Computer Systems

A computer system has both hardware and software.

Hardware are the physical components that make up a device or computer system. These include both the internal components (eg motherboard, CPU, RAM) and peripheral devices such as printers.

Software is the computer code, programs and algorithms that give instructions to the hardware to make it perform the desired task. Without the software the hardware will not get any instructions and it will not do anything.

Software Classification

Software is split into two types: application software and system software

Application software is a program designed to perform a specific task that the user interacts directly with (eg spreadsheets, web browser and word processor, disk defragmentation).

System software is concerned with the running of the computer. Its purpose is the control the computer hardware and manage the application software. (eg operating system, antivirus, backup tools, firewall)

The **operating system (OS)** is the most important piece of system software. The OS handles management of the processor, memory, input/output devices, applications and security.

- **Application management** - Application software does not need to concern itself with interaction and complexities of managing the hardware because this is dealt with by the operating system. Application software runs on top of operating system which is an intermediary and takes care of interaction with the hardware.

- **Processor resources** – Allows multiple applications to be run simultaneously by manages the processing time between applications and cores and switching processing between applications very quickly. Multiple applications will access the processor resources via a schedule that alternates process between applications. High priority applications will have more CPU time, but it means that lower priority applications will take longer to run.

- **Memory management** – Distributes memory resources between programs and manages transfer of data and instruction code in and out of memory. Ensures that each application does not use excessive memory.
- **Security** – Tools such as anti-virus software and firewalls help protect the computer from attack. In addition requirement for passwords and control of access rights
.
- **Input / Output devices** – OS controls interaction with input (eg keyboard) outputs (eg. Monitor) and storage (eg hard disk) using hardware drivers. Allows users to save files to the hard disk and print documents for instance.

Cloud Computing

- Can store data and files on a server elsewhere that can be accesses via the internet.
- Can use applications over the internet
- Can sync files so that all your devices see the same files
- Can share documents with others
- Can access your files anywhere if you have a good internet connection

Advantages of cloud computing
- Only pay for storage that you use
- Data and files available from anywhere in the world where there is an internet connection
- Data automatically backed up

Disadvantages of cloud computing
- Need a reliable network connection
- Files are hosted elsewhere so a security concern
- the most recent versions of software is often not available
- Transfer of data over the internet will slow down performance.

Advantages of local storage
- Files can be accessed even when there is no internet connection
- More secure as files to not need to be transferred over the network and the user has more control

Disadvantages of local storage
- Users need to organise their backup solutions
- Not so easy to share documents
- Can only access the files locally

Memory

Volatile memory (main memory) When the computer is turned off the contents of volatile memory is lost. When there is no power, volatile memory is erased.

Non-volatile memory (secondary storage) Even when here is no power, the data remain unchanged and can be accessed once again once power has been resumed. This allows you to store files for he long term.

ROM (Read Only Memory) Data can only be read from the device, and cannot the memory cannot be edited or deleted. ROM is only used for situations where you can be sure that updates will not be needed. The computer's BIOS (basic input output system) which controls the boot up sequence is stored on a ROM chip.

RAM (Random Access Memory) - When applications are executed they are loaded into RAM first. RAM is volatile.

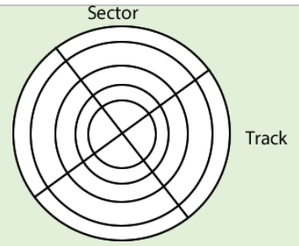
Embedded Systems

An embedded system is a computer system that is designed for a specific function, in contrast to a general-purpose computer that can carry out many tasks. Embedded systems typically have a minimal or no user interface. Thus, they can be optimised for size and power consumption, for instance. Examples of embedded systems include digital watches, MP3 players, washing machines, cars and mobile phones.

Secondary Storage

Secondary storage is necessary for saving files long and software including the operating system. Even when the computer is turned off, the data remain unchanged, and can be accessed again once the power supply has been turned on.

Magnetic Hard Disk
- Tracks on the disk platters contain tiny magnets, each holding 1 bit of data.
- The polarity (negative or positive) of the magnets determines whether the bits are 0 or 1.
- The write head modifies the polarity of the magnet as appropriate.
- The read head identifies whether each magnet is negative or positive.
- The tracks are laid out as a series of concentric rings.



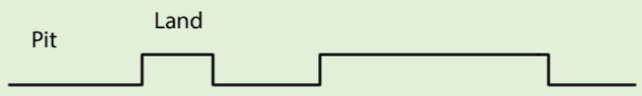
Advantages
- Cheap form of storage

Disadvantages
- Less reliable because it contains moving parts that can break
- Electromagnetic surge can corrupt the data held
- Slow speed of read/write access

Optical Disks
- Tracks on the disk contain pits and lands.
- The track is a spiral.
- A laser is emitted and the laser light is reflected when it hits the lands, but is scattered when it hits the pits.
- Depending on whether the light is scattered light is encoded as a binary value of 0 and reflected light is encoded as a 1.
- The sensor is able to detect light reflected, but not scattered.
- Example: Blue-Ray (25 Gb) DVD (4.7 Gb), CD (700 Mb).

Advantages
- Can transfer easily between computers

Disadvantages
- Can scratch easily
- Not much storage compared with other methods.
- No unlimited writes to the hard disk



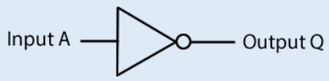
Solid state Drive
- Use millions of switches called floating gate transistors on microchips to store data.
- Electrons are stored in gates and this is encoded as 0 when there is an electron present and encoded a 1 when there is no electron present.
- The electros remain trapped even when there is no flow of electricity.
- Contain no moving parts and are therefore more robust that optical and magnetic storage.

Advantages
- Much faster that other means of storage
- More reliable than other means if you are only reading
- Quiet

Disadvantages
- More expensive per volume of storage
- Reliability might be an issue if you do a lot of writing

Boolean Logic

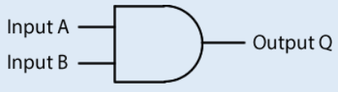
NOT gate - The output is the opposite of the input



NOT truth table

Input	Output
0	1
1	0

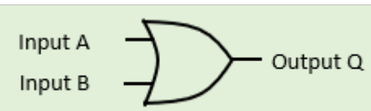
AND gate - has two inputs and will have a true output if the two inputs are true otherwise the output will be false



AND truth table

Input - A	Input - B	Output
0	0	0
1	0	0
0	1	0
1	1	1

OR gate - has two inputs and will have a true output if either or both the inputs are true



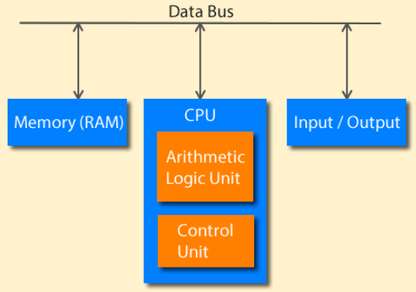
OR truth table

Input - A	Input - B	Output
0	0	0
1	0	1
0	1	1
1	1	1

System Architecture

CPU (Computer Processing Unit) or processor Fetches, decodes and executes instructions and performs logical and arithmetic operations.

Von Neumann architecture is the stored program concept, where program instructions and the data to be processed can be stored in the same memory.



Components of a CPU

Bus Wires through which data and instructions are transferred between computer components

Clock keeps all the CPU components synchronised

Arithmetic Logic Unit (ALU) Every operation takes place here. This is where the arithmetic (eg adding two binary numbers) and logic operations (eg checking to see if one number is bigger than another) take place.

Control Unit Decode the machine code instruction so that the ALU knows what to do with the instruction. Controls and monitors data transfer between different input and output hardware components

Factors affecting CPU performance

Clock speed is the number of cycles that a processor carries out per second. Each cycle of the CPU allows a single action (instruction) to be carried out. The greater the clock speed, the greater the number of operations and the faster the computer will run.

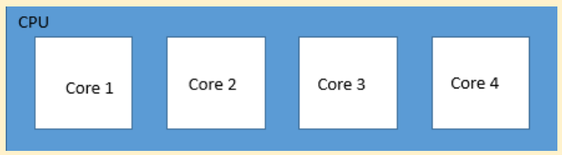
Number of processor cores A core is CPU in its own right. Nowadays most CPUs have multiple cores. Having multiple cores allows instructions to be carried out concurrently (at the same time), whereas a single core will only allow carry out instructions in serial (one at a time).

Latency Delay in transfer of data between components

Cache size Cache is a volatile memory store on the processor. Cache is much faster but smaller that RAM. Frequently used data and instructions within an application can be stored in cache instead of fetching from RAM which is quite slow. The bigger the cache the greater the volume of data and instructions that can be stored thereby reducing latency and improving performance of the CPU.

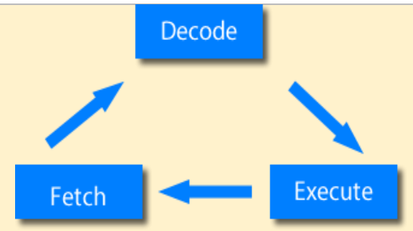
Cache type There are three levels of cache**.** Cache Level is a trade off between size and speed

- *Level 1 Cache* closest to the CPU and is the fastest cache (lowest latency), but does not have much capacity
- *Level 2 Cache* – is slower and further away from the CPU than L1 cache so latency is greater, but has more storage capacity.
- *Level 3 Cache* is the slower than L1 and L2 cache; much faster than RAM; has greater capacity than L1 and L2 cache.



Fetch execute cycle

1. Instructions are loaded into memory
2. Processor fetches the instruction from the main memory
3. Instruction is decoded so the CPU knows what to do with the instruction
4. Processor then executes the instruction
5. Result of the instruction can be stored in memory
6. Next instruction is then fetched from main memory and the cycle repeats itself.



Classification of programming languages

High level programming languages are closer to human language and is therefore easier to understand. A translator is used to convert the instructions into code that the computer understand. High level languages allow programs to be written that is independent of the type of computer. High level programming languages allow code to be written that is independent of the type of computer system. It is up to the compiler to translate the code into the right machine code for a particular code. There is a huge variety of high level programming languages, and the choice depends on the application.

Low level programming languages refer to machine code and assembly language. The Low level refers to low level of abstraction. The low level language is close to the language understood by the computer where operations map to the instruction in the processor instruction set. However it is difficult for humans to understand. Low level languages are appropriate for developing new operating systems, embedded systems and hardware device drivers

Machine code is expressed in binary values 0 and 1. This is the language that computers understand. All codes whether assembler or high level programming languages need to be translated into machine code. Machine code is specific to a processor.
Machine code instructions are made up of two parts the operator and the operand. The processor decodes the operator to identify the task that is to be carried out (eg. Add, load). The operand is the value or memory address that that instruction is to be operated on

Machine code instruction	
Operator	Operand
0011	10010100

Assembly language provides basic computer instructions for programs to run. There is a one to one relationship between machine code and assembly code instructions. One assembly language instruction maps to one machine code instruction, thus the structure of assembly language and machine code is the same, but where machine codes uses 0 and 1 which are very difficult for programmer to understand, assembly language uses mnemonics which is easier for the programmer.

Assembly language sample Instruction set

```
LOAD #23 # Load from RAM to processor
MOV a 23 # Transfer in number 23 into the variable a
ADD 2 3  # Add 2 values
STORE    # store data in RAM
```

Each type of processor has its own instruction set and therefore its own assembly language and machine code. So Assembly code written for one type of processor will not run on another.

Low level languages versus high level languages

	Advantages	**Disadvantages**
Low level	Produce code that is faster and better optimised than high level languages. Appropriate for developing new operating systems, embedded systems and hardware device drivers	Difficult to understand and modify Assembly code is written for a specific processor architecture, and so is not portable to other computer architectures
High level	High level programming languages allow code to be written that is more portable. Thus code can be run on different of the types of computer system with different processor architecture. Easier to understand Easier to modify	Needs a translator run slower because of the layers of abstraction and there is inefficiency in translator.

Program translators allow programs to be translated into machine code so the than programs can be run on a computer.

Interpreter converts high level languages into machine code one instruction at a time on-the-fly while the program is running. Each instruction is converted to machine code once the previous instruction has been executed. Interpreters are good for debugging code because the program stops as soon as the error has been found. However running code this way is much slower running compiled code. The machine code is not saved.

Compiler A program that converts high level languages into machine code before the program is run. A compiler saves the machine code, so the source code is no longer needed A compiler allows a program

to be run faster than interpreted code. Software is normally distributed as compiled machine code. For proprietary software this is good because other people cannot copy the code and use it for their own applications.

Assembler Assembler converts assembly language instructions into machine code.