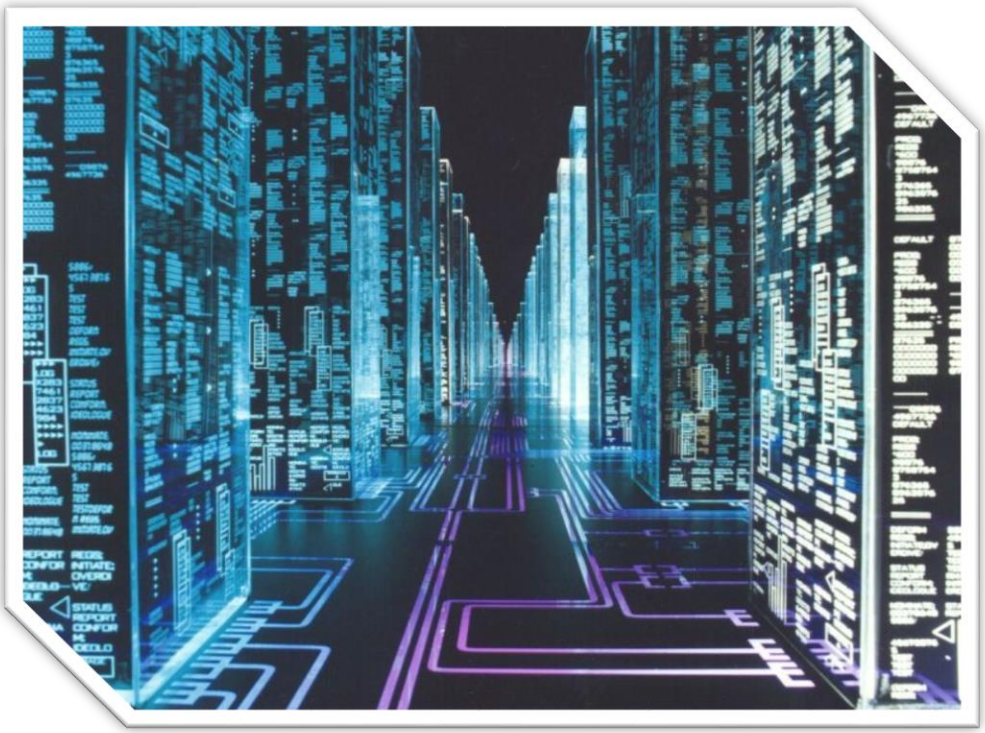
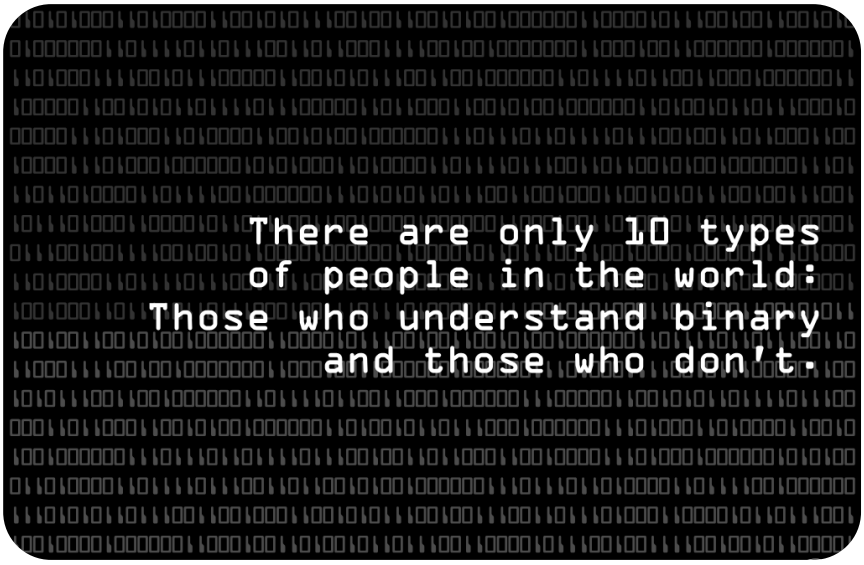
COMPUTER ARCHITECTURE



NAME _____

Data Representation



Computers are used to store a variety of information including numbers, text, graphics and even sound. Regardless of the type of information represented, it is **all stored as bit patterns made up from the digits 1 or 0**. In other words everything that is stored on the computer is eventually broken down into its simplest form, which is a pattern of 1s and 0s.

All of the data and programs that are used by a computer are represented as bits within the main memory. The storage of these bits is made more manageable by **grouping them together in multiples of eight**.

Advantages of Using Binary Numbers

- Binary is a simple two-state system (1 or 0) which is **ideal when representing a two state system of power on/power off**
- There are only a **few rules for addition**, making calculations simpler.
- A **degraded signal can still be detected** as representing 1

Storing Positive Integers

We count in decimal numbers such as 0,1,2,3,4,5,6,7,8,9. Computers use binary numbers such as 0 or 1. **We count in base 10 where as a computer counts in base 2**. We will look at how a computer stores positive integers using a simple table to help you.

This table shows the decimal equivalent to 2^n

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

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

64 + 8 + 4 = 76

So the number 76 is represented in binary as 01001100

Storing Negative Numbers (Twos Complement)

We use twos complement to store negative numbers in a computer. In this course you only need to know 8-bit twos complement. So how does it work?

Lets start with the number we want to represent, -10.

Represent 10 in binary.

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

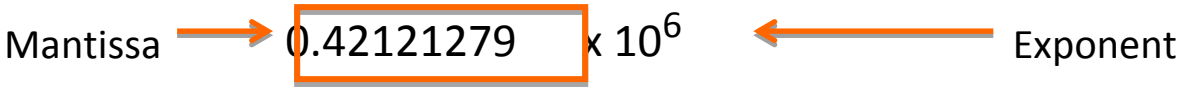
-128 + 64 + 32 + 16 + 4 + 2 = -10

Storing Real Numbers

Having found a method of representing positive whole numbers we now have to consider **how to represent very large and very small numbers**. If we used conventional binary methods then too much memory would be used just to represent numbers.

The technique used to solve this problem is similar to standard form, which you are taught in mathematics, and it is called **floating-point representation**.

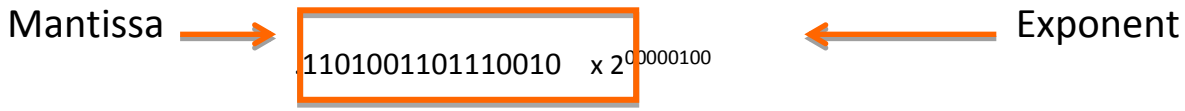
In standard form you are taught to write the number 421 212.79 as



The rule is to place the decimal point just after the first digit and to count the number of places that it has been moved. This number is then written as the power. In this case the point was moved five places.

Now that we have looked at how floating-point representation is used with decimal numbers lets look at how it is used with binary. Binary has a binary point just like a decimal point so we do the same as before with moving the binary to after the binary point.

So the binary number: 1101.001101110010 is written as



Notice that we have **moved the binary point four places** but the **exponent is written as 00000100**. This is not one hundred; it is the number 4 in binary.

In the example above we have allocated 2 bytes for the mantissa and 1 byte for the exponent. Computers more commonly allow 4 bytes for the mantissa and at least 1 byte for the exponent.

The computer **only needs to store the value of the mantissa and the exponent** to represent any real number. Floating point is easy to implement and **saves storage space**.

Accuracy & Range of Floating Point Numbers

- Increasing the mantissa increases the accuracy of the numbers represented.
- Increasing the exponent increases the range of numbers that can be represented.

Storing Characters

When you are using a program and you press a key on the keyboard **the program has to have some way of identifying which key you pressed**. This is true for any program whether it is a word processing package, spreadsheet or game. **Each character on the keyboard has a unique binary code** allocated to it.

This is called **ASCII**. It stands for **American Standard Code for Information Interchange**. **ASCII is a 7-bit code that represents 128 code values**.

ASCII code includes :

- Non-printing characters: <return>, <tab> (control characters)
- Numbers: 0-9
- Upper and Lower Case Letters: A-Z, a-z
- Punctuation and other symbols: $, %, !, ?, @

All of the above are examples of the **character set**. This is the group of letters and numbers and characters that a computer can represent and manipulate.

Most ASCII characters are either displayed on the screen or can be printed on a printer but there are some that serve a different purpose. **Control characters** include keys such as RETURN, TAB and DELETE. **They are the first 32 characters in ASCII**. These are used to send a control signal to a printer e.g. BACKSPACE or NEW LINE. Sometimes control characters are referred to as '**non-printable characters**'.

ASCII Table

Below is an ASCII table. The table shows the binary representation for each character.

ASCII Code: Character to Binary			
0	0011 0000	**L**	0100 1100
1	0011 0001	**M**	0100 1101
2	0011 0010	**N**	0100 1110
3	0011 0011	**O**	0100 1111
4	0011 0100	**P**	0101 0000
5	0011 0101	**Q**	0101 0001
6	0011 0110	**R**	0101 0010
7	0011 0111	**S**	0101 0011
8	0011 0111	**T**	0101 0100
9	0011 1001	**U**	0101 0101
A	0100 0001	**V**	0101 0110
B	0100 0010	**X**	0101 1000
C	0100 0011	**Y**	0101 1001
D	0100 0100	**Z**	0101 1010
E	0100 0101	**.**	0010 1110
F	0100 0110	**,**	0010 0111
G	0100 0111	**?**	0011 1111
H	0100 1000	**!**	0010 0001
I	0100 1001	**(**	0010 1000
J	0100 1010	**)**	0010 1001
K	0100 1011	**SPACE**	0010 0000

Extended ASCII

Extended ASCII is an 8-bit code that can represent 256 characters. Many systems use this.

Unicode

With an increase in worldwide communication and the need to represent different languages symbols a **16 bit character code (65, 536 characters) called Unicode** is used. This **represents foreign languages such as Japanese or Arabic**. **Unicode files sizes are larger** since it takes 2 bytes to store each character compared to 1 byte with ASCII.

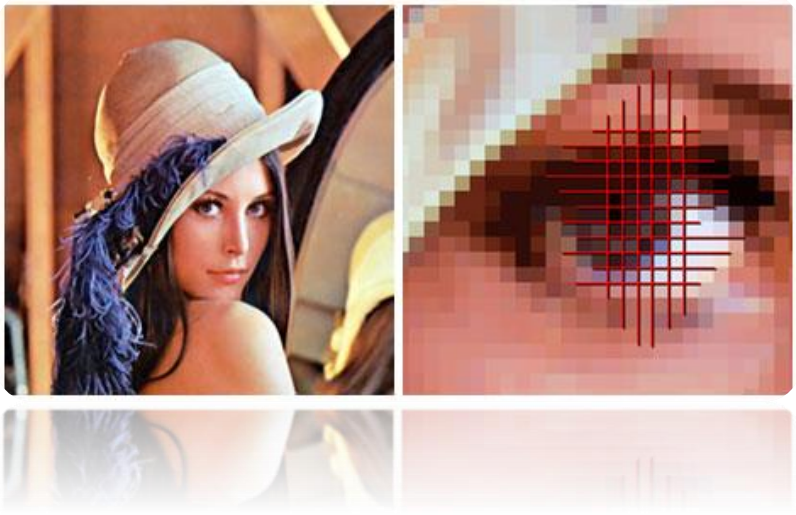
Character Representation

Text Representation	Bits	Amount of Characters
ASCII	7 Bit	128
Extended ASCII	8 Bit	256
Unicode	16 Bit	65536

Storing Graphics

A pixel is the most basic component of any computer graphic. Pixel stands for picture element. It corresponds to the smallest element that can be drawn on a computer screen. **Every computer graphic is made up of a grid of pixels**. When these pixels are painted onto the screen, they form an image.

The picture shows when zoomed into the woman's eye you can see lots of little blocks. These are pixels. Each pixel stores a separate colour. Today's digital cameras take high-resolution photographs that store thousands of pixels, which create a clear image.



Storing Black and White Bitmap Graphics

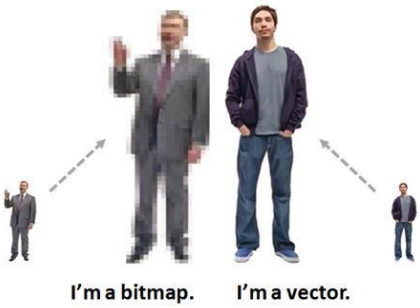
The above graphic is a really simple black and white image saying "HI". This image is displayed in a 8 x 8 grid table with **each box represents a pixel**.

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

In black and white, **each pixel can be represented by 1 bit**: 1 if the pixel is black or 0 if the pixel is white. The computer represents the image in memory as a file of 0s and 1s. The computer opens this file then starts looking for numbers that describe image information. Every time it comes to a 0 it draws a white pixel. When it comes to a 1 it draws a black pixel. **The file is known as a bit map.** Paint is an example of a bit map graphics package.

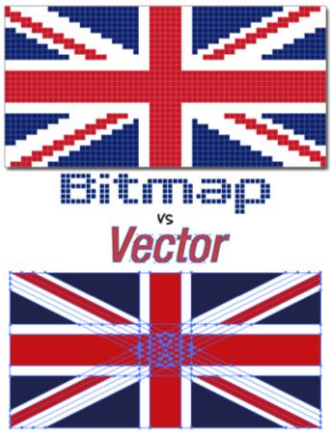
In the 8 x 8 bit-mapped grid above each pixel requires 1 bit of storage. **There are 64 pixels so this means the image needs 64 bits or 8 bytes of storage (8 bits = 1 byte)**. Graphics tend to be much larger than this simple example.

Storing Vector Graphics



In a program such as Serif or Photoshop the computer **stores information about an object by its attributes** i.e. a description of how it is to be drawn. For a rectangle these attributes might be: **start x and y position, length, breadth and angle of rotation thickness and colour of the lines, colour fill etc.**

This means that the rectangle can be **selected at any later time and altered** by changing its length, dragging it to a new position etc. It is not possible to change the colour of any individual part of the rectangle though it is possible to change the colour of the lines forming the rectangle and the interior fill.



Though the image on the screen is still stored as a bit-map**, the drawing package stores the attributes for each object** (rectangle, line, circle, ellipse, text etc.) that is drawn. When the drawing is saved, only the list of objects and their attributes is stored which **greatly reduces the file size**. When the drawing is loaded the drawing package redraws all the objects. This means that **if you increase the resolution of the screen the object will remain clear and crisp.**

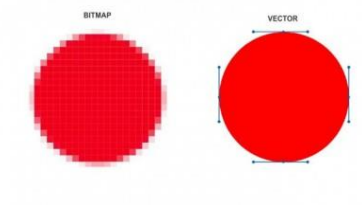
Storing Bitmap Graphics

You have already had a look at storing bitmap graphics. These are made up of tiny dots called pixels. **Paint is an example of a bitmap graphics package**.

On a bitmap graphic **each individual pixel is stored**.

Advantages of bit-mapped graphics:

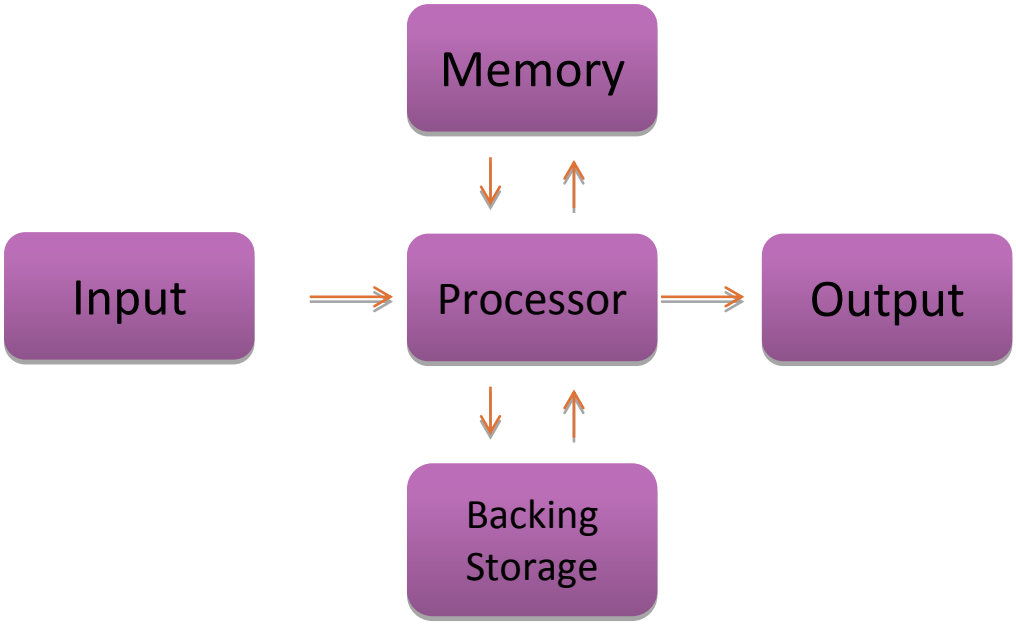
- You are able to edit at pixel level. This means that you can zoom right into each pixel and edit it.
- Storing a bitmap graphic will take the same amount of storage space no matter how complex you make it.



Disadvantages of bit-mapped graphics:

- They can demand a lot of storage space, particularly when using lots of colours.
- When using a bitmap package you cannot edit an individual object, if you use Paint on the computer you will be familiar with this.
- When you re-size a bitmap graphic it becomes pixelated and jagged.

Computer Architecture



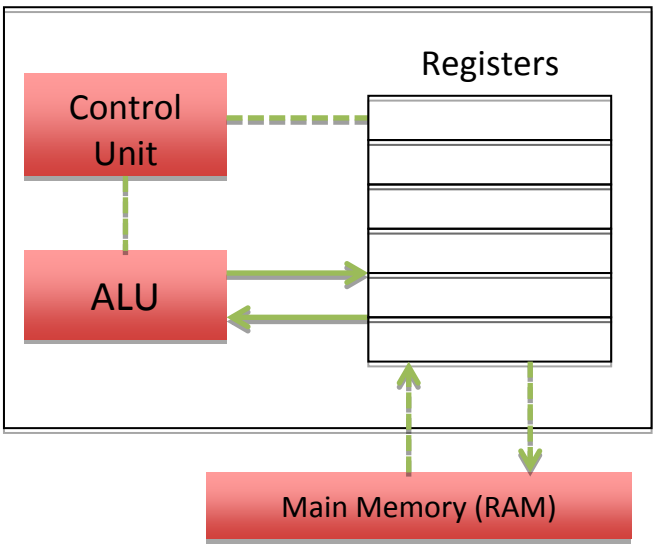
This is simple representation of how a computer works.

The Processor (CPU)

The processor is the **brains of the computer** and **deals with all the movement of data and any calculations to be carried out**. Computers can carry out instructions very quickly because the CPU can process billions of instructions every second although it only does one at a time.

The processor is made up from:

- The Control Unit (CU)
- The Arithmetic and Logic Unit (ALU)
- Registers



The Control Unit (CU)

The job of the control unit is to **fetch instructions from the main memory. It will then try to understand these instructions and carry them out.**

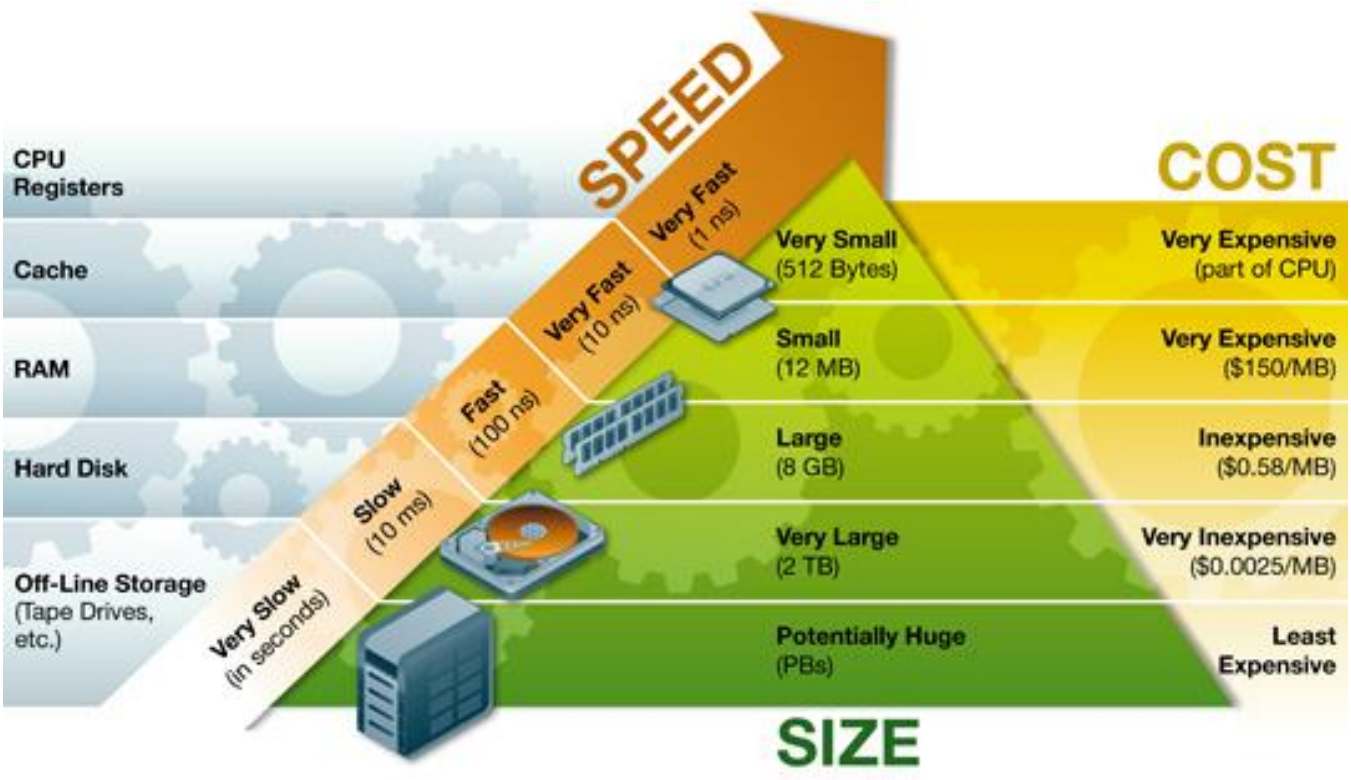
To simplify the control unit is **responsible for running programs that are loaded into main memory.**

The Arithmetic and Logic Unit (ALU)

- Carries out **computers arithmetical functions such as addition, subtraction, multiplication** etc
- Carries out the computers logical functions such as comparing values using IF, AND, >, <, WHILE

Registers

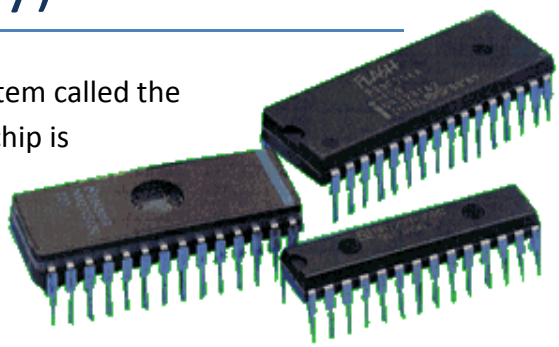
- Registers are **small temporary memory locations** located on the processor.
- They are used to **store the data for the current instruction** being processed



Main Memory (RAM and ROM)

People often get confused between main memory and backing storage, they are **not** the same. **Main memory is located inside the computer system**. It can either be RAM or ROM. **Backing storage is outside the main processor**, e.g. Hard Drives, CD/DVD drives, USB Flash Memory (Pen Drives). Main memory in today's computers is on average around 4-6 Gb of RAM. Backing Storage is much bigger with average computers having around 500 Gb or more.

ROM (Read Only Memory)



ROM is used to store a small part of the operating system called the bootstrap loader. This is loaded onto ROM when the chip is manufactured.

- Data is **stored permanently in ROM**,
- Data is **not lost when the power goes off**
- Data in ROM cannot be changed

RAM (Random Access Memory)

This is where the operating system is stored; it also **holds all programs and data.** You can purchase additional RAM chips and install them in your desktop computer, which normally speeds up multi-tasking.

- The processor **can write to and read from RAM** at high speed
- Data held in RAM can be changed
- **All data in RAM is lost when the power is switched off (RAM is volatile)**

Cache Memory

Cache memory is an area of **fast access memory**. It is located either on the processor chip or relatively close to it. It is normally a **small amount of memory for example up to 6 Megabytes.** This is used to **temporarily store data and instructions that are used frequently**.

Since the cache stores frequently used data and instructions it **saves the computer reading from main memory, which is much slower.**

Backing Storage

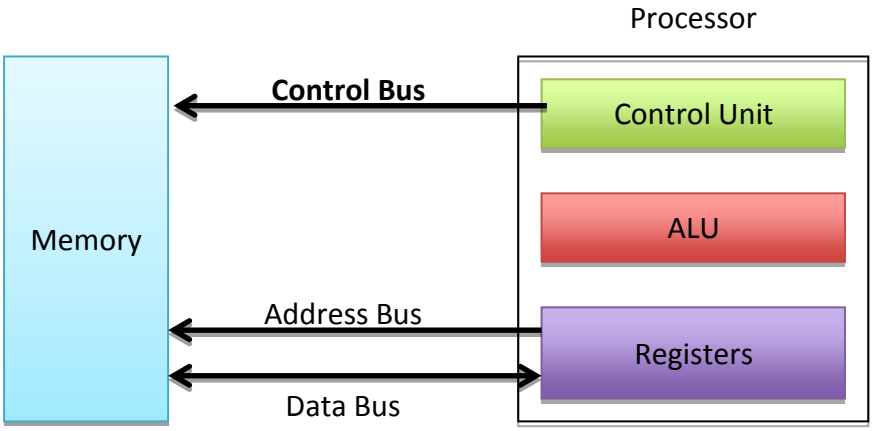
This is another form of memory and is used to store data such as music, movies, documents and software. **Backing storage is the slowest form of memory**.

All of your files on a computer are stored in backing storage and when the power is switched off you do not lose these, unlike RAM that gets wiped when the power is turned off.

Types of Memory

Type of Memory	Function	Speed
Registers	Small temporary memory locations used to store the data for the current instruction being processed or to store the address of a memory location.	Very fast access time
Cache	Temporarily store data and instructions that are used frequently	Slower than registers but still very fast
Main Memory	Stores running programs in RAM and system software such as the bootstrap loader in ROM	This is fast but not as fast as registers or cache
Backing Storage	Permanently stores user data	Slowest of all the types of memory

Buses



The Processor (CPU) has buses. These are multiple lines that connect the processor and main memory and used to transfer data and send signals between them.

Address Bus

Address Bus is used to **specify the address of the memory location that is to be read from or written to**. The bus is **uni-directional** (one way). The address bus is made up of parallel wires each carrying a single bit. **The size of the address bus will determine how many memory locations can be directly accessed**

$2^{\text{width of address}}$ = Number of Unique addresses possible

Modern computers will typically have an address bus 32 lines wide although 64-bit address buses are now becoming normal in everyday computers. **By increasing the width of the address bus it will increase the total number of memory locations that a processor can address.**

Data Bus

This bus is used to **transfer data between main memory and the processor**. It is **bi-directional** (two way) since data can be transferred from a memory location and vice versa.

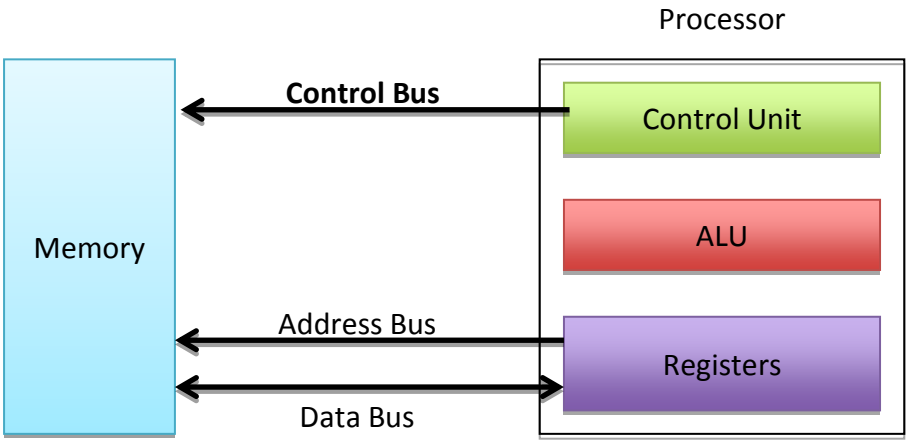
The amount of wires in the in the data bus will determine how much data can be transferred from main memory and the processor**. Increasing the width of the data bus would increase the performance of the computer system**.

Control Bus

The control bus is made up of individual lines with specific functions giving instruction to the rest of the system from the control unit:

- **Read:** used to initiate a memory read operation which reads the contents of a memory location into the processor
- **Write:** used to initiate a memory write operation which writes an item of data from the processor into a memory location
- **Clock**: sends a series of pulses into the processor to synchronize events. The time interval between pulses is called a clock cycle.
- **Reset:** causes the computer to stop the current program and then reboot
- **Interrupt:** peripheral devices such as printers can send a signal on the interrupt line into the processor when they require attention.
- **NMI (Non-Maskable Interrupt):** requires serious attention such as a power failure and cannot be ignored

Fetch Execute Cycle



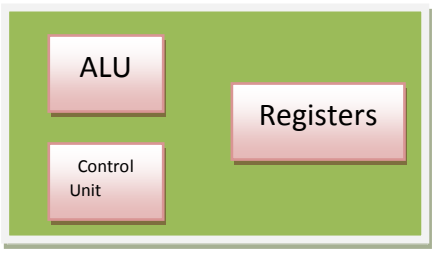
This is the process where an instruction is retrieved from memory, decoded and then carried out.

1. The processor sets up the address bus with required memory address.
2. The processor activates the read line on the control bus.
3. An instructions is fetched from the memory location using the data bus and stored in the instruction register
4. The instruction it is decoded and executed.

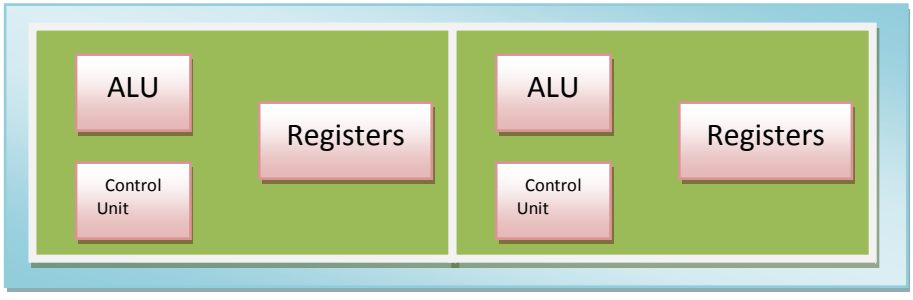
Factors Affecting System Performance

Number of Processors (Cores)

A core is a complete processing unit within the CPU, it has an ALU, control unit and registers.



Originally, there was only one processing core in a CPU, but as time moved on the processing core could be made even smaller, and around 2006 it became possible to put two cores inside a single CPU chip.



Now it is common for a CPU chip to have two, four, eight cores or more. A CPU with two cores is called 'dual' core and one with four cores is called 'quad core'. Each core can have its own internal cache to improve performance as well as shared cache within the CPU. Increasing the number of cores improves performance in two ways.

Multi-Tasking
Multi-tasking is the ability to carry out more than one task at the same time. Clearly, with two cores, a CPU can run two tasks simultaneously. A quad core can handle four tasks. So for example, one core could be running a photo editing application whilst another is handling a word processing application.

Parallel Processing
Parallel processing is when a single task (program) is split into two or more parts and each part is processed at the same time. In theory this would double performance on a dual core CPU as each part is processed independently.

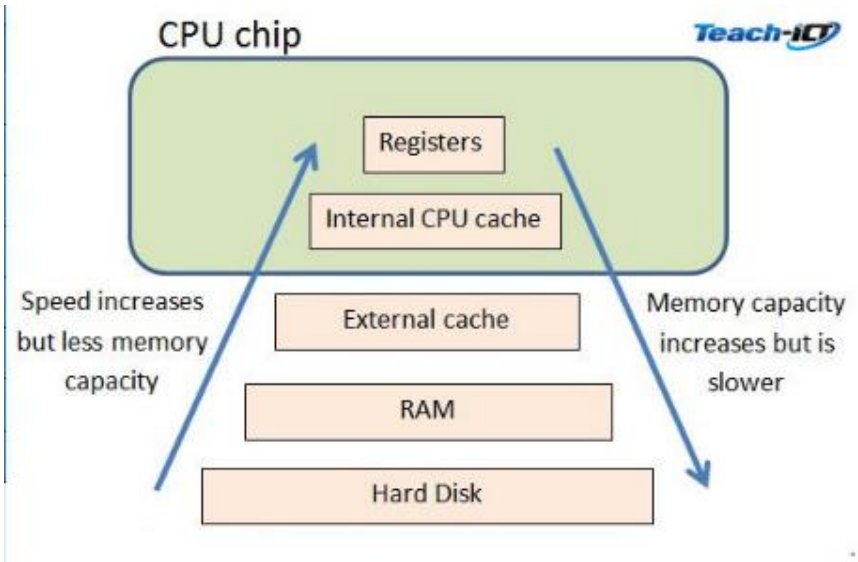
However, you do not always get twice the performance because many programs cannot be split neatly into two independent parts. This is because they are mostly sequential i.e. each task depends on the outcome of a previous task and so it has to wait for the first task to be completed.

An example of when parallel processing is possible is a task to update all the pixels on the screen. The colour of one pixel does not tend to depend on another pixel.

Cache

In an ideal world, when the CPU needs data, it should be available instantly so as not to slow down the processing operation.

A cache is a small, but extremely fast type of RAM, located inside the CPU chip itself.
Having a larger cache will speed up processing because the CPU can access data and instructions faster than it can from RAM.



However, cache memory is more expensive than standard RAM so it is a balance between performance and cost.

Cache often consists of high-speed SRAM i.e. Static Random Access Memory whilst main memory makes use of cheaper, slower DRAM i.e. Dynamic Random Access Memory.

Data and programs currently in use are stored in RAM. But RAM is comparatively slow to access when compared to the speed at which the registers work.

So to help speed up the processing time, cache memory is used to store instructions or data that are either frequently used, have recently been used or are about to be used. This means that they don't have to be fetched directly from RAM.

There are different levels of cache. Level 1 (L1) cache is the fastest but the smallest size and resides inside the CPU chip. Level 2 (L2) cache is a bit slower but is larger, and so can hold more data, then Level 3 (L3) cache is slower still but can hold the most data. Level 2 and Level 3 caches can be found either inside or outside the CPU, depending on the motherboard design. A cache though is always faster than the main RAM.