# COMPUTER ARCHITECTURE

# Outcomes and Assessment Standards

## Outcome 1

**1 Explain how programs work, drawing on an understanding of advanced concepts in software development and computer architecture by:**

1.1 Reading and explaining code
1.2 Describing the purpose of a range of programming constructs and how they work
1.3 Describing how a range of standard algorithms work
1.4 Describing how programs relate to low-level structures and operations

## Outcome 2

**2 Develop modular programs using one or more software development environments by:**

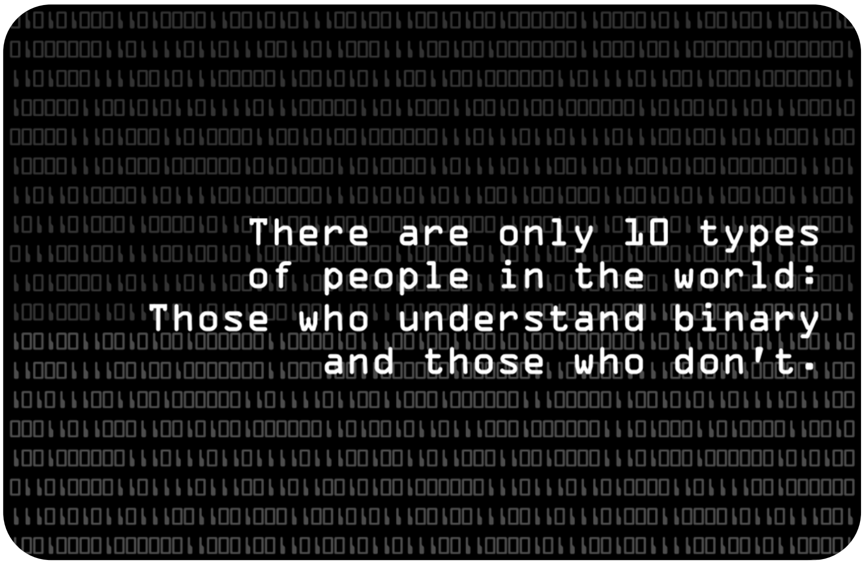
2.1 Applying contemporary design and development methodologies
2.2 Selecting and using combinations of appropriate constructs
2.3 Selecting and using appropriate simple and structured data types, including 1-D arrays
2.4 Testing digital solutions systematically
2.5 Identifying and rectifying program errors
2.6 Applying aspects of good programming technique — meaningful variable names, internal commentary, indentation

## Outcome 3

**3 Produce a detailed report on the impact of contemporary computing technologies, by analysing and evaluating:**

3.1 Current trends in software development languages and environments
3.2 Current trends in the development of intelligent systems
3.3 Current trends in online systems

# 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 |

Now that we know the decimal equivalent to the base two system we can use a simple table to help convert a number into binary.

**For example lets convert the number 76 to binary**. All we need to do is **start at the left of the table and see if we can take away the number without going below 0.**

- Can we take 128 away from 76? No. So we put a 0 inside that box.
- Can we take 64 away from 76? **Yes**. So we put a 1 in there. We now have 12 left (76-64=12)
- Can we take 32 away from 12? No. So we put a 0 in there.
- Can we take 16 away from 12? No. So we put a 0 in there.
- Can we take 8 away from 12? **Yes**. So we put a 1 in there. We now have 4 left (12 - 8 = 4)
- Can we take 4 away from 4? **Yes**. So we put a 1 in there. We now have 0 left (4 – 4 = 0)
- As we are now at 0 we fill the remaining boxes in with 0.

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

**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?

Note that in binary adding 1 and 1 together gives 0 and you will carry the 1 onto the next column to the left.

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

Represent 10 in binary.

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

0000 1010 = 10

Next thing is to change all the 0s to 1s and vice versa. Then add 1.

| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | |
|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 0 | 1 | 0 | 1 | 0 | = 10 |
| **1** | **1** | **1** | **1** | **0** | **1** | **0** | **1** | Change all the 0s to 1s and vice versa |
| | | | | | | **+** | **1** | |
| **1** | **1** | **1** | **1** | **0** | **1** | **1** | **0** | = -10 |

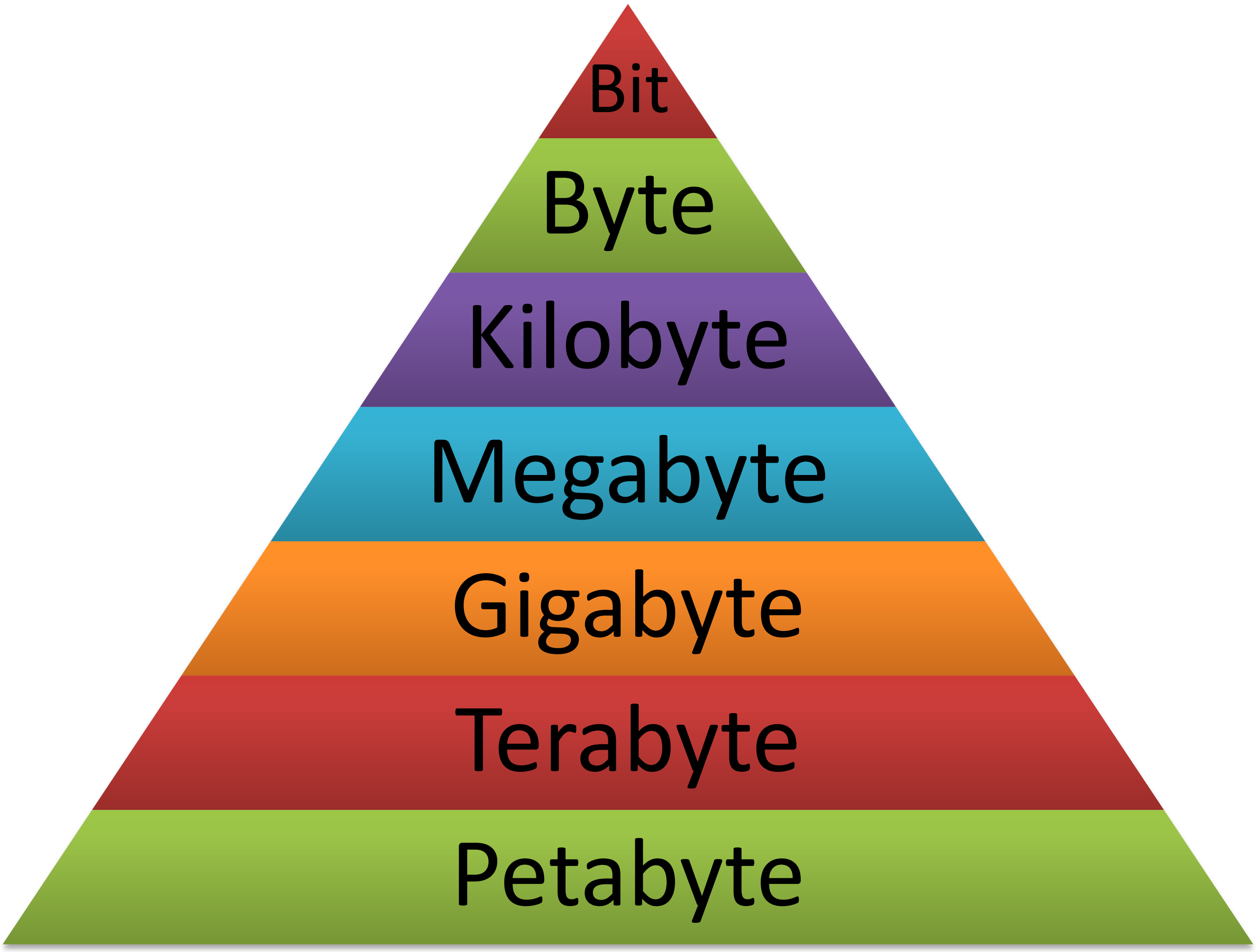
Lets do another one. This time we want to represent – 37.

| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | |
|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 1 | 0 | 0 | 1 | 0 | 1 | = 37 |
| **1** | **1** | **0** | **1** | **1** | **0** | **1** | **0** | Change all the 0s to 1s and vice versa |
| | | | | | | **+** | **1** | |
| **1** | **1** | **0** | **1** | **1** | **0** | **1** | **1** | = -37 |

# Units of Storage

We use these terms to measure a computers memory.

| | |
|---|---|
| 1 Bit | **Binary digit: a single 1 or 0** |
| 1 Byte | 8 Bits |
| 1 Kilobyte | 1024 bytes |
| 1 Megabyte | 1024 Kilobytes |
| 1 Gigabyte | 1024 Megabytes |
| 1 Terabyte | 1024 Gigabytes |
| 1 Petabyte | 1024 Terabytes |



# 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

Mantissa → $4.2121279 \times 10^{5}$ ← Exponent

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

Mantissa → $.1101001101110010 \times 2^{00000100}$ ← Exponent

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

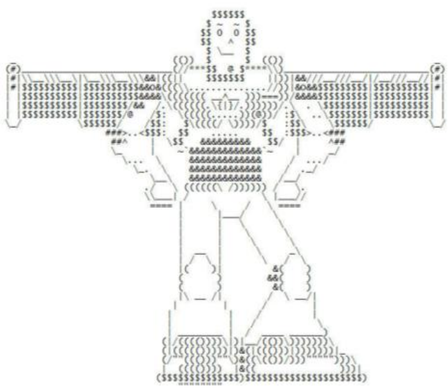
When you are using a program and you press a key on the keyboard **the program has to have some way of identifying which key you pressed**. This is true for any program whether it is a word processing package, spreadsheet or game. **Each character on the keyboard has a unique binary code** allocated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



### Control Characters

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.

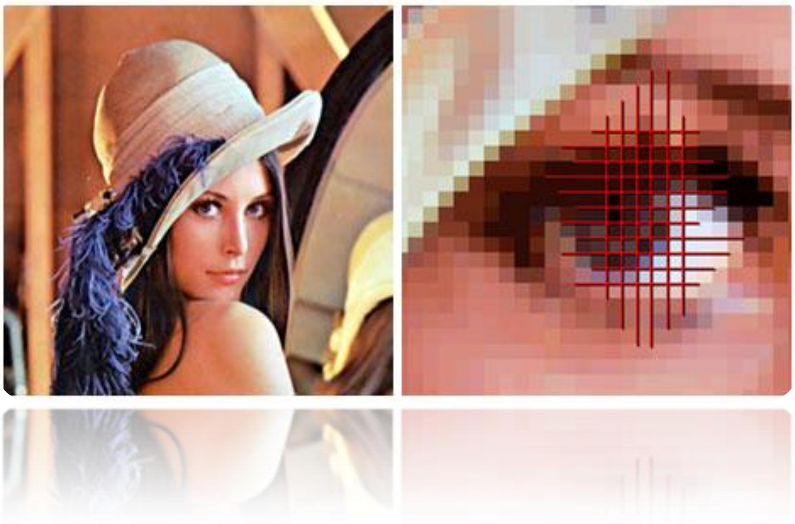
# Character Representation

| 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

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 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 |

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**.

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.

# Calculating File Size


The black and white image of Scotland to the left is 3 inches by 6 inches at 300 pixels per inch.

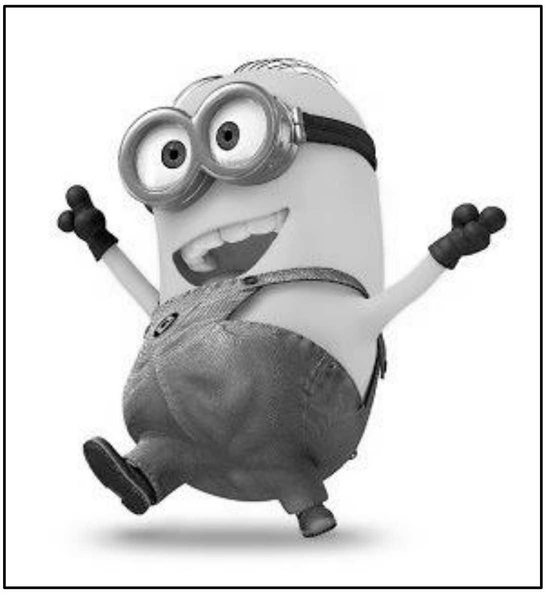
Calculate the file size of the graphic.

1. 3 (inches) x 300 (pixels) = 900 pixels
2. 6 (inches) x 300 (pixels) = 1800 pixels
3. 900 x 1800 (pixels) = 1,620,000 pixels (bits)
4. 1,620,000 / 8 = 202,500 Bytes
5. 202,500 / 1024 = 197.7 kilobytes (kB)
6. 197.7… / 1024 = 0.19 megabytes (MB)

The black and white image of a minion to the right is 5 inches by 6 inches at 400 pixels per inch.

Calculate the file size of the graphic.

1. 5 (inches) x 400 (pixels) = 2000 pixels
2. 6 (inches) x 400 (pixels) = 2400 pixels
3. 2000 x 2400 (pixels) = 4,800,000 pixels (bits)
4. 4,800,000 / 8 = 600,000 Bytes
5. 600,000 / 1024 = 585.93… kilobytes (kB)
6. 585.93… / 1024 = 0.57 megabytes (MB)



# Calculating Colour Image File Size

The colour image to below can be calculated much the same way as a black and white image. The **only difference is that you need to multiply by the colour depth**.

The **higher the colour depth = more colours available**. So if the image has a colour depth of 16bits then there are 65536 colours available ($2^{16}$ = 65536)

| Colour Depth | Amount of Colours |
|---|---|
| $2^0$ | 1 |
| $2^1$ | 2 |
| $2^2$ | 4 |
| $2^3$ | 8 |
| $2^4$ | 16 |
| $2^5$ | 32 |
| $2^6$ | 64 |
| $2^7$ | 128 |
| $2^8$ | 256 |
| $2^9$ | 512 |
| $2^{10}$ | 1024 |
| $2^{16}$ | 65,536 |
| $2^{24}$ | 16,777,216 |
| **24 Bit Colour Depth = True Colour** | |

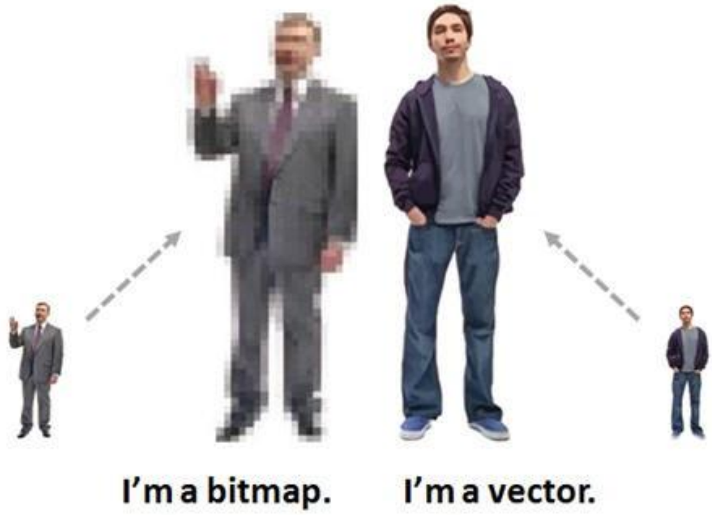
6 inches by 5 inches at 300dpi (dots per inch) with a colour depth of 16.


Calculate the file size of the image:

1. 6 (inches) x 300 (dpi) = 1800 pixels
2. 5 (inches) x 300 (dpi) = 1500 pixels
3. 1800 x 1500 (pixels) **x 16 (colour depth)** = 43,200,000 pixels (bits)
4. 43,200,000 / 8 = 5,400,000 bytes
5. 5,400,000 / 1024 = 5,273.43.. kilobytes (kB)
6. 5,273.43.. / 1024 = 5.15 megabytes (MB)

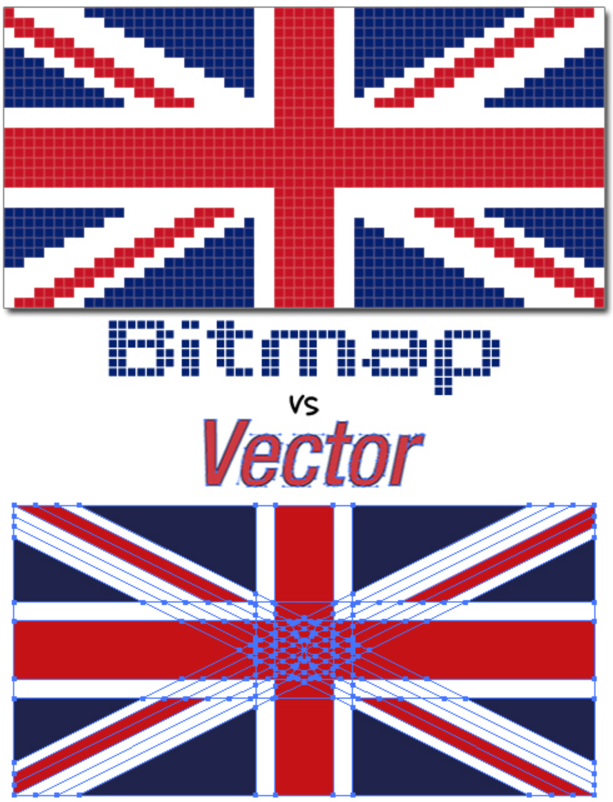
# Storing Vector Graphics

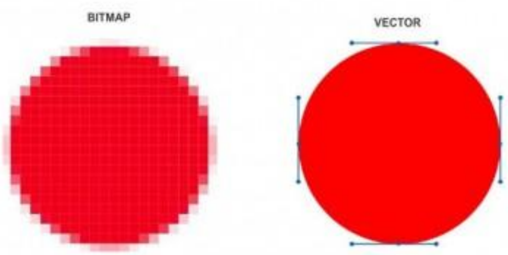
In a program such as Serif or Photoshop the computer **stores information about an object by its attributes** i.e. a description of how it is to be drawn. For a rectangle these attributes might be: **start x and y position, length, breadth and angle of rotation thickness and colour of the lines, colour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

Some of the disadvantages are that 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. Another disadvantage is that **when you re-size a bitmap graphic it becomes pixelated and jagged**.

# Storing Sound

The size of an audio file depends on the following factors:

- The length of time the audio file is
- The sampling frequency
- The number of channels
- The sample depth

The length of time the audio file is has to be in seconds. So for example a **3 minute long clip will be 180 seconds**. If you are struggling with this then multiply the minutes by 60 to convert it into seconds.

The sampling frequency (**sometimes called the sampling rate**) is how many times per second the sound is sampled. This is measured in KHz. **For example 44.1 KHz. To be able to use this in the formulae we have to covert it to 44 100**. If you are struggling with this then multiply the sampling frequency by $10^3$.

The number of channels will depend on how the sound was recorded. If there was **only one channel then this is called mono**. If it were recorded in **2 channels it would be stereo**.

The sample depth is how many bits are used to store each sample. For example this could be 8 bits. **You do not have to convert this to anything**.

### Formula

Audio file size = [time in seconds] x [sampling frequency] x [number of channels] x [sampling depth]

### Example

Calculate the file size of a two-minute audio file with a sampling frequency of 44.1KHz. The audio was recorded in stereo and has a sampling depth of 8 bits.

Audio file size = [time in seconds] x [sampling frequency] x [number of channels] x [sampling depth]
Audio file size = [120] x [44100] x [2] x [8]
Audio file size = 84,672,000 bits / 8
Audio file size = 10,584,000 bytes / 1024
Audio file size = 10,335.93 kilobytes… / 1024
Audio file size = 10.1 megabytes

## Storing Video

The size of a video file depends on the following factors:

- The length of time the video file is
- The frame rate
- The bit depth
- The number of pixels in one frame

The length of time the video file is has to be in seconds. So for example a **2 minute long clip will be 120 seconds**. If you are struggling with this then multiply the minutes by 60 to convert it into seconds.

The frame rate is how **many frames are stored per second**. For example 30 frames per second. This may be **represented by using FPS** (frames per second).

The bit depth is used to tell you **how many colours are in each individual frame**. This links back to when you looked at storing colour graphics. For example 16-bit colour depth.

The number of pixels in each frame will tell you the **resolution of each frame**. For example this could be something like 640 x 480.

## Formula

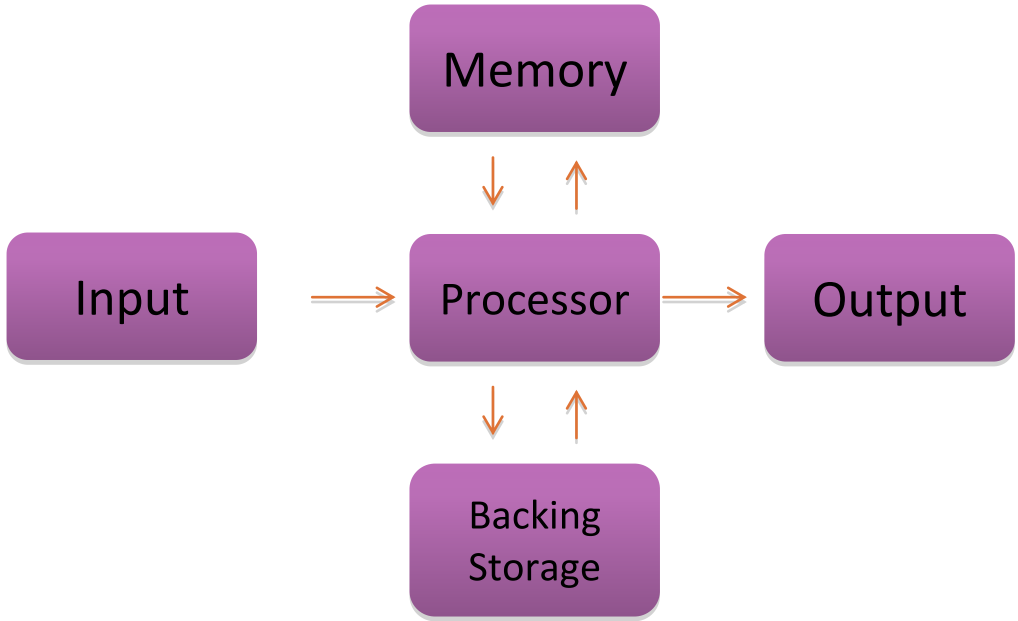
Video file size = [time in seconds] x [frame rate FPS] x [bit depth] x [number of pixels in one frame]

## Example

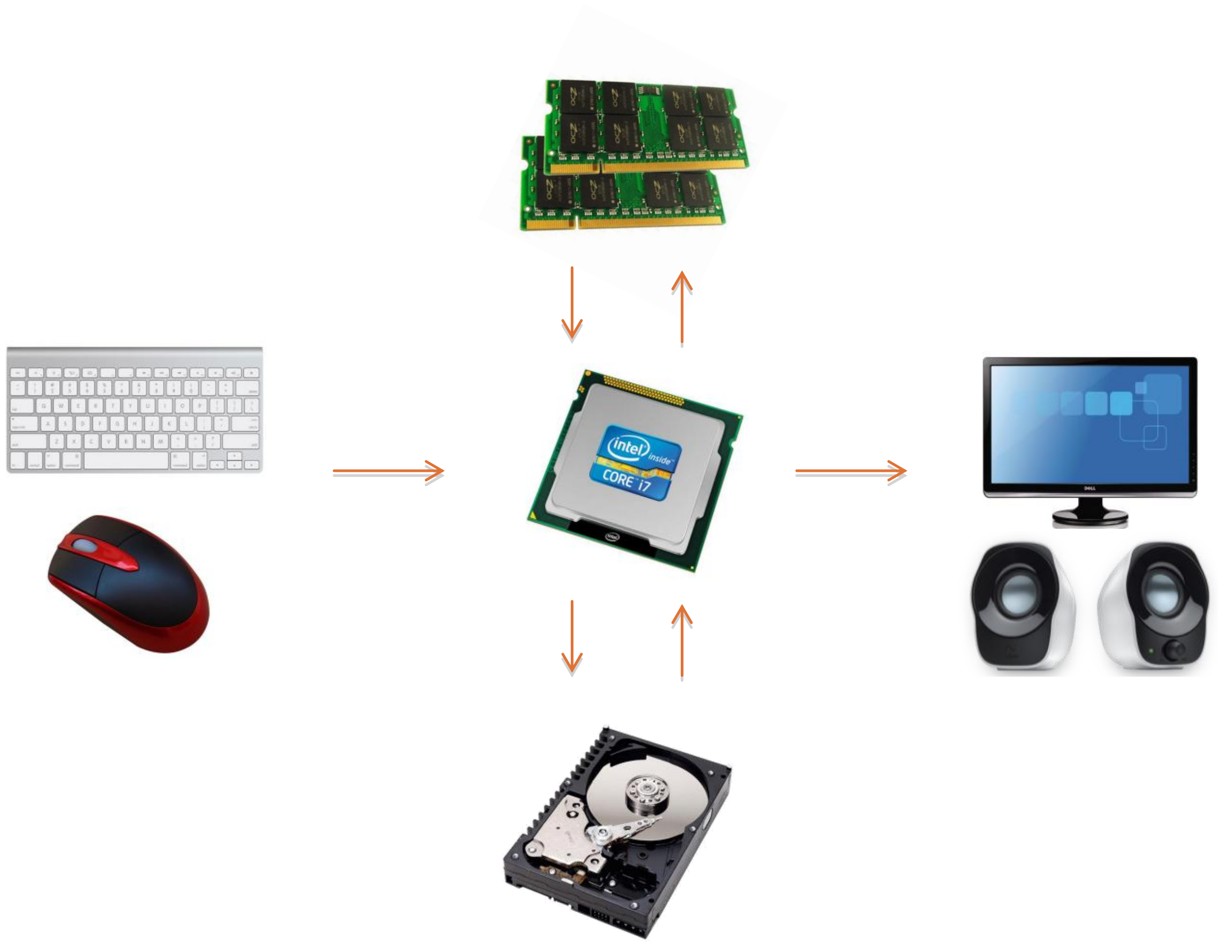
Calculate the file size of a twenty second video file with a frame rate of 25. The video has a resolution of 640 x 480 and a bit depth of 16.

Video file size = [time in seconds] x [frame rate] x [bit depth] x [number of pixels in one frame]
Video file size = [20] x [25] x [16] x [640 x 480]
Video file size = 2,457,600,000 bits / 8
Video file size = 307, 200, 000 bytes / 1024
Video file size = 300,000 kilobytes / 1024
Video file size = 292.968… megabytes / 1024
Video file size = 0.29 Gigabytes

# Computer Architecture




**This is simple representation of how a computer works.**

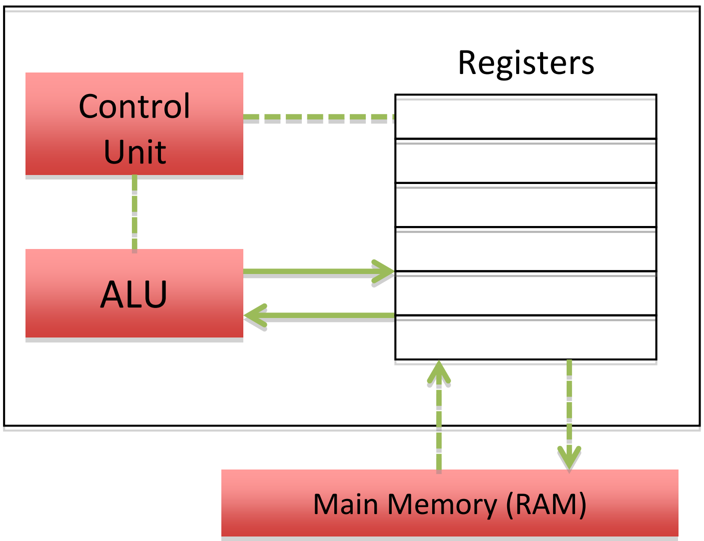


# The Processor (CPU)

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.

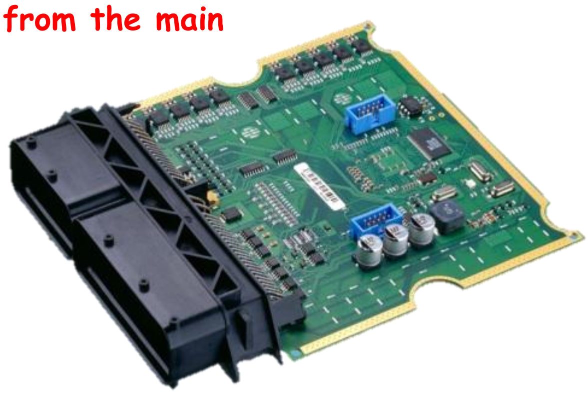

The processor is made up from:

- 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.**



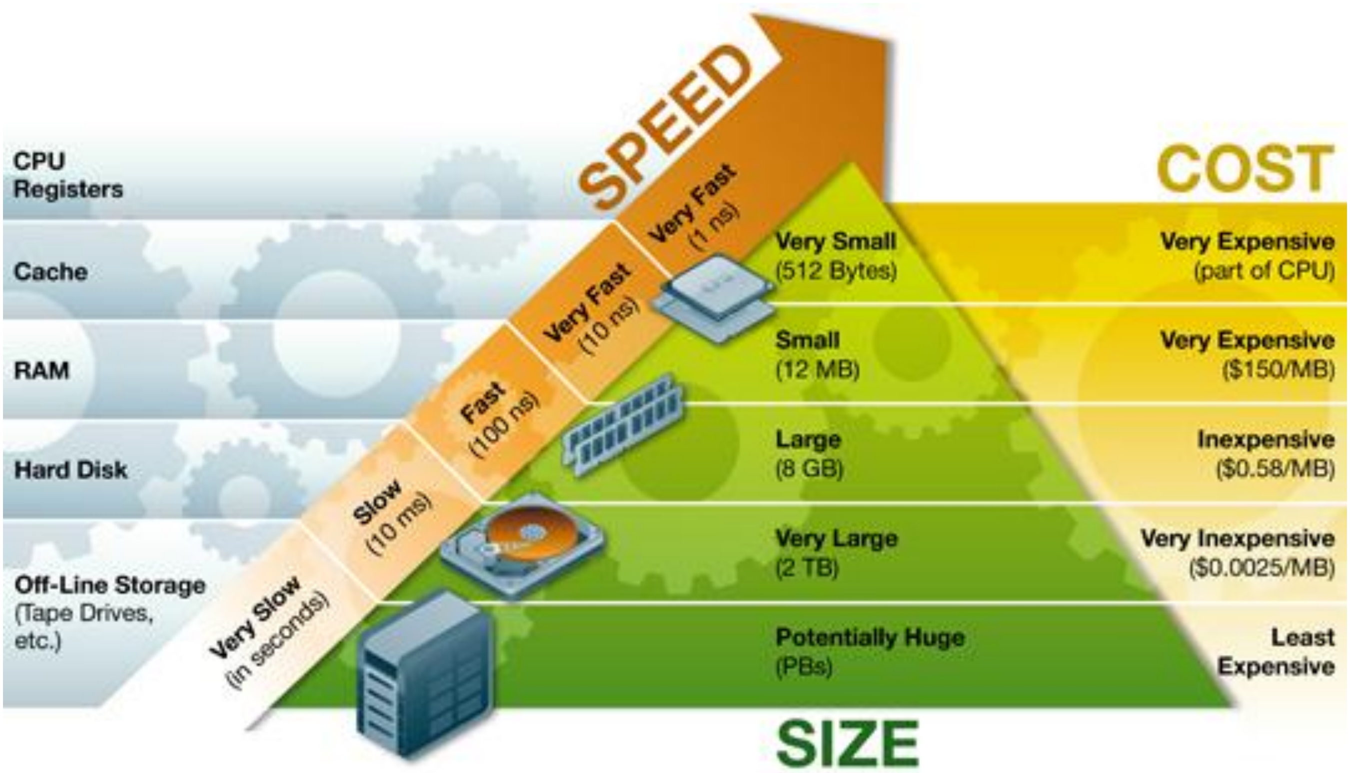
To simplify the control unit is **responsible for running programs that are loaded into main memory.**

# 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
- 3

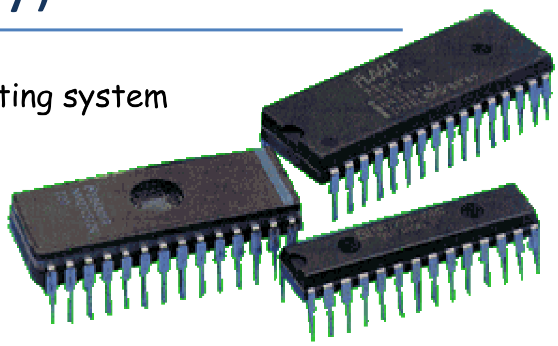


# Main Memory (RAM and ROM)

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.



- 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**

## 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

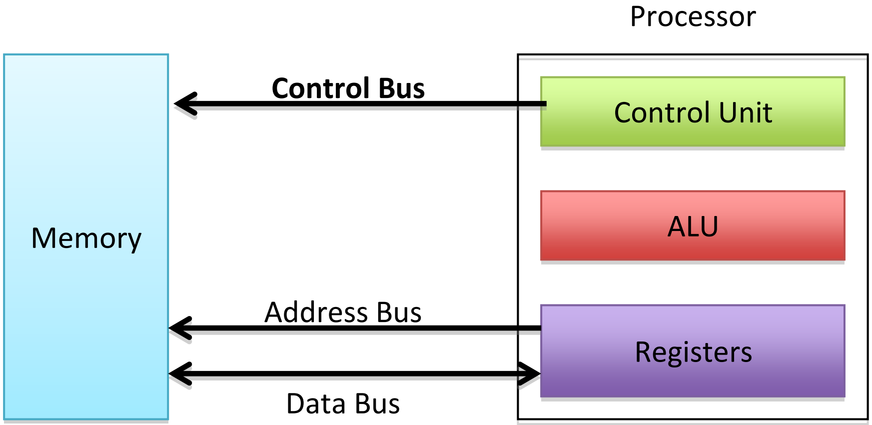
This is another form of memory and is used to store data such as music, movies, documents and software. **Backing storage is the slowest form of memory**.

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

# Addressability

The location of each item is stored in a computer's memory is important because the computer has to be able to find any given item of data.

**Each storage location has its own unique address in the computer's main memory**. The number of storage locations in the computer's memory that a processor can identify depends on the number of bits in the address or the number of wires in the address bus.

The quantity of memory which may be addressed, can be calculated using the formula:

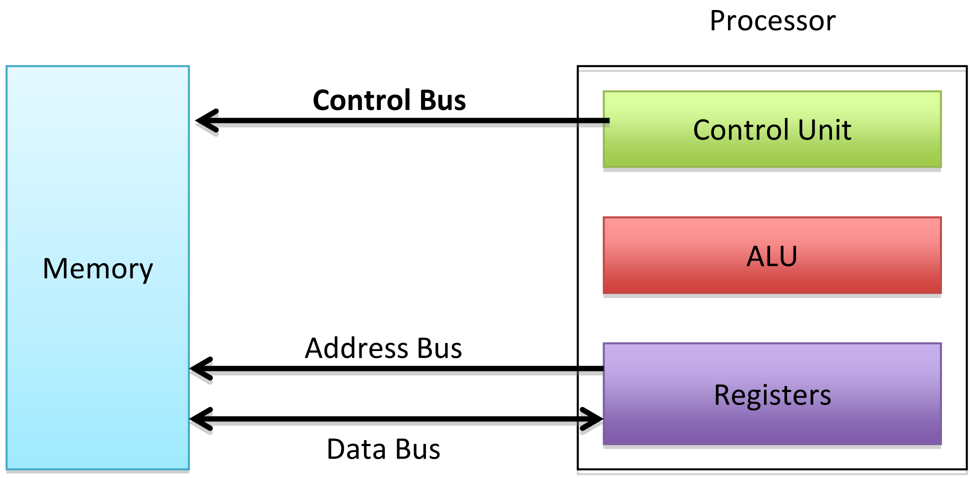
Maximum addressable memory = $2^{\text{width of the address bus}}$ x data bus

Here is an example of a calculation.

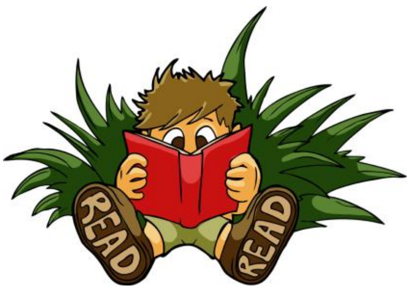
A computer contains a processor with a 32-bit address bus and a 128-bit data bus.

$2^{\text{Addressbus}}$ x data bus
$2^{32}$ x 128 = 549755813888 Bits / 8
= 68719476736 Bytes / 1024
= 67108864 kB / 1024
= 65536 MB / 1024
= 64 GB

# Fetch Execute Cycle




## Memory Read



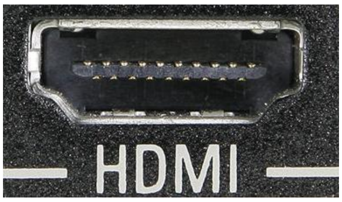

1. The processor sets up the address bus with required memory address.
2. The processor activates the read line on the control bus.
3. The contents of the memory location are transferred along the data bus into the processor.
4. If it is an instruction it is decoded and executed.

## Memory Write

1. The processor sets up the address bus with required memory address.
2. The processor sets up the data bus with the data to be written to memory.
3. The processor activates the write line on the control bus
4. The data is transferred along the data bus to the storage location in the computers memory

# Interfaces

Interfaces **go between the computer and a device so they can work together**. We plug most of our peripherals into **USB ports** although there are other interfaces such as HDMI, VGA, SD Card Slots and many more. Between the computer and the device there are various functions the interfaces do.


| Interface | Mbps | MB / s |
|---|---|---|
| **USB 2.0** | 480 | 60 |
| **USB 3.0** | 5 Gbps | 640 |
| **Gigabit Ethernet** | 1000 | 125 |

# Buffering

A buffer is a region of a physical memory storage used to **temporarily store data while it is being moved from one place to another**. Typically, the data is stored in a buffer as it is retrieved from an input device (such as a microphone) or just before it is sent to an output device (such as speakers). Data can be transferred to the buffer and the CPU can then carry on with other processing tasks. A Buffer uses RAM to store the data.

# Spooling

Spooling involves writing a file to a backing storage device so that the **slower peripheral such as a printer can print the file at a speed that it can handle**. Again this is done so that the processor can be freed up to continue with other processing tasks. Spooling uses a hard disc to the store data.

# Functions of an Interface

## Data Conversion

**Computers use digital but the majority of its peripherals and communication in the outside world are in analogue**. To make sure the computer understands the information converters are used. An Analogue to Digital Converter (ADC) is used when signals come into interface and Digital to Analogue Converter (DAC) when signals go out from the computer.

## Speed

Devices **send and receive data at different rates** and the interface needs to set it so both the computer and the device has the right speed.

## Status Information

Provide **information on the current state of the device**. For example a printer interface will provide information such as ready to accept more data, out of paper etc.

## Data Storage

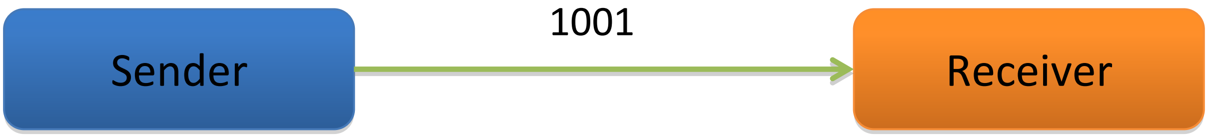
An **area to store data until the slower peripheral device is ready to accept it**. For example, a printer buffer is required as the processor can send many pages at once and printer is too slow to deal with them, the CPU puts them in the buffer and can get on with other work and printer prints from the buffer at its own speed.

## Protocol Conversion

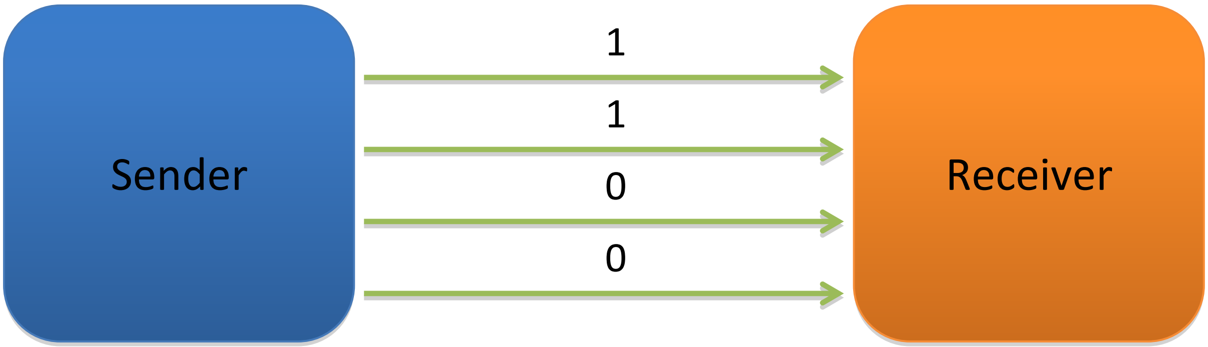
This is a set of rules agreed so that the software and hardware can communicate with each other.

# Parallel & Serial Interfaces

**Serial Interface** – is where data is transferred **one bit after another down a single channel.**




**Parallel Interface** – is where **multiple bits are transferred down multiple parallel channels.** This type of data transmission is much faster than serial transmission.



# Mobile Devices

A mobile device is a **small, handheld computing device**, typically having a display screen with touch input and/or a virtual keyboard.

Apple, Samsung, HTC and BlackBerry are just a few examples of the many manufacturers that produce these types of devices.

A mobile device has an **operating system** (OS), and can **run various types of application software**, known as apps. Most mobile devices can also be equipped with Wi-Fi, Bluetooth, and GPS capabilities that can allow connections to the Internet. A camera or media player feature for video or music files can also be typically found on these devices along with a stable battery power source such as a lithium battery.

| Spec | iPhone 5S | Samsung Galaxy S4 |
|---|---|---|
| **CPU** | Dual Core 1.3GHz | Quad-core 2.3 GHz |
| **RAM** | 1 GB DDR3 | 2GB |
| **Battery Life** | 250 Hours | 270 Hours |
| **Internal Storage** | Up to 64 GB | 32GB (expandable) |
| **Screen Resolution** | 640 x 1136 pixels | 1080 x 1920 pixels |
| **Camera** | 8 MP | 13 MP |


The above specifications of a couple of the current smartphones on the market give you an idea of the power of the current mobile devices. **The above CPU speeds where only once seen in desktop computers and now are becoming increasingly popular in mobile devices.**

## Virtual Machines

A virtual machine is a **self-contained operating environment that behaves as if it is a separate computer**. For example a piece of software called VM Ware can let you **run Windows programs on a Mac**.

Software like VM Ware is able to **launch other operating systems** in such a way that each operating system behaves **as if it were installed on a self-contained computer with its own set of programs and hardware resources**.

## Emulators


Emulators allow your computer to **act like the actual hardware of the emulated system**, such as a console system like a GameBoy or a computer system like the Apple 2. They can also emulate the hardware of various classic arcade games.

Emulators are **complex pieces of software**, and most emulators do not perfectly emulate the abilities of the system it is trying to copy. With some emulators, the imperfections may be very minor, such as a slight timing problem. Some emulators won't run certain games/programs at all, or they may display problems. Other emulators may lack sound, joystick support, or other significant features due to the difficulty of emulating them.