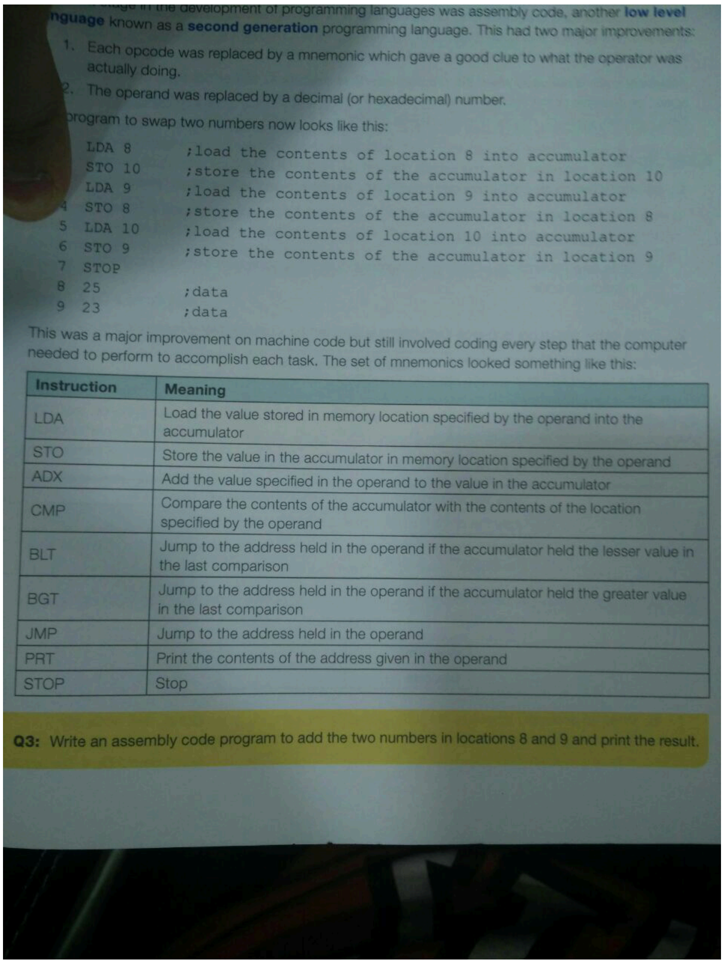
Example 1

The following is part of a machine code program to swap two numbers held in locations 8 and 9

The above table shows the possible opcodes of lots of different commands. Machine Code is a very low-level programming language, because the code reflects how the computer carries out the command – it is dependent on the specific architecture of the computer.

Assembly Code

From here, there was the development of a slightly higher (second generation) level programming language, where the opcode was replaced by a mnemonic which showed what the operator was doing. Additionally, the binary for the operand was replaced by a denary value.


...nguage in the development of programming languages was assembly code, another **low level** ...nguage known as a **second generation** programming language. This had two major improvements:

1. Each opcode was replaced by a mnemonic which gave a good clue to what the operator was actually doing.
2. The operand was replaced by a decimal (or hexadecimal) number.

...program to swap two numbers now looks like this:

```
  LDA 8     ;load the contents of location 8 into accumulator
  STO 10    ;store the contents of the accumulator in location 10
  LDA 9     ;load the contents of location 9 into accumulator
4 STO 8     ;store the contents of the accumulator in location 8
5 LDA 10    ;load the contents of location 10 into accumulator
6 STO 9     ;store the contents of the accumulator in location 9
7 STOP
8 25        ;data
9 23        ;data
```

This was a major improvement on machine code but still involved coding every step that the computer needed to perform to accomplish each task. The set of mnemonics looked something like this:

Instruction	Meaning
LDA	Load the value stored in memory location specified by the operand into the accumulator
STO	Store the value in the accumulator in memory location specified by the operand
ADX	Add the value specified in the operand to the value in the accumulator
CMP	Compare the contents of the accumulator with the contents of the location specified by the operand
BLT	Jump to the address held in the operand if the accumulator held the lesser value in the last comparison
BGT	Jump to the address held in the operand if the accumulator held the greater value in the last comparison
JMP	Jump to the address held in the operand
PRT	Print the contents of the address given in the operand
STOP	Stop

Q3: Write an assembly code program to add the two numbers in locations 8 and 9 and print the result.

High Level Programming Language

From here, there was the development of high level programming languages, where a single line of code no longer represented one instruction of the computer (one clock cycle). This allows the coder to easily code algorithms, using the **imperative high-level languages (FORTRAN = FORmula TRANsmission)**, without needing to worry about how the machine would handle the code. They are known as imperative languages, since every line is a single command to do something. Additionally, the language was identical for every computer, with different architectures, with different processors, which would require different machine and assembly codes. Though the system is often easier for programmers to write to use, it is sometimes less efficient than using the assembly code, programming the complete system. This means for some systems where the program needs to execute as fast as possible, such as in embedded systems, the program is better off being written in Assembly Code. Additionally, Assembly Code takes up much less space on a disk than higher level programming languages, also not requiring a translator.

Imperative Programming involves writing your program as a series of instructions that can actively modify memory, focusing on how, in the sense that you express the logic of your program based on how the computer would execute it.

Functional programming involves writing your program in terms of functions and other mathematical structures, focusing on what, attempting to specify the logic of your program, hence allowing for the rules of the way in which the program runs in order to solve the problem.

Program Translators

Assembler: Converts each assembly line (**source code = code made by the user**) piece of code into machine code (**object code = the code which is created post assembly or compilation which is executable and can be run directly by the computer**). This is specific to the computer that is being used, since the machine code instruction set would differ (though slightly) between different architectures.

Compiler: Compiler converts high-level code into machine code, therefore still requiring the compiler to be specific to the exact architecture and platform of the computer.

Interpreter: Interpreting is a different option to compilation, where the interpreter will look at each line in term, checking for syntactical errors, before translating it into machine code and running it. (In fact, the interpreter completes a cursory scan looking for flow issues, such as missing brackets etc before beginning the translation of the first line). This means that if there is a problem in the code, then the computer will run until that point before registering a problem. Therefore, it is particularly useful for testing and finding problems in long pieces of code.

Bytecode: In fact, the majority of interpreted languages do not convert to machine code, instead going to bytecode, which can be executed using a bytecode compiler.

	Compiler	**Interpreter**
Advantages	Object code can be saved on disk and run without needing to recompile the program. Once compiled, object code runs faster than interpreted code. For interpreted code things like loops are very inefficient, since they are converted to machine code every time they are encountered, instead of being translated once as would happen with compiled languages. Object code is more secure – hard to reverse engineer Can be distributed without a compiler present.	Faster, as the entire code does not have to be compiled before running the program. This is particularly important during the program development phases when there would be a number of minor errors, which would each require lengthy recompilations. It is easier to partially test and debug programs.

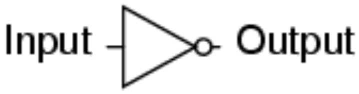
Logic Gates

Logic gates allow the user to complete conditionals, changing an outputted voltage (or signal) depending on the various inputs that are occurring. There are a number of different logic gates:

NOT gate

$$Output = \overline{Input}$$

NOT gate truth table

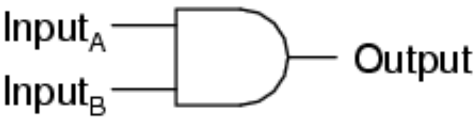


Input	Output
0	1
1	0

AND gate

$$Output = Input_A \cdot Input_B$$

2-input AND gate

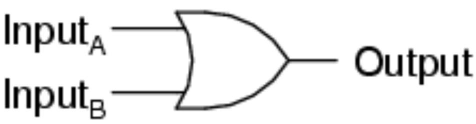


A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1

OR gate

$$Output = \ Input_A + \ Input_B$$

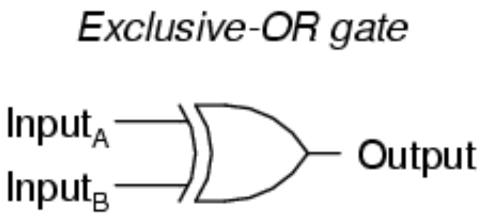
2-input OR gate



A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1

XOR gate

$$Output = Input_A \bigoplus Input_B = (Input_A \cdot \overline{Input_B}) + (\overline{Input_A} \cdot Input_B)$$

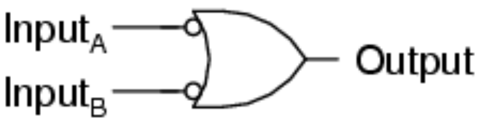


A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

NOR gate

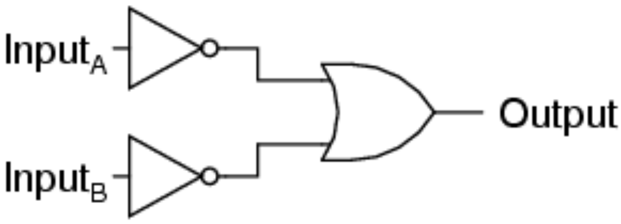
$$Output = \overline{Input_A + Input_B}$$

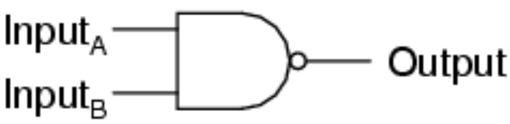
2-input Negative-OR gate



A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0

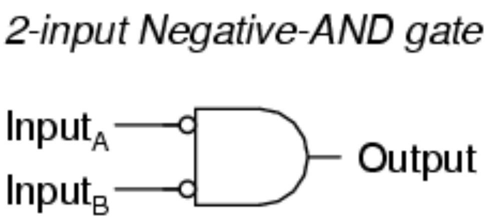
Equivalent gate circuits



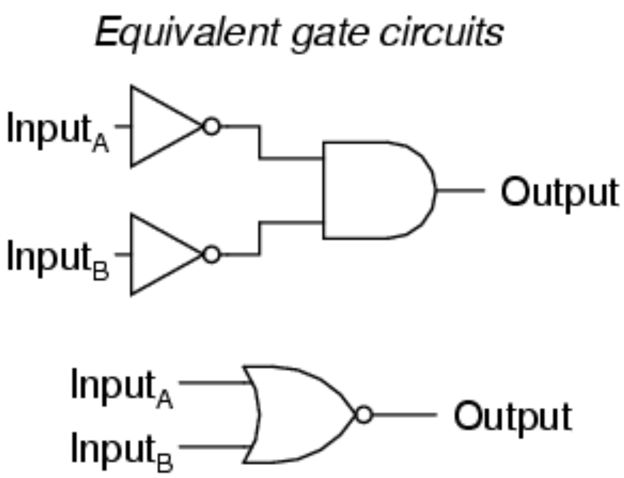


NAND gate

$$Output = \overline{Input_A \cdot Input_B}$$



A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0



Boolean Algebra

De Morgan's Laws

$$\bar{A} \cdot \bar{B} = \overline{A + B}$$

$$\bar{A} + \bar{B} = \overline{A \cdot B}$$

Other Laws

X · 0 = 0

X · 1 = 1

X · X = X

X · X' = 0

X + 0 = X

X + 1 = 1

X + X = X

X + X’ = 1

X’’ = X

X · Y = Y · X

X + Y = Y + X

X(Y·Z) = (X·Y)Z

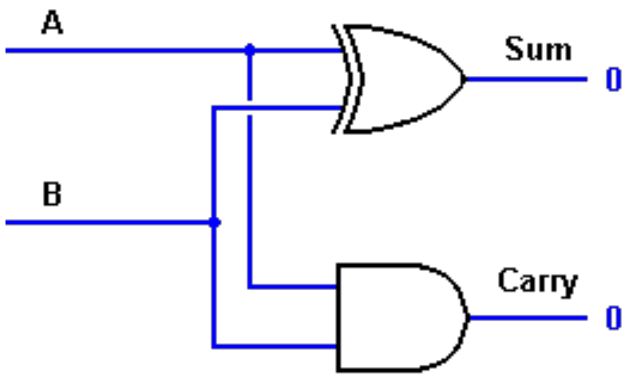
X + (Y + Z) = (X + Y) + Z

X(Y + Z) = X·Y + X·Z

(X+Y)(W+Z) = X·W + X·Z + Y·W + Y·Z

ADDERS and D-Type Flip Flops

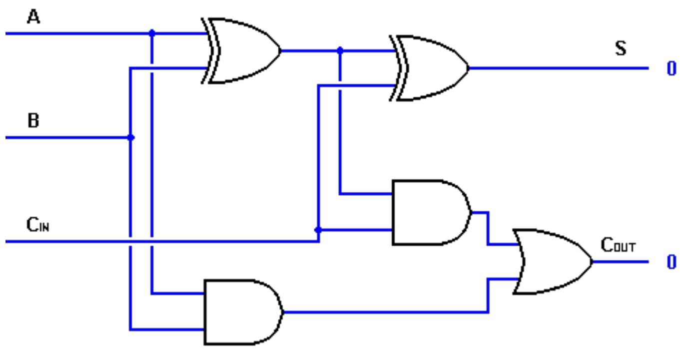
Half-Adder



S = A XOR B

C = A . B

Full Adder



S = A XOR B XOR C

C = (A.B)+(C.(A XOR B))

By connecting multiple full adders together, to complete a concatenated adder capable of adding a binary number of n bits.

D-type flip-flops

Elemental Sequential Logic Circuit that can **store one bit** and flip between states - has two inputs, a control input labelled D and a clock signal.

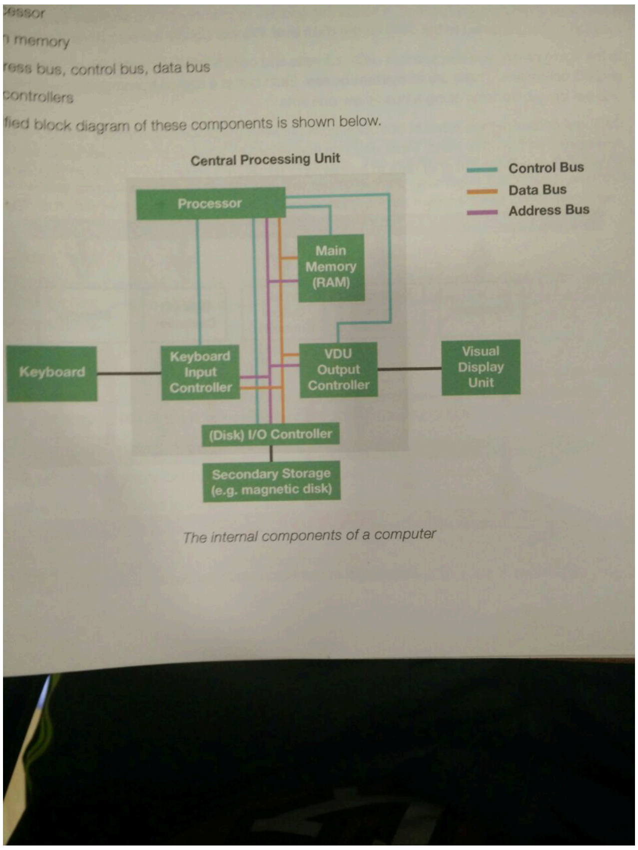
D type flip flop is a positive edge-triggered flip-flop, so it can only change input value from 1 to 0 or visa verca when the clock is at a a rising edge.

When the clock is not at a positive edge, the input value is held and does not change. HENCE, CAN BE USED AS A MEMORY CELL.

4.7 – Fundamentals of Computer Organisation and Architecture

Computers have **Internal components** – those within the computer shell and **external components** such as input / output and storage devices. The internal components include:

- Processor
- Main memory
- Address bus, control bus, data bus
- I/O controller



The Processor

The processor contains the Control Unit, the Arithmetic Logic Unit (ALU) and Registers. The Control Unit coordinates and controls all the operations carried out by the computer, completing the Fetch/(Decode)/Execute cycle.

Fetch: The next instruction to be completed is fetched from the memory

Decode: The signals to control the registers and the ALU are completed – decoding the instruction.

Execute: The instruction is completed.

ALU: The ALU completes all the arithmetic functions, such as addition, subtraction, multiplication and division, while it also performs the logic functions, such as comparisons between two numbers, and bitwise operators such as AND and OR.

Registers: Registers are memory cells which are very high speed because they are largely very small and they are located very close to the Control Unit.

General Purpose Registers: There are generally upto 16 general purpose registers in the CPU. All arithmetic, logical or shift operations take place in registers, which are very fast. Generally, especially in old computers, there is one dedicated register for the accumulator.

Dedicated Registers

Program Counter (PC): This holds the address of the next instruction to be executed. This may be the next instruction in a sequence, or if the instruction is a branch or jump instruction, the next address to jump to.

Current Instruction Register (CIR): Holds the current instruction being executed.

Memory Address Register (MAR): This holds the address of the memory location from which the data to be fetched or where data is to be written.

Memory Buffer Register (MBR): Used to temporarily store the data read from or written to memory – also sometimes known as the memory data register.

Status Register (SR): Contains bits which are set or cleared depending on the result of the instruction. It includes whether there was an overflow, whether the result was negative or zero. There is also bit which is used to store whether there is a carry.

Control Unit: The control unit controls and coordinates the activities of the CPU, directing the flow of data between the CPU and other devices. It accepts the next instruction, decodes it into several sequential steps such as fetching addresses and data from memory.

System Clock: The system clock generates a series of signals, switching between 0 and 1 around seven million times a second. Most fetch execute cycles take one clock cycle but a few take a number of clock cycles.

Buses

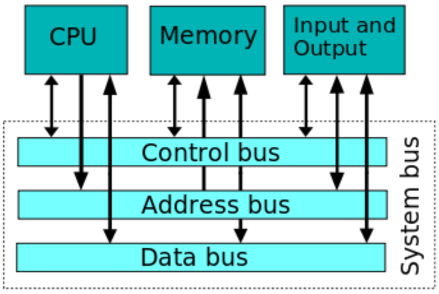
The processor is connected to the main memory by three different bus (**A bus is a set of parallel wires connecting two or more components of a computer)**. The address bus is used when the memory from a specific place on memory is required. This data is then returned making use of the data bus, while control signals, especially from the peripherals are sent along the Control Bus. While both the data bus and control bus are bidirectional with signals passing from the control unit to the memory and visa-versa, it is important to note that the address bus only carries data from the control unit to the memory.

Control Bus: Given that the data and address bus are shared by all components the system, the control lines must be provided to ensure there is no conflict, transmitting timing and status information between components. The various commands sent over the Control Bus includes:

- Memory Write; data bus information stored into addressed location
- Memory Read: data from addressed location into data bus
- I/O write: Data from data bus to I/O port
- I/O Read: Data from I/O port into data bus
- Bus Request: indication of a device request to use the data bus
- Bus Grant: indication that the Control Unit has granted access to the data bus

Data Bus: The data bus provides paths for moving data containing information from the memory as well as instructions to the control unit and between other registers. The width of the data bus us a key factor in determining overall system performance. Most computers are 32 or 64 bit, indicating that the have a data bus of this width.

Address Bus: The address bus contains the address of the word (collection of memory) that the control unit should access. As with the data bus, the wider the address bus, the greater amount of memory the control unit can address. For example, a system with a 32-bit address bus can address 2^{32} memory locations (with general word lengths is generally around 4GB). During I/O operations, the address bus is also used to address the I/O ports. However, for the address bus, generally this is unnecessary, and so it is more common for the address bus to make use of **multiplexing**, where the first half of the address is sent in one signal and then the rest of it in another signal.



I/O Controller: The IO Controller is a device which interfaces between an input or output device and processor. Each device has a separate controller which connects to the control bus. IO Controllers receive input and output requests from the processor and then can send signals to the device they control. They also manage the data flow to and from the device. The controller is made up of three main parts:

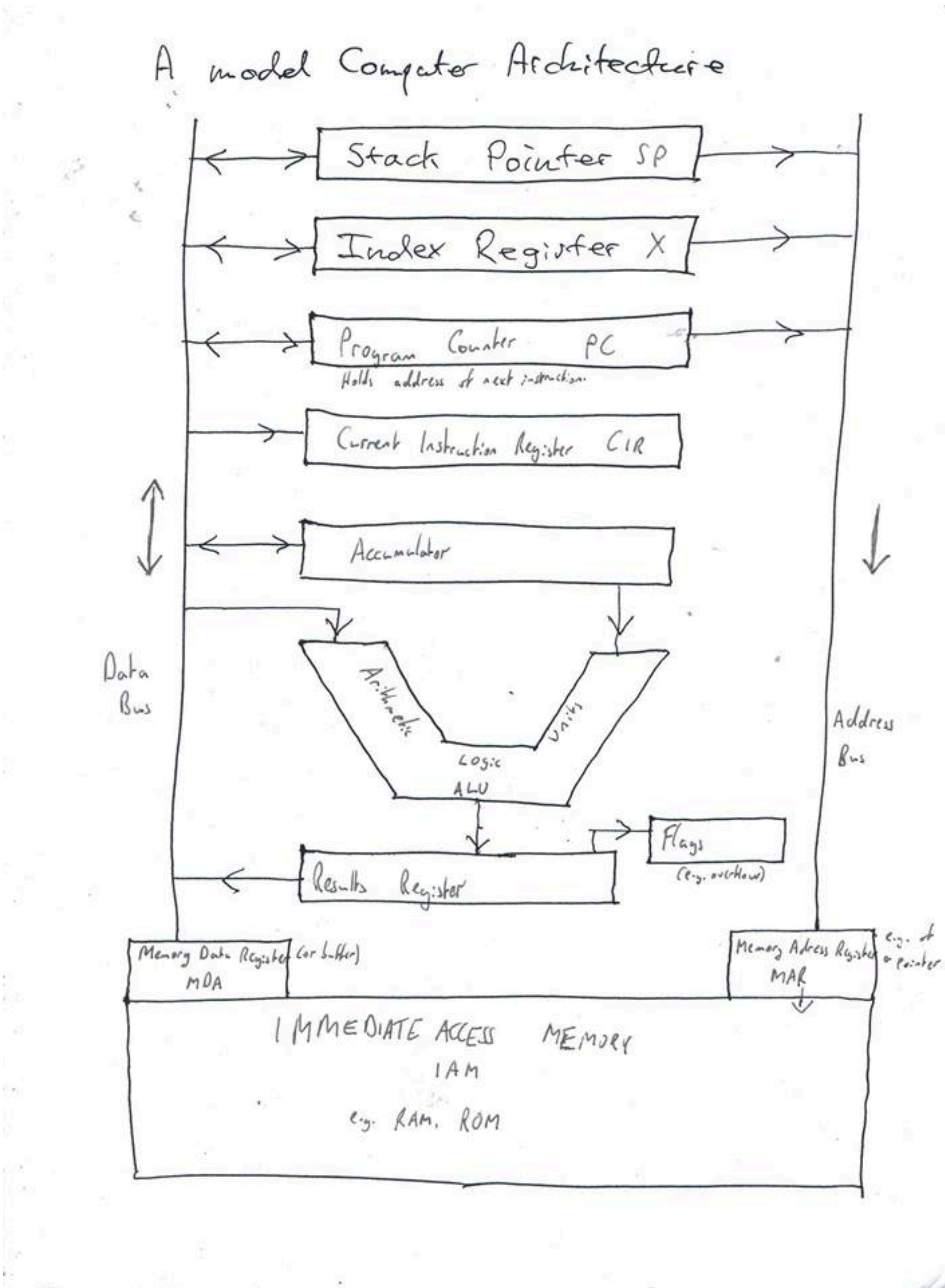
- An interface that allows connection of the controller to the system or IO bus
- A set of data, command and status registers.
- An interface **(standardised form of connection defining things such as signals, number connecting pins / sockets and voltage levels)** that allows connection of the controller to the device.

Von Neumann Machine: The von Neumann machine is a machine where there is one main memory store, where the instructions and the data on which the instructions were being carried out on were both stored. Almost all computers made today are built on this principle.

Harvard Architecture: The Harvard Architecture is a computer architecture where there are physically separate memories for instructions and data, allowing the data and instructions to be fetched in parallel instead of competing for the same bus. Additionally, the two memories can have different properties, allowing embedded systems where the instructions should never change to make use of faster and more stable Read only Memory (ROM). Additionally, the system would allow for different sized data buses (to main memory) and instruction buses to allow for times when the instructions are complex and long, and hence should have a much larger bus.

Fetch Execute Cycle

- **The address of the next instruction is copied from the PC to the MAR**
- **The instruction held at that address is copied to the MBR**
- **PC is incremented**
- **MBR is copied to CIR**
- **Instruction held in CIR is decoded**
- **Instruction is executed**



Factors effecting processor performance

- **Number of cores**
    a. Computers today make computers faster by tagging a number of identical processors together in the same integrated circuit (IC). This means there are dual-core, quad-core, octa-core etc processors which would allow a number of instructions simultaneously, since each of them would be able to complete a fetch-execute cycle at the same time. However, it is important to note that the performance is not necessarily proportional to the number of cores, as the ability of software to optimise the carrying out of multiple actions simultaneously is generally not perfect.
- **Amount of Cache Memory**

    a. The cache memory is the very small expensive memory which is very close to the CPU, so is generally used to store information which is required very often. The larger this memory, the more information that can be accessed very fast, therefore making the computer faster.
    b. There are three types of cache classed depending on where they are (closer to CPU is faster)
        i. L1 – very small (2-64KB) – very close to CPU
        ii. L2 – (256KB-2MB) – fairly close to CPU
        iii. L3 – relatively large (hundreds of MB) – quite close to RAM (so far from CPU)
- **Clock Speed**
    a. Most actions are completed at the ticking of the clock, though some actions will take longer than a single clock cycle.
    b. Therefore, a faster clock speed will mean that more actions can be carried out per second.
- **Word Length**
    a. The words size of a computer is the number of bits the CPU can process simultaneously, therefore how much they can process through registers such as the ALU. Typical word lengths are 32 or 64 bits.
- **Address Bus Length**
    a. The larger the address bus, the fewer signals would be required to address a single index on the memory, therefore allowing the processes to happen faster.
- **Data Bus Width**
    a. The larger the width of the data bus, the more data can be transmitted simultaneously, therefore reducing the time required to complete an instruction, therefore speeding up the computer.

The Processor Instruction Set

Each processor has its own instruction set, comprising of all the instructions which are supported by its hardware. This includes the following types of instructions:

- Data Transfer – LOAD, STORE
- Arithmetic Operations – ADD, SUBTRACT
- Comparison Operations - =, <, ≤
- Logical Operations – AND, OR, NOT, XOR
- Branching – conditional or unconditional
- Logical – shift bits right or left
- Halt

Format of a machine code instruction

Operation Code								Operand							
Basic Machine Operation						Addressing Mode									
0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	1

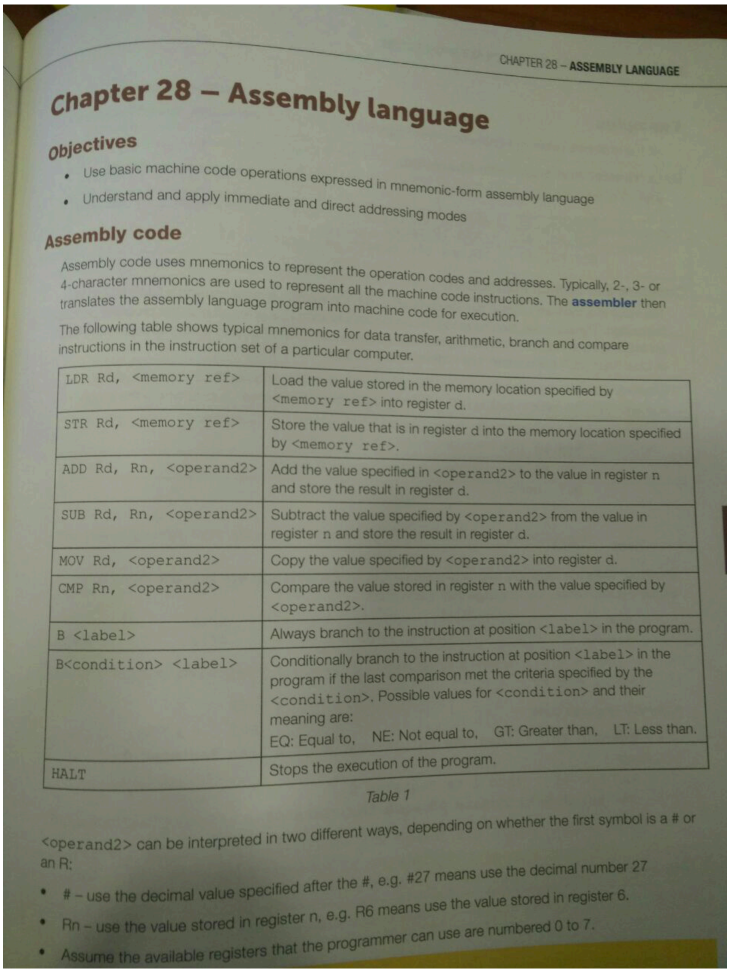
The number of bits allocated to the operation code (opcode) and the operand will vary according to the architecture and word size of the particular processor type. The above example shows an instruction held in one 16-bit word. In this particular machine, all operations are assumed to take place in a single register called the accumulator. In more complex architectures, each instruction may occupy up to 32 bits and allow for multiple operands – including load from memory address x into register y.

Addressing Modes: There are a number of addressing mdoes which are shown in the two binary digits which talk about the addressing mode. One of these modes **is immediate addressing**, where the operand is the actual numerical value to be operated on. In **direct addressing**, the operand holds the memory address of the value to be operated on. Other addressing modes include **Register Addressing** where the register contains the value. There is also indirect addressing, where the operand will contain the memory address of a memory address where the number to operate on is stored.

Assembly Code: Assembly code directly makes use of this system, representing the system with:

Opcode AddressingMode Operand

Though the system makes use of mnemonics for the opcode as opposed to directly using the binary. Additionally, the addressing modes make use of characters such as brackets, a '#' and a lack of anything to indicate the addressing mode being used, while the operand would be represented in denary (or sometimes hex) instead of binary.



Chapter 28 – Assembly language

Objectives

- Use basic machine code operations expressed in mnemonic-form assembly language
- Understand and apply immediate and direct addressing modes

Assembly code

Assembly code uses mnemonics to represent the operation codes and addresses. Typically, 2-, 3- or 4-character mnemonics are used to represent all the machine code instructions. The **assembler** then translates the assembly language program into machine code for execution.

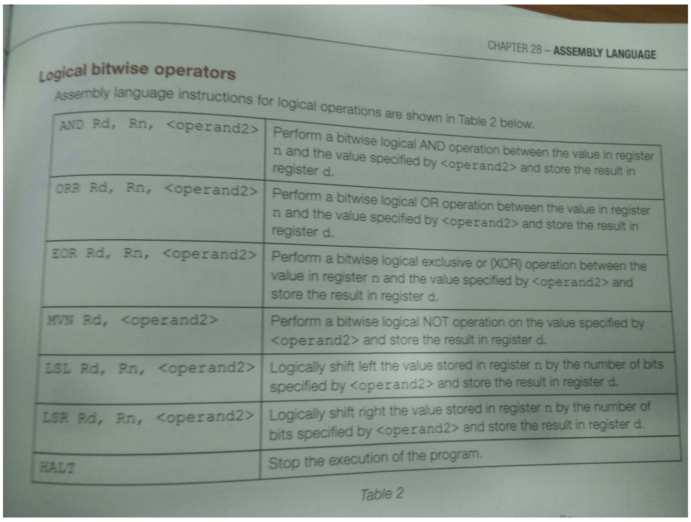
The following table shows typical mnemonics for data transfer, arithmetic, branch and compare instructions in the instruction set of a particular computer.

LDR Rd, <memory ref>	Load the value stored in the memory location specified by <memory ref> into register d.
STR Rd, <memory ref>	Store the value that is in register d into the memory location specified by <memory ref>.
ADD Rd, Rn, <operand2>	Add the value specified in <operand2> to the value in register n and store the result in register d.
SUB Rd, Rn, <operand2>	Subtract the value specified by <operand2> from the value in register n and store the result in register d.
MOV Rd, <operand2>	Copy the value specified by <operand2> into register d.
CMP Rn, <operand2>	Compare the value stored in register n with the value specified by <operand2>.
B <label>	Always branch to the instruction at position <label> in the program.
B<condition> <label>	Conditionally branch to the instruction at position <label> in the program if the last comparison met the criteria specified by the <condition>. Possible values for <condition> and their meaning are: EQ: Equal to, NE: Not equal to, GT: Greater than, LT: Less than.
HALT	Stops the execution of the program.

Table 1

<operand2> can be interpreted in two different ways, depending on whether the first symbol is a # or an R:

- # – use the decimal value specified after the #, e.g. #27 means use the decimal number 27
- Rn – use the value stored in register n, e.g. R6 means use the value stored in register 6.
- Assume the available registers that the programmer can use are numbered 0 to 7.



Logical bitwise operators

Assembly language instructions for logical operations are shown in Table 2 below.

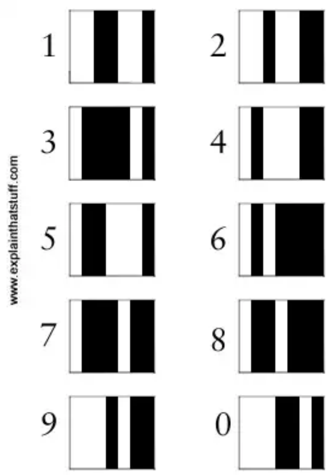
AND Rd, Rn, <operand2>	Perform a bitwise logical AND operation between the value in register n and the value specified by <operand2> and store the result in register d.
ORR Rd, Rn, <operand2>	Perform a bitwise logical OR operation between the value in register n and the value specified by <operand2> and store the result in register d.
EOR Rd, Rn, <operand2>	Perform a bitwise logical exclusive or (XOR) operation between the value in register n and the value specified by <operand2> and store the result in register d.
MVN Rd, <operand2>	Perform a bitwise logical NOT operation on the value specified by <operand2> and store the result in register d.
LSL Rd, Rn, <operand2>	Logically shift left the value stored in register n by the number of bits specified by <operand2> and store the result in register d.
LSR Rd, Rn, <operand2>	Logically shift right the value stored in register n by the number of bits specified by <operand2> and store the result in register d.
HALT	Stop the execution of the program.

Table 2

Input-Output Devices

Barcodes

There are two main types of barcodes, QR (Quick Response) codes – which can store far more information than 1D barcodes and 1D barcodes, such as those found on products, which are made of differing thicknesses and patterns of black and white rectangles.



Barcode Readers: There are a few types of barcode readers available, including pen-type readers, laser scanners, CCD readers and camera based readers.

In **Pen-type readers,** a light source and a photo diode are placed next to each other in the tip of a pen. To read a barcode, the tip of the pen is dragged across all the bars at an even speed. The photo diode measures the intensity of the light returning (I guess therefore it can't be a photodiode – since this would simply measure if light was coming back…) and therefore, due to the different reflectivity of white and black (black absorbs more light than white) can measure the widths of the bars and spaces in the barcode. The simplest systems make use of a specific thickness with black meaning 1, and white meaning 0. However, there are also more complex alphabetic system, such as the UPC (universal product code) alphabet, which has an entirely different pattern for each number. Pen-type scanners are the most durable type of scammers, however, they most come in contact with the barcode, so cannot be used for many applications. Additionally, it is sometimes challenging to use them, as one has to move at a very constant speed.

In **Laser Scanners**, the system is the same, however it uses a line of laser light with an array of light sensors. This is much faster and they are reliable and economical for low-volume applications, such as in supermarkets.

In **CCD (Charge-Coupled Device)** readers, there is a large array of tiny light sensors lined up in a row at the head of the reader, relying on the fact that there is anyway light that the white parts would reflect. Therefore, in the same way, the pattern could be found, and the barcode number be found.

In **Camera Based Readers,** a camera is used to take an image of the barcode. From here, there is a use of image processing tools to locate a barcode, reorientate and deal with any issues of alignment in all three axes, and then measure the thicknesses of black and white bars. Hence, the method could be used when the code is damaged or poorly printed. This is generally also much cheaper than other methods, as one could even use the hardware on a smartphone.

Digital Cameras

A digital camera makes use of a CMOS (Complementary Metal Oxide Semiconductor) sensor comprising of millions of tiny light sensors arranged in a grid. When the shutter opens, the light enters the camera and projects an image onto the the sensor at the back of the lens (which focuses the light). Each sensor measures the colur and the brightness of the light which is reaching it, before sensing the information to a microcontroller which can recombine the readings of all of these sensors into a complete image.

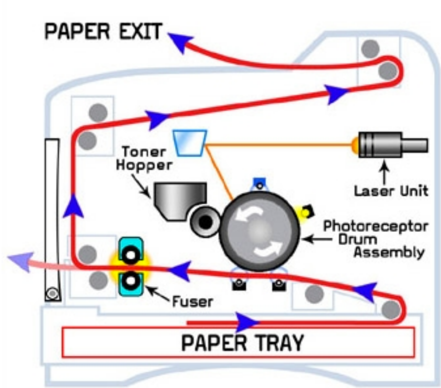
Some cameras (tend to be higher end) make use of a CCD sensor, since they produce a far higher quality image, however taking up far more power, and being more expensive, since there are fewer nodes between the individual sensors and the analogue-digital converters.

RFID

An RFID tag is useful, as, as opposed to barcodes they can be read from over 300 metres away, as well as being able to transfer more information, passing information between the transmitter and the receiver wirelessly. The RFI chip consists of a small microchip transponder and an antenna. Since so little is required, the tag can be very small, even the size of a grain of rice. There are two types of tags, active tags and passive tags. Active tags also contain a battery to power the tag so it actively transmits a signal for a reader to pick up. This means that they can also be picked up from further away. Passive tags instead make use of a radio wave emitted from a reader in order to proved sufficient electromagnetic power to the card using the antenna. Once powered, the transceiver inside the RFID tag can send the data to the receiver. These are much smaller, and cheaper, and are generally far more used, from everything from credit cards to oyster cards.

Laser Printer

A laser printer makes use of powered ink stored in a toner. The printer generates a bitmap image of the printed page and using a laser unit and a mirror, draws a negative, reverse image onto a negatively charged drum. The laser light causes the affected areas of the drum to lose their charge. The drum then rotates past a toner hopper to attract charged toner particles onto the areas which have not been lasered. Then the ink is bonded to the paper using heat and pressure.



Laser printers are affordable and are largely used when the volume of printing is high. They are also generally very fast, though colour laser printers (which use cyan, magenta and yellow as well as black) require the same process to be repeated 4 times for each colour take more time. Additionally, the quality of a laser printer is generally lower than the quality which can be achieved using an inkjet printer.

Storage Devices

A secondary storage device is needed because the other memory RAM, which is the memory that is directly accessed by the processor loses its contents when the computer's power is turned off. In order for the computer to be able to store any data for extended periods of time therefore, there must be another place where the data could be stored more permanently, and this is generally the hard disk, though today, there has been a switch towards Solid State Drives (SSDs).

Hard Disk: A hard disk uses rigid rotating platters coated with magnetic materials. The iron particles on the disk are polarised to be either north or south state – representing 0s or 1s. The disk is separated into concentric circles, and each track is subdivided into sectors.

The disk spins very quickly at speeds up to 10,000 RPM, allowing the drive head to move across the disk to access different sectors, using a magnet to read and write data to the disk. Finally, the hard disk likely makes use of a number of platters, each with its own drive head in order to increase the amount of data which could be read from a single hard disk, without vastly increasing the disks size.

As opposed to Solid State drives, which are much faster, since they use a number of flash ships, hard drives generally have very large capacities, allowing people to store all required information on them comfortably. Additionally, they tend to be very cheap.

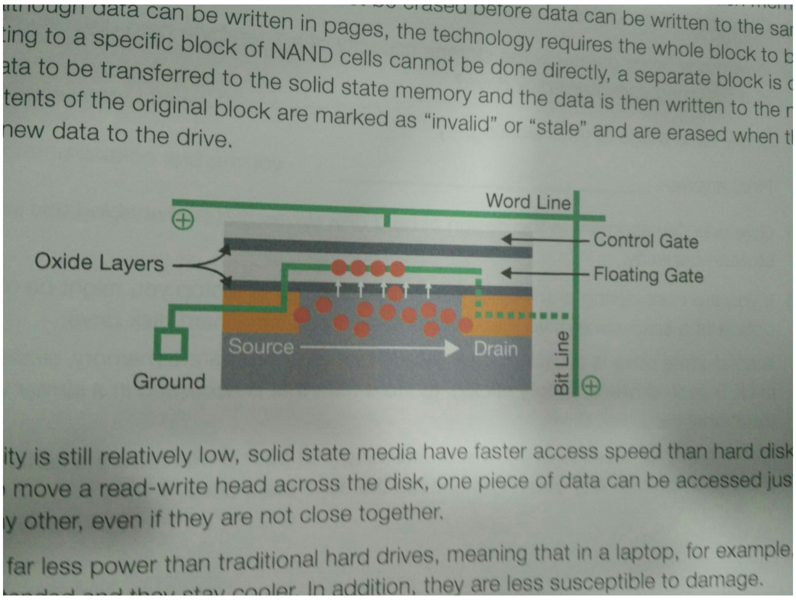
Optical Disks: Optical Disks (CD ROMs, DVDs and Blu-Ray disks) make use of a high powered laser to burn sections of its surface, making it less reflective at those points. Hence a laser at a lower power is used to read the disk by measuring how much light is absorbed and how much reflected by that part of the disk. Reflective and non-reflective areas are read as 1s and 0s respectively. There is generally only one track on an optical disk, generally arranged in a spiral.

CD ROMs have a max storage space of 650MB, while a Blu Ray can store around 50GB, since it uses a much shorter wavelength laser, meaning the thickness of the track is reduced.

Rewritable compact disks use a laser and a magnet to heat a spot on the disk and then set its state to become a 0 or 1 before it cools again.

A DVD-RW makes use off a phase change alloy that can change between amorphous and crystalline states by changing the power of the laser beam.

Optical storage is very cheap to produce and can be easily and safely distributed. However, it is easily damaged by excessive sunlight or scratches.



Solid State Drives (SSDs): SSDs make use of an array of chips arranged on a board, making use of a large number of NAND flash memory cells and a controller that manages the memory. The cell works by delivering a current along the bit to lead to electrons going to the floating gate. Hence, this floating gate can be measured to find if it is a 0 or 1.

Data is stored in pages (of around 4KB) grouped in blocks. Flash memory cannot be overwritten instead requiring the old data to be removed before the new data can be written to the same location. However, the individual pages cannot be erased, requiring instead that the entire block be erased. Hence, the rest of the data needs to be copied to another block before erasing the block and then moving the required data back.

Solid State Drives tend to have much lower capacities than Hard Disks as well as being much more expensive, though costs of SSDs are beginning to reduce over time. They are much faster than HDDs (lower latency), with the read/write speeds being almost double as fast.

4.8 – Consequences and Uses of Computing

Much of this chapter appears to be common sense and a general knowledge and reading of technology.

Developments in Computer Science and the Digital Technologies have dramatically altered the shape of communications and information flows in societies, enabling massive transformations in the capacity to:

- Monitor behaviour
- Amass and analyse personal information
- Distribute, publish, communicate and disseminate personal information.

Computer scientists and software engineers therefore have power, as well as the responsibilities that go with it, the algorithms that they devise and the code that they deploy should have some positive impact on the society – embedding moral and cultural values.

One problem that came to one computer scientist, which will be used as a case study, is Edward Snowden, when he found that the NSA (National Security Agency) was illegally (and immorally) using software to spy on people. He faced the moral issue of betraying his countries and risking exposing national secrets versus serving the moral good – telling the population and turning against the NSA.

Edward Snowden

- Revealed information to Guardian journalist in April 2013
- Handed over informayion about a program called PRISM, which allowed the NSA to get information from the world's largest Internet companies, including Facebook, Google, Yahoo, Microsoft, Apple, YouTube, AOL and Skype.
- Included information showing NSA employees how to use the surveillance capabilities and how to misuse the capabilities.
- Further information showed the actions of the NSA
    - Tapping of internet servers, satellites, underwater fibre-optic cables, local and foreign telephone systems and personal computers
    - Tapped individuals from people suspected of terrorist activities to private citizens from the US, to other citizens and even diplomats in UK and Europe (Angela Merkel and Nicolas Sarkozy were allegedly hacked).
    - Also got metadata from all calls from all calls to / from the USA
        - In one month in 2013 – collected data on more than 97 billion emails and 124 billion phone calls.

Cyber-Attacks

- **ESTONIA**
    - In 2007, Estonia suffered a series of cyber-attacks which swamped websites of organisations including the Estonian Parliament, banks, ministries, newspapers and broadcasters. This was one of the largest cases of state-sponsored cyber warfare every seen, sponsored by Russia in retaliation for the relocation of the Bronze Soldier of Tallinn, an elaborate Soviet-era grave marker, and war graves in Tallinn.
    - In 2008, an ethnic Russian Estonian national was charged and convicted of the crime.
- **SONY PICTURES**
    - In June 2014, the North Korean government threatened action against the US government if the movie 'The Interview' was released.
    - In November, Sony Pictures computers were hacked by the 'Guardians of Peace', a group believed by the FBI linked to the North Korean government.

Challenges of the Digital Age

Economic impact of the internet

- When Tim-Berners Lee created the World-Wide Web, he had intended it all to be free, with no intension to create money for himself or anyone else.
- However, according to many people, it simply created a new small group of companies which have accumulated huge wealth at the expense of a number of other small regular companies – therefore, leading to a number of people losing their jobs.
- **Amazon**
    - Started as an online bookstore in 1994
    - Diversified into DVDs, software, video games, toys, furniture, clothes and thousands of other products
    - In 2013, company turned over $75 bn in sales.
    - Now accounts for 65% of all digital purchases of book sales.
    - Now fewer to 4,000 bookshops – 1/3 less than 2005
        - Fewer people in jobs
        - Where bookshop employs 47 people per $10 bn Amazon employs 14
- **eBay**
    - 41,000 trading goods worth $7.2 mn in 1995
    - 162 mn users worth $227.9 bn in 2014
- **Destruction of Jobs**

        - 2013 paper by Carl Benedikt Frey and Michael Osborne
            - 47% of total US employment is at risk
            - *The Future of Employment: how susceptible are jobs to computerisation*
            - Examined 700 individual occupations
            - In the 10 jobs that have a 99% likelihood of being replaced by software and automation within the next 25 years, Frey and Osbourne include tax preparers, library assistants, clothing factory workers, and photographic process workers.
                - In fact, jobs in photographics industries have already all but vanished
                - In 1989 – Kodak employed 145,000 people – market cap of $31 bn
                - In 2013, the company filed for bankruptcy.
                - **INSTAGRAM**
                    - Meanwhile, Instagram boomed – 25,000 people downloaded the app on 6th October 2010
                    - A month later, it had a million users.
                    - By early 2012, it had 14 mn users and by November, 100 mn users, with the app hosting 5bn photos.
                    - When it was sold for $1bn in 2012 – still had only 13 full-time employees.
        - User-Generated Content
            - Sites like YouTube have taken money (and jobs) away from the traditional movie industries.
    - **TROLLS**
        - Trolls, Cyber-Bullying and Misogyny has become a fact of everyday life on the internet.
        - 2010-2011 Arab Spring
            - Originally served a good purpose – allowing people to rise up against tyrannical regimes
            - However, began to provoke hatred and serves an anonymous place where people can use religious hatred.
        - Feminist writers and journalists receive hundreds of death threats and rape threats.
        - Savage bullying on various social networking sites have led to multiple (many of them teen) suicides.

4.9 – Fundamentals of Communications and Networking

4.9.1 - Communications

Data communication involves sending and receiving data from one computer or device to another. Data communication applications include e-mail, supermarket electronics point of sale terminals, cash dispensers, cell phones and voice over IP.

Data communications also takes place within the CPU and between the CPU and its peripheral devices, for example, data and addresses along the data and address bus between the processor and memory, and data is transferred between memory and storage and other peripheral devices.

4.9.1.1 - Communication Methods

Serial Transmission

- Bits are sent via an interface one bit at a time over a single wire from the source to the destination.
- Very high transfer rates are possible
    - Fibre-Optic (using light) – 50Mbps to 100Gbps

- Much cheaper than serial transmission due to the reduction in number of wires required.
- There is no 'crosstalk' – where there is a skew leading to blurring of simultaneous transmission in parallel transmission. This becomes even more pronounced when the signal frequency or the length of communication increases.
- Much more reliable over greater distances.
- Signal frequency can be much higher than with parallel transmission, resulting in a higher net data transfer rate even though less data is transferred per cycle.

Parallel Data Transmission

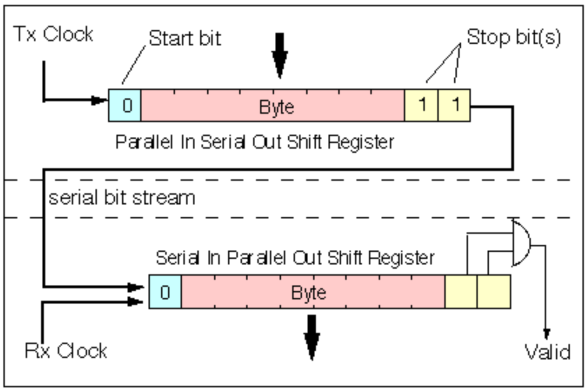
- Several bits are sent simultaneously over a number of parallel wires.
- The method is used inside the computer (using the various computer buses) and for very short distances of up to a few metres.
- Parallel transmission will transmit data more quickly than serial transmission – however there is the possibility of skew – where bits sent at the same time may reach at different times – hence only used for short distances.

Synchronous Transmission

- In synchronous transmission, data is transferred at regular intervals which are timed by a clocking signal, allowing for a constant and reliable transmission for time-sensitive information.
- Generally used by parallel transmission, for example in the CPU

Asynchronous Transmission

- Using asynchronous transmission, one character is sent at a time, with each character being preceded by a start bit and an end bit.
- The start bit alerts the receiving device and synchronises the clock inside the receiver ready to receive the character – ensuring the baud rate at the receiving end is the same as the sender's baud rate.
- A parity bit is also included to check against incorrect transmission, thus for every character, 10 bits are required – start bit, parity bit and end bit.
- The start bit can be either a 0 or a 1, with the end bit being the other one.



- This method is generally used by computers when communicating with peripherals.

4.9.1.2 - Communication Basics

Bit Rate: The speed at which data is transmitted serially, measured in bits per second.

Baud Rate: The rate at which the signal changes in voltages.

In baseband mode of operation, the bit rate is equivalent to the baud rate. However, with higher bandwidths, more than one bit can be coded into a signal and the bit rate will be higher than the baud rate – by making use of a number of different frequencies of the wave.

Bandwidth: The bandwidth is the amount of data that can be transmitted in a fixed amount of time. It is usually expressed in bits per second (bps) – since there is a direct proportionality between bandwidth and bit rate – the higher the bit rate, the higher the bandwidth.

Latency: Latency is the time delay between the moment the first byte of a communication starts and when it is received at its destination. It is a function of how long it takes information to travel at the speed of light from source to destination.

Protocol: A protocol is an agreed-upon format for transmitting data between two devices. This includes a number of things including:

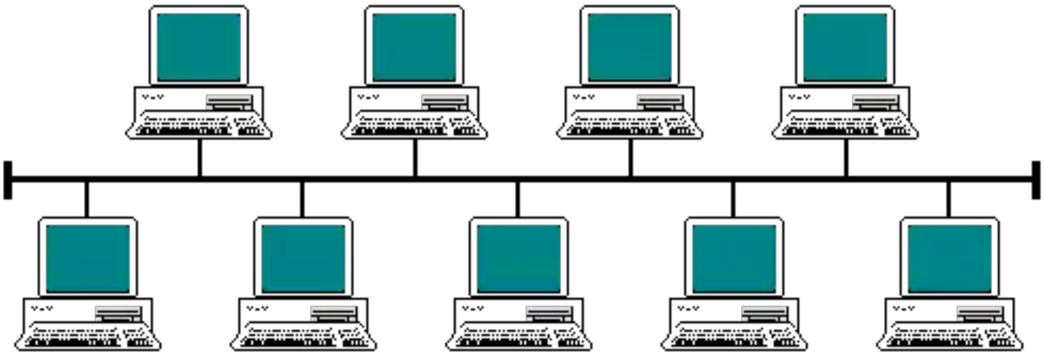
- Physical Connections and Cabling
- Mode of transmission (serial or parallel)
- Speed
- Baud and bit rate
- Data format
- Parity
- Error detection and correction

4.9.2 -Networking

LAN (local area network): A LAN consists of a number of computing devices on a single site connected together by cables, connecting a number of PCs, printers and scanners and a central server. Users can communicate with each other as well as sharing data and hardware devices such as printers. LANs can transmit data very fast but only over a very short distance and there is a limit to the number of computers that can be connected to a single LAN.

4.9.2.1 - Network Topology

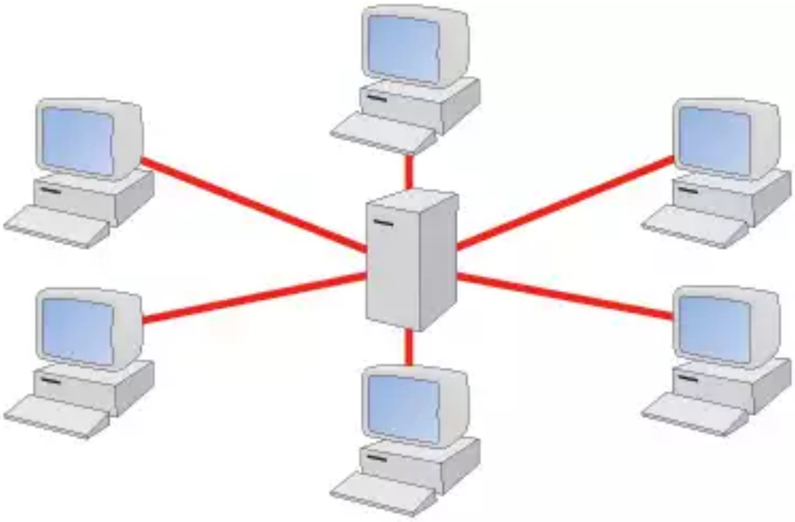
Bus Topology



- In a Bus topology, all computers are connected to a single cable, with the ends of the cable connected to a computer or a terminator.
- Data is transmitted in one direction only at any one time – where all the traffic has equal transmission priority.
- A device wanting to communicate with another device sends a broadcast message onto the wire that all other devices see, but only the intended recipient can accept and process the message, with the system using the system of **Carrier Sense Multiple Access / Collision Detect (CSMA/CD).**
- Inexpensive to install as it requires less cable than a star topology and does not require any additional hardware.
- New devices can be easily added without disrupting the network.
- Well suited to small networks not requiring high speeds.
- If the main cable fails, the whole network will go down.
- Limited cable length and number of stations

- The performance of the system will degrade with heavy traffic.
- There is a relatively slow security, where all the computers on the network can see all data transmissions.

Star Topology



- A star network has a central node which may be a switch, hub or computer which acts as a router to transmit messages
- *A hub simply transmits all the messages received from one line to all other lines, whereas a switch or router can be more clever, saving where the computer is coming from (using the MAC address) and only sending the messages to the specific computer that it should be sent to.*
  - **MAC Address:** Every computer device has a **Network Interface Card (NIC),** which would have a unique media access control address (MAC address) which is assigned and hard-coded into the card by the manufacturer and it uniquely identifies the device.
- If one cable fails, only one station is affected, so it is easy to isolate faults
- Consistent performance even when the network is being heavily used
- Performance is better than bus network
- No problems with collisions of data
- Messages are more secure
- Easy to add new stations without disrupting the network
- Different stations can communicate with the switch using different protocols.
- More costly, because more cables are required, as well as a central hub or server.
- If the central device fails, the entire network will go down.

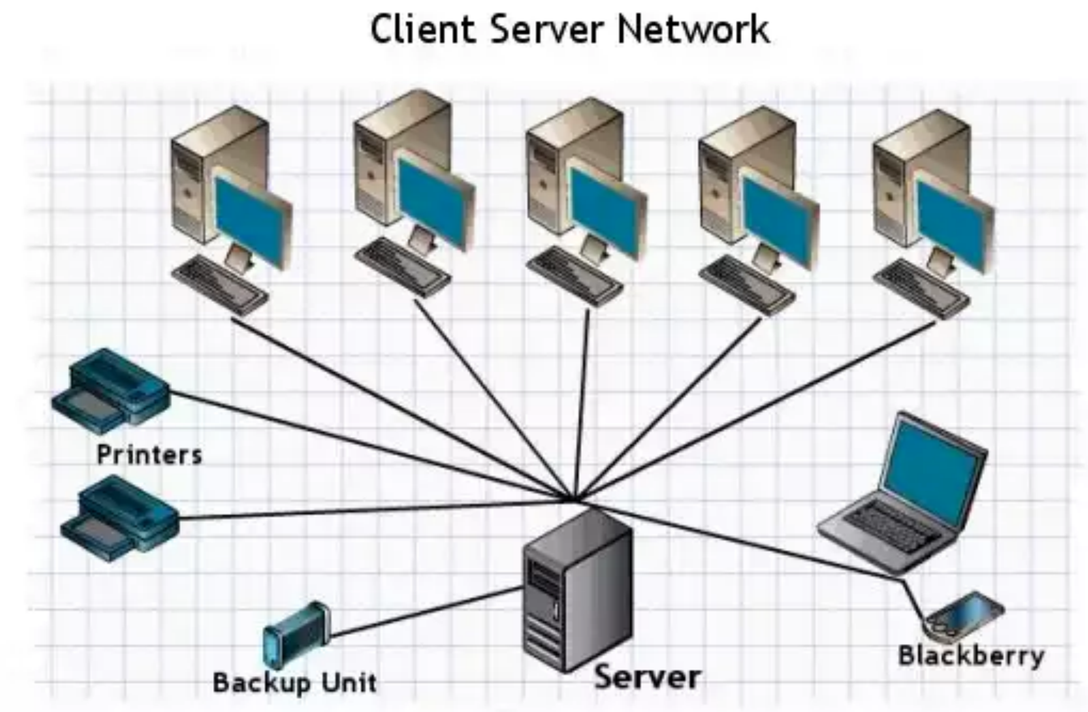
Physical vs Logical Topology

The physical topology of a network is the actual design layout, which describes the wiring scheme.

The logical topology is the path in which the data travels and describes how components communicate across the physical topology

4.9.2.2 - Types of Networking Between Hosts

Client Server Networking



- In a client-server network, one or more computers known as **clients** are connected to a powerful central computer known as the **server.** Each client may hold some of its own files and resources such as software, and can also access resources held by the server.
- In a large network, there may be several servers, each performing a different task.
- In a client-server network, the client makes a request to the server which then processes the request
- The security is better, since all files are stored in a central location and access rights are managed by the server.
- Backups are done centrally so there is no need for individual users to back up their data.
- Data and other resources can be shared.
- It is expensive to install and manage
- Professional IT staff are needed to maintain the servers and run the network.

Peer-to-Peer Networks

- In a P2P network there is no central server. Individual computers are connected to each other, either locally or over a wide area network so that they can share files.
- Generally, most used (legally) in small local area networks because.
    - It is cheap to set up
    - It enables users to share resources such as a printer
    - It is not difficult to maintain.
- P2P Problems
    - Has been widely used for online piracy – impossible to trace the files which are illegally downloaded.
    - Piracy Sites attracted 53 billion visits each year.
    - 432 million unique Web users actively searched for content that infringes copyright.

4.9.2.3 - Wireless Networking

Wi-Fi

- Local area wireless technology that enables you to connect to a device such as a PC.
- Via a Wireless Network Access Point (WAP)
- Generally, has a range of about 20 metres indoors, and more outdoors.

- In 1999, the Wi-Fi Alliance was formed to establish international standards for interoperability and backward compatibility.
    - The alliance consists of a group of several hundred companies around the world, and enforces the use of standards for device connectivity and network connections
- **Components Required**
    - The computer requires a Wireless Network Interface Controller.
        - Combination of computer and interface controller is called a station.
    - All stations share a single radio frequency communication channel.
    - **In order to connect to the internet, the WAP usually connects to a router, but it can also be an integral part of the router itself.**
- **Security**
    - Security protocols used to secure wireless networks include Wi-Fi Protected Access (WPA) and Wi-Fi Protected Access II (WPA2)
    - WPA2 is built into wireless network interface cards, and provides strong encryption of data transmissions, with the generation of a 128-bit key for each packet being sent.
    - Each Wireless Network makes use of a password to prevent unauthorised people from joining the network.
        - One can recognise a specific network using the correct **SSID (Service Set Identifier)**
    - **Whitelists**
        - Some people can use MAC address whitelists to control who is allowed on the network – based on the devices MAC address.

CSMA/CA (Carrier Sense Multiple Access / Collision Avoidance)

- Protocol for carrier transmission in Wireless LAN networks.
- Prevents collisions before they happen – a collision results in the data being lost
    - CSMA/CD (Collision Detect) deals with detecting when collisions occur.
- Prior to transmission, a node first listens for signals on the wireless network is transmitting. If a signal is detected, it waits for a period of time for the node to stop transmitting and then listens again.
- **With RTS / CTS (Request to Send / Clear to Send)**
    - Optionally used when determined that no other node is transmitting.
    - Counteracts the problem of hidden nodes
        - Nodes which the WAP can hear but not the node that is transmitting.
    - It involves sending a signal directly to the WAP alone before hearing a signal from the WAP that the node can send the signal

4.9.3 - The Internet

- **Structure of the internet**
    - Each continent uses backbone cables connected by trans-continental leased lines fed across the sea beds.
    - ISPs connect directly to this backbone and distribute the internet connection to smaller providers.
- **URL**
    - Uniform Resource Locator is the full address for an internet resource - specifies the location of a resource on the internet
    - Made up of: Method, Host, Location, Resource
- **Internet Registries and Registrars**

- **Registries** required to ensure that a particular domain name is only used by one organisation, and they hold records of all existing website names and the details of those domains are available to purchase.
    - 5 global organisations governed by ICANN
    - Allocate IP addresses and keep track of which addresses a domain is associated with as part of the DNS
- **Domain Names and the Domain Name System (DNS)**
    - Domain name identifies the area or domain that an internet resource resides in.
    - Structured into a hierachy of smaller domains.
    - Each had one or more equivalent IP addresses.
- **FQDN**
    - **Domain name that includes the host server name - properly identifies exactly which website to visit.**
- **IP addresses**
    - IP address is a unique address that is assigned to a network device.
    - Indicates where a packet of data is to be sent or has been sent from.
    - **Routers can use this address to direct the data packet accordingly.**
- **PACKET SWTICHING AND ROUTERS**
    - Packet switching is a method of communicating packets of data across a network on which other similar communications are happening simultaneously.
    - Data transmitted across a network is broken down in more manageable chunks called **packets**.
    - Size of each packet in a transmission can be fixed or variable.
    - Each packet has a header and a payload
        - Most may also use a section with a checksum or Cyclical Redundancy Check.
        - Header contains information for the packets to be reassembled
            - Number of items in sequence
            - Item number in sequence.
            - Protocol being used with the packet.
            - Also includes a **TIME TO LIVE (TTL) or HOP LIMIT,** after which point the data packet expires.
    - **ROUTING PACKETS ACROSS THE INTERNET**
        - Success of packet switching relies on the ability of packets to be sent from sender to recipient along entirely separate routes from each other.
        - Can be easily reassembled at the receiving end

- **Routers**
    - Routers are used to connect at least two networks.
    - Routers use routing tables to store and update the locations of other network devices and the most efficient routes to them.
    - The routing algorithm used to decide the best route can become a bottleneck in network traffic since the decision making process can be complicated.
- **Gateway**
    - Routing packets from one another requires a router if the networks share the same protocol.
    - Where these protocols differ between networks, a gateway is used rather than a router to translate between them.
    - All the header data is stripped from the packet, in the format of the new network.

- **4.9.3.2 - Internet Security**
    - FIREWALL
        - Firewall is a security checkpoint designed to prevent unauthorised access
        - Can be implemented in both hardware and / or software.
        - Firewall consists of a separate computer containing two NICs, with one connected to the internal network and one connected to the external network.
            - Each data packet that attempts to pass through the two NICs is analyses against rules (packet filters), then accepted or rejected.
            - May also act as a proxy server.
    - PACKET FILTERING
        - Also known as static filtering - contros access according to network admin rules and policies
        - Looks at source and destination IP addresses - check if they are on a whitelist and / or blacklist.
        - Can also be based on the protocols and the port numbers it is trying to access.
        - Certain protocols use certain ports - eg Telnet uses port 23.
            - Can be denied or rejected.
    - Stateful Inspection
        - Rather than relying on the IP addresses, protocols and port numbers to govern packet's safe passage, stateful inspection or dynamic filtering can examine the contents of a data packet to better assess it for safety.
        - Can see if things are a response
        - Can also make contextual rules based on the passage of previous packets.
    - Proxy Server

    - Intercepts all packets entering and leaving a network, hiding the true network addresses of the source from the recipient.
    - Enables privacy and anonymous surfing.
    - Can also maintain a cache of websites commonly visited.
- Encryption
    - Symmetric Encryption
        - **Private Key Encryption**
        - Uses the same key to encrypt and decrypt data - therefore key must also be transferred.
        - Key can be intercepted.
    - Assymetric Encryption
        - Uses two separate, but related keys.
        - One key, known as public key.
            - People encrypt using this.
        - Other key, known as private key.
            - Data can be decrypted using this key.
        - However, the data can be encrypted using a public key and sent by a malicious third party impersonator pretending to be you.
            - Data can be digitally signed.
- **Digital Signatures**
    - Sender creates a message with a checksum. Encrypts with his own private key - therefore, the other person can test who the person is by decrypting with the persons public key (and recalculating the checksum and checking it is the same as expected).
    - Data and time can also be included in the original message.
    - However, digital signatures could be created using a bogus private key claiming to be that of a trusted individual.
        - Therefore, a digital certificate verifies that a sender's public key is formally registered to that particular sender.
            - A certificate is issued by official certificate authorities, such as Symantec or Verisign and verifies the trustworthiness of a message sender or website.
            - Certificate allows the holder to use the Public Key Infrastructure.
- **Worms, Trojans and Viruses**
    - All types of malware or malicious software.
    - **Viruses and Worms**

            - Have the ability to self-replicate by spreading copies of themselves.
            - Worm is a sub-class of virus, the difference between the two is that viruses rely on other host files to be opened to spread.
                - Worms spread themselves - replicate without user interventions.
                - Once a virus is in memory, any other uninfected file that runs, becomes infected when it is copied into memory.
                - Worm generally enter computer through a vulnerability.
        - **Trojans**
            - Trojan is generally a part of the other programs.
            - They open a backdoor to your computer systems that the trojan creator can exploit.
            - This can be order to harvest your personal information or to use your computer power and network bandwidth.
            - Trojans cannot self-replicate.
    - **System Vulnerabilities**
        - Malware exploits vulnerabilities in our systems, be they human error or software bugs.
        - How to protect against it
            - Firewall
            - Being aware of administrative rights.
            - Improve code quality
                - Often buffer overflow is used - when program accidentally writes values to memory locations too small to handle them.
                - Can take advantage of this by forcing the program to write something to memory, which may consequently alter its behaviour.
            - Be aware of Social Engineering
                - Therefore education
                - Spam filtering
            - Regular OS and antivirus software updates will also help to reduce the risk of attack
            - Virus checkers.
- **4.9.4 - TCP / IP**
    - TCP / IP stack is a set of networking protocols that work together as four connected layers, passing incoming and outgoing data packets
        - Application layer

    - Application layer sits at the top of the stack and uses protocols relating to the application being used to transmit data over a network.
    - Chooses the appropriate higher-level protocol for the communication, such as HTTP (Hypertext Transfer Protocol), POP3, and FTP.
  - Transport layer
    - Transport layer uses TCP (Transmission Control Protocol) to establish an end-to-end connection with the recipient computer.
    - Data split into packets and labelled with a port number, the total number of packets and the port number.
  - Network layer
    - Network layer, or internet layer, adds the source and destination IP addresses.
    - Routers act on the network later and will use these IP addresses to forward the packets onto the destination.
    - Socket specific (IP address + port together) to which device the packet must be sent to and the application being used on that device.
  - Link layer
    - Link layer is the physical connection between network nodes and adds the Media Access Control (MAC) address, identifying the NICs of the source and destination devices.
    - These means that once the packet finds the correct network using the IP address.
    - Can then locate the exact device.
    - Unless two computers on the same network, destination MAC address will be the MAC address of the first router that the packet is sent to.
    - At receiving end, the MAC address is stripped off by the link layer, which passes the packets onto the network layer.
- **MAC Addresses**
  - **MAC address** is a unique 12-digit hex code that is hardcoded on ever NIC during manufacture.
- Well known ports

Server Port Number	**Protocol**
20	FTP
21	FTP
22	SSH
23	Telnet
25	SMTP (Simple Mail Transfer Protocol)
80	HTTP
110	POP3 (Post Office Protocol v3)

143	IMAP (Interim Mail Access Protocol)
443	HTTPS (Hypertext Transfer Protocol Secure)

- **FTP**
  - **FTP** is a simple method to transfer files across the Internet.
  - Works as a very high level protocol in the Application Layer.
  - Using a client on top of the protocol, user actions can generate the FTP commands automatically.
  - Moet FTP sites require a username and password to be used, but some are configured to allow anonymous use without the need for any login information.
- **SSH**
  - Used for remotely accessing and managing a computer.
  - Replace telnet, which used no encryption.
  - Similar to MS-DOS commands.
  - Can use SSH with other application level protocols which means that you can create a 'tunnel' through which HTTP, POP3 or SMTP requests can operate. This means that an HTTP GET can be sent securely and bypass any network restrictions which have been imposed.
- **Mail Server**
  - Mail server acts as a virtual post office for incoming and outgoing email.
  - Servers route mail according to its database of local network user's email address.
  - POP3 is responsivble for retrieving emails from a mail server that temporarily stores incoming emails.
    - When retrieved, they are transferred to a local computer and deleted server.
    - Therefore, only accessible on one device.
  - IMAP stores emails on the server.
  - SMTP is used to transfer outgoing emails from one server to another or from the email client to the server.
- **Web Servers**
  - Web server hosts website and handles client requests - using HTTP to send content to users who want to view pages of the website.
  - Stored as text filed (HTML, CSS and JavaScript)
  - Rendering web pages
    - When HTTP response is received, it is parsed to fit a hierachical model.
    - HTML broken down into Document Object Model (DOM) tree.
    - CSSDOM for css and related to HTML tags.
    - Further requests for any required images or other resources must be used.

    - Lastly, JavaScript is parsed and executed as required or event handlers established.

- **IP address (4,9.4.3 - 4.9.4.9)**
  - Unique numerical address used to identify a host computer or network node trying to communicate over IP on the Internet.
  - Two standards
    - IPv4 (1983)
      - 4 Billion addresses
      - 32 Bits
    - IPv6 (1999)
      - 128 Bits
  - Written in dotted decimal - each part represents a 8-bit binary pattern (uptil 255)
  - Reserved Ips
    - 127.x.x.x
    - x.x.x.0 - Network Identifier
    - x.x.x.255 - Reserved as the broadcast address on that subnet where data is sent to all hosts.
    - x.x.x.1 - Generally the router.
  - Network ID and Host ID (IP address is split into these parts)
    - Classless addressing
      - Which class this was a member of, determined the number of bits used for the network id
      - Eg, class A had 7 bits of network ids.
    - Classful addressing
      - Defines the number of bits in the subnet mask
        - Eg. 103.27.104.92/24
        - Therefore first 24 bits are of the network id and the rest are the host id.
  - **Subnet Masking**
    - Subnet Mask is used in conjunction with an IP address to identify the two unique parts of the address.
    - 255.255.255.0 therefore 24 bits for network id.
    - ANDed with the IP address to separate network ID from the full ip address
    - **Subnetting**
      - Network admin of large organisation can create subnetwork segments to ease the management of the system and improve efficiency.

            - Subnetting reduces the size of the broadcast domain, therefore improving security, speed and reliability.
        - Subnet ID is created by using the most significant bits of the host ID and altering them.
    - **Public and Private IP addresses**
        - Public IP address is globally unique and can be addressed directly by any other computer in the world.
        - Company's web server or home internet router would require a public IP address.
        - Within the local network, addresses can be private and the web server can forward all data going through to the correct device.
        - Therefore, uses Network Address Translation to convert from public address to private address by the router.
    - **NAT**
        - When passing through a public address space - for example router.
        - Eg, changing the IP address on packet to the public facing IP address.
            - Also logs this request on a translation table - therefore when the response is received, it looks this up in the table and returns the traffic to the correct machine.
        - Also offers an additional layer of security by automatically creating a firewall between the internet and external networks.
    - **Port Forwarding**
        - Product of NAT
        - When a public computer is attempting to communicate with a computer on the private network, the NAT needs to forward all the responses to a specific private IP address.
        - Therefore, can forward all requests to a certain port to a certain private IP address.
    - **DHCP**
        - **Dynamic Host Configuration Protocol**
            - Used to automatically assign a dynamic IP address from a pool of available IP addresses to a computer attempting to operate on the public network.
            - Since IP addresses are in short supply.
            - Allows computers to only use IP address when they are active and then return it to the pool when done.
            - Also provides the Subnet mask and other automatic configuration details.
- **Client Server Model**
    - In client server model, client will send a request message to a server which should respond with the data requested.

    - For example with an HTTP request
- **API (APPLICATION PROGRAMMING INTERFACE)**
    - API is a set of protocols that governs how two applications should interact with one another.
    - An API sets out the format of requests and responses between a client and server and enables one application.
    - **WebSocket Protocol**
        - Another example of an API
        - WebSocket is a modern application layer protocol that facilitates a persistent bi-directional communication channel between the client and the server over a single line.
        - Known as full-duplex communication
        - Greatly reduced in size as they use much less header information.
            - All packets are accepted without any security checks.
- **Web CRUD applications**
    - **Create, Retrieve, Update, Delete**
    - Four fundamental operations of a content management system
    - Related to the four main functions of databases
        - INSERT, SELECT, UPDATE, DELETE
- **REST**
    - **Representational State Transfer**
    - Style of systems design that relies upon HTTP request methods to interact with online databases, via a web server.
    - Client computer requires no knowledge of how the server is likely to fulfil the request, how the data is stored or where it will gather the data.
    - The separation allows any client or server to be updated and developed independently of each other without any loss of functions.
    - **HTTP request methods**
        - Uses common standards of verbs or actions
            - GET, POST, PUT, PATCH, DELETE
- **Json vs XML**
    - **Javascript Object Notation vs Extensible Markup Language)**
    - They are the two methods for transfering data between the server and the web application.
    - Advantages of JSON vs XML

- Easier for a human to read
- More compact
- Easier to create
- Easier for computers to parse (and therefore quicker)

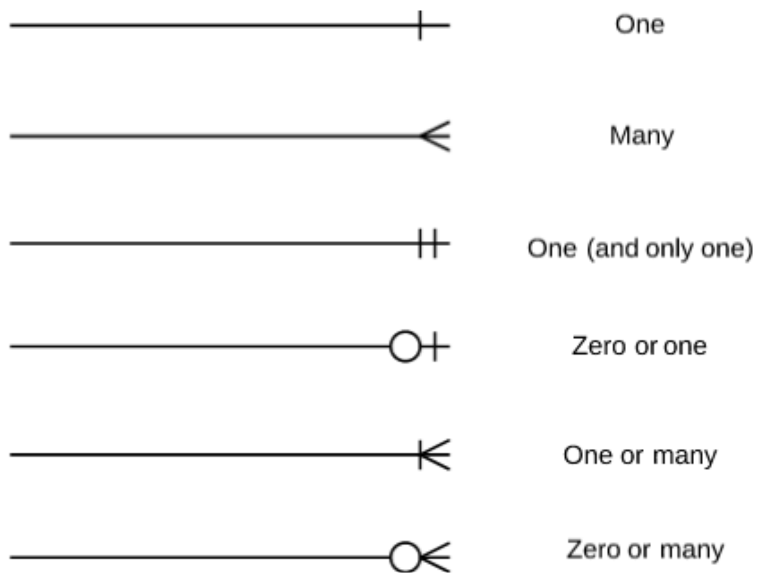
- **Thick vs Think Client Computing**
  - Refers to the level of processing that it does compared to the server it is connected to.
  - More processing and storage that a server does, the thinner the client becomes.

	+	-
Thin-Client	Easy to set up, maintain More secure Easier to update	Reliant on the server Requires powerful and reliable server - therefore expensive Server demand and bandwidth increased Takes more battery on the client
Thick Client	Robust and reliable Can operate without continuous connection to server Generally better for running more powerful software applications	More expensive client computers required Software must be installed on all client computers Integrity issues with distributed data

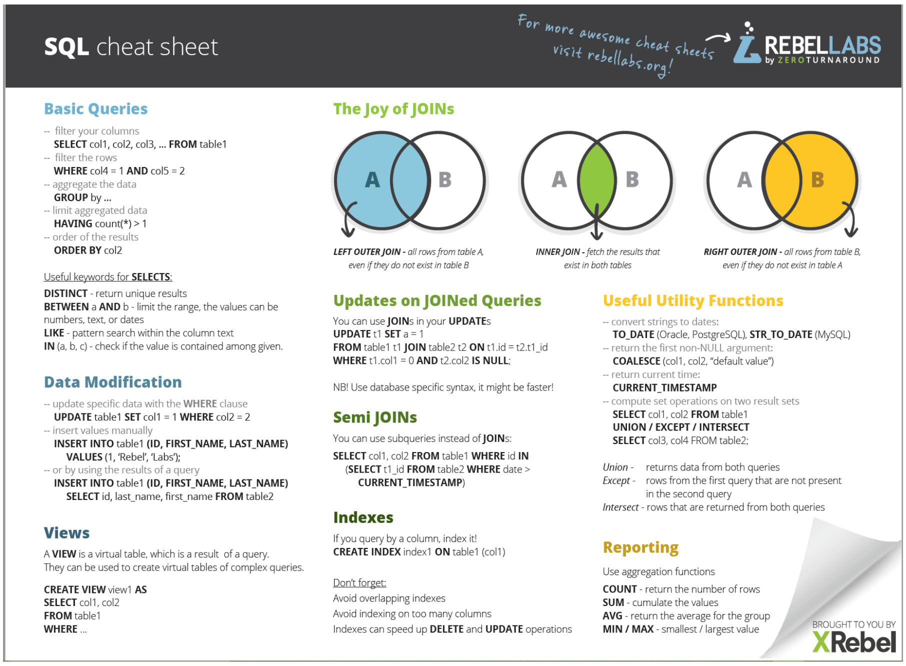
4.10 – Fundamentals of Databases

- Data Models and Entity Relationship Models
- Relational Databases
- Database design and normalisations
- SQL
- Client Server Databases - Concurrent access

- **Entity Relationship Modelling**
  - **Entities** is a category of object, person, event or thing of interest to an organisation about which data is to be recorded.
    - Each entity can have a number of attributes
      - **Characteristic of an entity.**
  - **Entity description written as Entity1(Attribute1, Attribute2 …)**
  - Each entity needs to have an entity identifier which uniquely identifies the entity
    - In a relational database, this is the primary key.
    - This is underlined in the entity description
- **Entity Relationship Diagrams**

- 

- **Concept of a relational database**
  - **Relational Database is a collection of tables in which relationships are modelled by shared attributes.**
  - Foreign Key: attribute that creates a join between two tables.
    - Attribute that is common in both tables.
  - Composite Primary Key: A primary key which consists of more than one attribute.
- **Normalisation**
  - **Why?**
    - Maintaining and modifying the database
    - Faster sorting and searching
    - Deleting records
  - **What?**
    - Process used to come up with the best possible design for a relational database.
    - Organised such that
      - No data is unnecessary duplicated
      - Data is consistent
      - Structure of database is flexible enough to allow to enter as many or as few items.
      - Structure should enable a user to make complex queries
    - **1st**
      - **NO REPEATING ATTRIBUTES**
    - **2nd**
      - **NO PARTIAL KEY DEPENDENCIES**
    - **3rd**
      - **NO NON KEY DEPENDENCIES**
- **SQL**

- 

SQL cheat sheet

For more awesome cheat sheets visit rebellabs.org!

REBELLABS by ZEROTURNAROUND

Basic Queries

-- filter your columns
SELECT col1, col2, col3, ... **FROM** table1
-- filter the rows
WHERE col4 = 1 **AND** col5 = 2
-- aggregate the data
GROUP by ...
-- limit aggregated data
HAVING count(*) > 1
-- order of the results
ORDER BY col2

Useful keywords for **SELECTS**:
DISTINCT - return unique results
BETWEEN a **AND** b - limit the range, the values can be numbers, text, or dates
LIKE - pattern search within the column text
IN (a, b, c) - check if the value is contained among given.

Data Modification

-- update specific data with the **WHERE** clause
UPDATE table1 **SET** col1 = 1 **WHERE** col2 = 2
-- insert values manually
INSERT INTO table1 **(ID, FIRST_NAME, LAST_NAME)**
VALUES (1, 'Rebel', 'Labs');
-- or by using the results of a query
INSERT INTO table1 **(ID, FIRST_NAME, LAST_NAME)**
SELECT id, last_name, first_name **FROM** table2

Views

A **VIEW** is a virtual table, which is a result of a query.
They can be used to create virtual tables of complex queries.

CREATE VIEW view1 **AS**
SELECT col1, col2
FROM table1
WHERE ...

The Joy of JOINs

***LEFT OUTER JOIN** - all rows from table A, even if they do not exist in table B*

***INNER JOIN** - fetch the results that exist in both tables*

***RIGHT OUTER JOIN** - all rows from table B, even if they do not exist in table A*

Updates on JOINed Queries

You can use **JOIN**s in your **UPDATE**s
UPDATE t1 **SET** a = 1
FROM table1 t1 **JOIN** table2 t2 **ON** t1.id = t2.t1_id
WHERE t1.col1 = 0 **AND** t2.col2 **IS NULL**;

NB! Use database specific syntax, it might be faster!

Semi JOINs

You can use subqueries instead of **JOIN**s:
SELECT col1, col2 **FROM** table1 **WHERE** id **IN**
(**SELECT** t1_id **FROM** table2 **WHERE** date >
CURRENT_TIMESTAMP)

Indexes

If you query by a column, index it!
CREATE INDEX index1 **ON** table1 (col1)

Don't forget:
Avoid overlapping indexes
Avoid indexing on too many columns
Indexes can speed up **DELETE** and **UPDATE** operations

Useful Utility Functions

-- convert strings to dates:
TO_DATE (Oracle, PostgreSQL), **STR_TO_DATE** (MySQL)
-- return the first non-NULL argument:
COALESCE (col1, col2, "default value")
-- return current time:
CURRENT_TIMESTAMP
-- compute set operations on two result sets
SELECT col1, col2 **FROM** table1
UNION / EXCEPT / INTERSECT
SELECT col3, col4 FROM table2;

Union - returns data from both queries
Except - rows from the first query that are not present in the second query
Intersect - rows that are returned from both queries

Reporting

Use aggregation functions
COUNT - return the number of rows
SUM - cumulate the values
AVG - return the average for the group
MIN / MAX - smallest / largest value

BROUGHT TO YOU BY XRebel

- **Defining an updating tables**
  - CREATE TABLE name
  - {

    Att1 INTEGER NOT NULL PRIMARY KEY,
    Att2 Varchar(100),
    Att3 Date,
    Att4 Currency
  - }
  - ALTER TABLE name
  - ADD att5 VARCHAR(10)
  - DROP COLUMN Att3
  - MODIFY COLUMN Att3 VarChar(1) NOT NULL

- **Client-Server Databases**
  - Many modern serves allow a service for a client-server operation.
  - Advantages are:
    1. Consistency of data as it is only stored in one place.
    2. Expensive Resource can be made available to a large number of users.
    3. Access rights can be managed centrally
    4. Backup and recovery.
  - PROBLEMS
    1. People simultaneously updating things.
       a. Methods of fixing this:
          i. When an item is updated, the entire block is copied to local memory area and when the record is saved, the block is rewritten to the server.
          ii. BUT, then people's changes can be lost
  - **Concurrency Fixes**
    1. **RECORD LOCKS**
       - **Technique of preventing simulataneous access to objects in a database in order to prevent updates being lost or inconsistancies in the data from arising.**
       - **Record is locked whenever a user retrieves it for editing or updating.**

- **PROBLEMS**
    - **Deadlock - if people need to access two records**
    - **Therefore other things can be used as well.**

2. **SERIALISATION**
    a. **Transaction cannot start until the previous one has finished.**
3. **TIMESTAMP ORDERING**
    a. When a transaction starts, it is given a timestamp.
    b. If two transactions attempt to affect the same object, the one with the later timestamp is ignored.
    c. Every object has a read and write timestamp
        i. When completing an update - reading it sets the timestamp to the correct timestamp
        ii. When it writes, it can check if there is another transaction which has read the record (is the write timestamp changed)
        iii. Therefore, allows problems to be avoided.
4. **COMMITMENT ORDERING**
    a. Transactions are ordered in terms of their dependencies on one another as well as the time when they were started.
    b. Can stop deadlock by blocking certain requests until others have been completed.

4.11 – Big Data

What is Big Data?
Big Data is a term used to describe data whose volume is too large to fit on a single server and is generally unstructured
It can be described in terms of:

1. VOLUME
2. VELOCITY
3. VARIETY

LACK OF STRUCTURE: This is the important part of the data, rather than its volume. This poses a number of challenges, such as that it becomes harder to analyse the data while relational databases are not appropriate because they require the data to fit into row and column format

Big Data collection and processing enables us to detect and analyse relationships within and among individual pieces of information that previously we were unable to grasp.

How to deal with it:
Machine learning techniques are needed to discern patterns in the data and to extract useful information from the data.

WHEN IT CAN NO LONGER FIT ON A SINGLE SERVER:

- Processing distributed across more than one machine
- **Functional Programming is better than procedural programming**

Why Functional Programming?

1. No side effects - it doesn't modify the state of the calling program in any way
2. They are stateless - the program's behaviour does not depend on how often a function is called or in what order they are called.

3. Supports higher order functions - allows things to be done very quickly and simply using functional programming
4. Variables are immutable.
5. Code can be easily be made to work over more than one server.

FACT BASED MODEL
Alternative to relational data model
Data is never deleted, just immutable fact stored with timestamps - so always able to determine what was in the past and what is in the future.
Each fact captures a single piece of information

GRAPH SCHEMA

Graphs can be used to represent connected data and to perform analyses on very large datasets.
Can be used to easily graphically represent the data
Data is stored as nodes and relationships (arrows) while properties describe the stored data.

4.12 – Fundamentals of Functional Programming

- Function types
- First class objects
- Function Application
- Partial Application Function
- Composition of Functions
- Lists in Functional Programming

Programming Paradigms
Programming Paradigms are styles of programming - eg Procedural Programming, Object Oriented Programming, Declarative Languages and Functional Programming

What is a function
A function is a mapping from a set of inputs (**domain**) to a set of possible outputs, known as the **co-domain**
A function is a rule, that for each element in the domain, assigns a chosen value from the co-domain, without necessarily using every member of the co-domain.

First Class Objects
First class objects are objects which may:
1. Appear in expressions
2. Be assigned to a variable
3. Be assigned to an argument
4. Be returned in a function call

Function Application
Function application means a function applied to the arguments of the function application.

Partial Function Application
A partial function application takes advantage of decomposing multi-argument functions into smaller functions with fewer arguments.
Eg
Add :: int -> int -> int

Add x y = x + y

addSix :: int -> int
addSix = add 6

Higher-order functions
A function is higher-order if it takes a function as an argument or returns a function as a result or does both.

What's special about functional languages?

STATELESSNESS: the value of a variable can never change - they are immutable.
NO SIDE EFFECTS: The only thing a function can do is calculate something and return a result, and it is said to have no side effects. Function which is called twice with the same parameters will always return the same result. **(Referential Transparency)**

Composition of Function
Two functions can be combined to get a single function - this is referred to as functional composition.

Map
Map is a higher-order function which takes a list and the function to be applied to the elements of the list as inputs and returns a list made by applying the function to each element of the old list.

Filter
Filter is higher-order function which takes a condition and a list and it returns elements in the list which satisfy the condition.

Fold (REDUCE)
Fold function reduces a list of values to a single value by repeatedly applying a combining function to the list values.
Foldl nmeans start from the left while Foldr means to start from the right.

Lists in Functional Programming
A list is a concatenation of a head and a tail. (head is the first element of the list and tail is the remainder of the list) (Therefore the tail is list in itself effectively)

A list can be empty

A list can have a number of functions carried out on it:
1. Return head
2. Return tail
3. Test for empty list
4. Return length of list
5. Construct empty list
6. Prepend an item to list
7. Append item to list

4.13 – Systematic Approach to Problem Solving
1. **Analysis**

- **Before the problem is solved, it must be defined by the end user.**
- **Process may involve prototyping**
- **Requirements of system defined in specification**
- **Could cover**
  - **Data**
  - **Procedures**
  - **Future**
  - **Problems with existing system**
- **AGILE MODELLING**
  - **Agile approach may be adopted, as the stages of software development may not be completed in a linear sequence.**
  - **Some analysis completed, then some design and implementation and visa versa.**
  - **As opposed to waterfall approach.**
  - **Throughout the process, feedback is got from the end-user - iterative process during which changes are made for the next part of the system.**
  - **PROTOTYPE**
    - **Prototype built with user participation - check the UI**
    - **Relies upon keeping the model simple and getting rapid feedback from the users.**

2. **Design**
   - **Processing:** The algorithms and appropriate modular structure for the solution, specifying modules with clear documented interfaces.
   - **Data Structures:** How data will be held and how accessed.
   - **Output:** Content, format, sequence.
   - **Input:** Volume, frequency, documents used
   - **Security:** How the data is to be kept secure from accidental corruption
   - **Hardware:** Selected on configuration
3. **Implementation**
   - **Once the design has been agreed, the program can be coded and the entire system can be decided.**
   - **Can be create using an iterative agile based approach.**
4. **Testing**
   - **Testing carried out at each stage of the development process.**
   - **System needs to be tested in two ways**
     - **With respect to the implementation**
     - **Wrt to the user's requirements**
       - **ACCEPTANCE TESTING**
       - **Objectives are:**
         1. Confirm that the system delivered meets the original customer specifications.
         2. Find out whether any major changes in operating procedures will be needed,
         3. Test the system in the environment in which it will be run, with realistic amounts of data,
         4. UNIT TESTING -> MODULE TESTING -> SUB-SYSTEM TESTING -> SYSTEM TESTING -> ACCEPTANCE TESTING
5. **Evaluation**
   - **EFFECTIVENESS, USABILITY and MAINTAINABILITY**
   - Evaluated after a waiting period (3-6 months)
   - Focus on:

- Comparison of the system's actual performance with the anticipated performance objectives
- Assessment of each aspect of the system against preset criteria
- Errors made during development
- Unexpected benefits and problems.