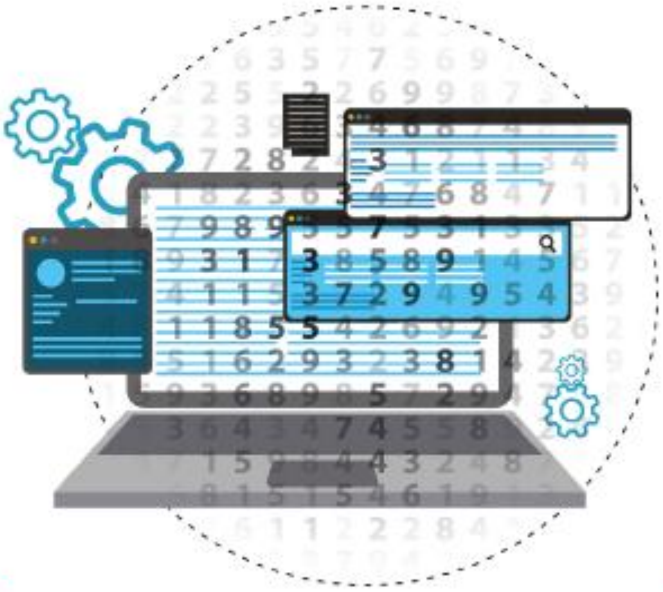
Cambridge International
AS & A Level
Computer Science 9618

Topic-1 Information Representation



Chapter-1 Content Overview

Measurement of the size of computer memory

- The smallest unit of data in a computer is represented by Bits.
- A group of 8 bits is called a byte.
- A common way of expressing multiples of bytes is to use **denary prefixes**.
- This system relies on the assumption that 1 kilo = 1000
- This assumption is based on the denary (base 10) number system

Name of memory size	Equivalent denary value (bytes)
1 kilobyte (1 KB)	1 000
1 megabyte (1 MB)	1 000 000
1 gigabyte (1 GB)	1 000 000 000
1 terabyte (1 TB)	1 000 000 000 000
1 petabyte (1 PB)	1 000 000 000 000 000

- The International Electrotechnical Commission (IEC) introduced memory measurement based on binary system (**Binary Prefix**)

Name of memory size	Number of bytes	Equivalent denary value (bytes)
1 kibibyte (1 KiB)	2^{10}	1 024
1 mebibyte (1 MiB)	2^{20}	1 048 576
1 gibibyte (1 GiB)	2^{30}	1 073 741 824
1 tebibyte (1 TiB)	2^{40}	1 099 511 627 776
1 pebibyte (1 PiB)	2^{50}	1 125 899 906 842 624

Unit	Equals
1 kB	1,000 bytes
1 MB	1,000 KB
1 GB	1,000 MB
1 TB	1,000 GB

Unit	Equals
1 KiB	1,024 bytes
1 MiB	1,024 KiB
1 GiB	1,024 MiB
1 TiB	1,024 GiB

Data Representation: -Number Systems

1. Denary or Decimal Number System

- Base 10 number system
- Uses the digits 0 to 9
- Digits are placed in **'weighted'** columns

10000	1000	100	10	units
3	1	4	2	1

- The denary number represented above is thirty-one thousand, four hundred and twenty-one.

2. Binary Number System

- Base 2 number system
- Uses the bits 0 and 1
- Bits are placed in **'weighted'** columns

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

A typical binary number would be:

1	1	1	0	1	1	1	0

3. Hexadecimal number system

- It's a base 16 number system.
- Because it is a system based on 16 different digits, the numbers 0 to 9 and the letters A to F are used to represent hexadecimal digits.
- A = 10, B = 11, C = 12, D = 13, E = 14 and F = 15.
- Hexadecimal (sometimes referred to as simply hex) is a base 16 system with the weightings:

1 048 576	65 536	4096	256	16	1
(16^5)	(16^4)	(16^3)	(16^2)	(16^1)	(16^0)

Use of Hexadecimal Numbers

- Memory Addressing: In computer systems, memory addresses are often represented in hexadecimal notation. Each address corresponds to a location in the computer's memory, and using hexadecimal simplifies the representation of large memory spaces.

- Colour Codes: Hexadecimal is commonly used to represent colours in web design and graphics. It allows for concise and precise descriptions of colours in the RGB (Red, Green, Blue) and ARGB (Alpha, Red, Green, Blue) colour models.

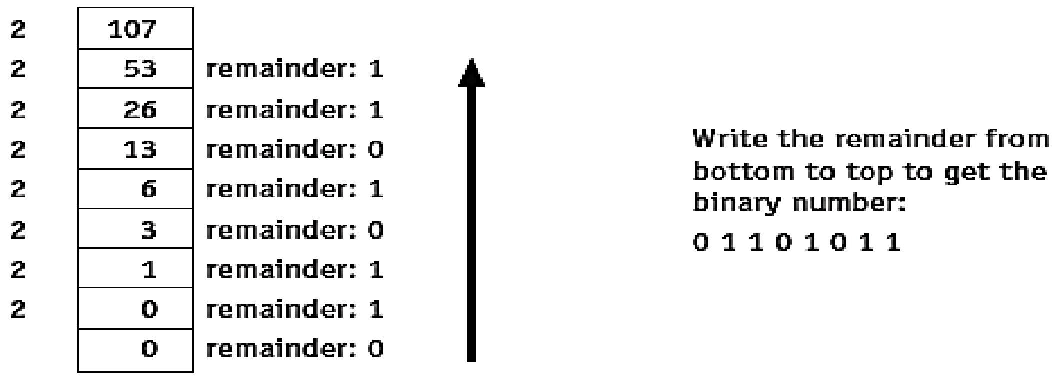
Number Conversions

1. Denary to Binary
2. Binary To Denary
3. Denary to Hexadecimal
4. Hexadecimal to Denary
5. Binary to Hexadecimal
6. Hexadecimal to Binary

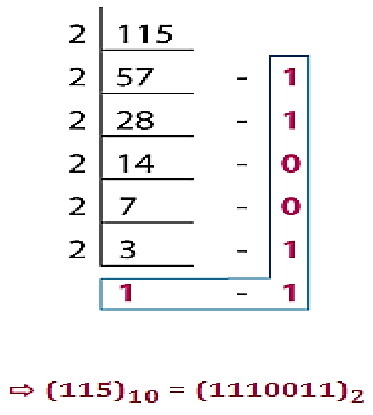
Converting Denary to Binary

There are two methods for converting a **denary** (base 10) number to **binary** (base 2).

- Method-1
  - This method involves successive division by 2, the remainders are then written from bottom to top to give the binary value.
  - For Example: Convert Denary number 107 to its equivalent Binary number



- Convert Denary number 115 to its Binary equivalent



- Method-2
  - This method involves placing **1s** in the appropriate position so that the total equates to the given denary number

- For Example: Convert denary number 107 to its binary equivalent

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

- After adding the place value of each 1's place, we get the total 107.
- 64 + 32 + 8 + 2 + 1 = 107
- So, the binary equivalent is 1101011

- **Convert Denary number 85 to Binary**
  Here, we place the 1s in appropriate positions to get the total value 85

2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	0	1	0	1	0	1

$(85)_{10} \Rightarrow (1010101)_2$

Converting Binary to Denary

- Place the binary bits into its place value columns.
- Each time a 1 appears in a column, the column value is added to the total
- The 0 values are simply ignored when calculating the total.
- For Example: Convert the binary number 1011 to its denary

2^3	2^2	2^1	2^0
1	0	1	1

- By looking at the place values, we can calculate the denary equivalent

$$1 \times (2^3) + 1 \times (2^1) + 1 \times (2^0) = 1 \times 8 + 1 \times 2 + 1 \times 1 = 8 + 2 + 1 = 11$$

- So, the denary equivalent of binary number 1011 is 11

Converting from binary to hexadecimal

1. Starting from the right and moving left, split the binary number into groups of 4 bits.
2. If the last group has less than 4 bits, then simply fill in with 0s from the left.
3. Take each group of 4 bits and convert it into the equivalent hexadecimal digit

Convert 1 0 1 1 1 1 1 0 0 0 0 1 from binary to hexadecimal.

Solution

First split it into groups of 4 bits:

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

Then find the equivalent hexadecimal digits:

B   E   1

Convert 1 0 0 0 0 1 1 1 1 1 1 1 0 1 from binary to hexadecimal.

Solution

First split it into groups of 4 bits:

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

The left group only contains 2 bits, so add in two 0s to the left:

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

Now find the equivalent hexadecimal digits:

2   1   F   D

Converting from hexadecimal to binary

- To convert a hexadecimal number to a binary number, we reverse the above procedure
- Take each hexadecimal digit and write down the 4 bit code which corresponds to the digit.

Convert this hexadecimal number to its binary equivalent.

4   5   A

Solution

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

Put the groups together to form the binary number:

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

Convert this hexadecimal number to its binary equivalent.

B   F   0   8

Solution

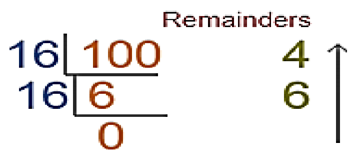
1 0 1 1   1 1 1 1   0 0 0 0   1 0 0 0

Then put all the digits together:

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

Converting Denary to Hexadecimal

Divide the number repeatedly by **16** until the quotient becomes **0**.

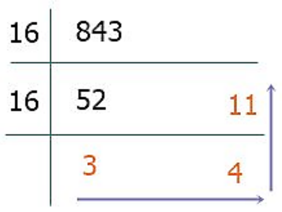


- When **100** is divided by **16**, the quotient is **6** and the remainder is **4**.
- When **6** is divided by **16**, the quotient is **0** and the remainder is **6**.

Write the remainders **from bottom to top.**

$(100)_{10} = (64)_{16}$

- Note: Remainders that are greater than 9 are represented by letters A to F
- Convert denary number 843 to Hexadecimal



$(843)_{10} = (34B)_{16}$

Another method to convert denary to hexadecimal is :

$$\begin{tikzcd}\text{Denary} \arrow[r] & \text{Binary} \arrow[r] & \text{Hexadecimal}\end{tikzcd}$$

Converting Hexadecimal to Denary

- Multiply each digit of the given number, starting from the rightmost digit, with the exponents of the base 16.
- The exponents should start with 0 and increase by 1 every time as we move from right to left.

356	**2AF**
$3 \times 16^2 + 5 \times 16^1 + 6 \times 16^0$	$2 \times 16^2 + A \times 16^1 + F \times 16^0$
$= 854$	$= 687$

Note: **Another method to convert denary to hexadecimal is :**

$$\begin{tikzcd}\text{Hexadecimal} \arrow[r] & \text{Binary} \arrow[r] & \text{Denary}\end{tikzcd}$$

Binary Addition

Rules to remember:

0 + 0	= 0	
0 + 1	= 1	
1 + 0	= 1	
1 + 1	= 10	[0 with carry 1]
1 + 1 + 1	= 11	[1 with carry 1]

Binary Addition (Unsigned numbers)

- Add binary numbers 0101 and 1000

$$\begin{array}{r} 1\ 0\ 0\ 0 \\ +\ 0\ 1\ 0\ 1 \\ \hline 1\ 1\ 0\ 1 \end{array}$$

- Add binary numbers 01111 and 11011

$$\begin{array}{r} 1\ 1\ 1\ 1\ \\ 0\ 1\ 1\ 1\ 1 \\ +\ 1\ 1\ 0\ 1\ 1 \\ \hline 1\ 0\ 1\ 0\ 1\ 0 \end{array}$$

Overflow

What is an overflow?

- Overflow occurs when the sum of two binary numbers exceeds the given number of bits
- In signed number representations, the leftmost bit often serves as the sign bit; overflow can flip this, incorrectly changing the sign of the result
- Let's add two 4-bit numbers:

$$\begin{array}{rl} 1110_2 & \text{(14 in decimal)} \\ +\ 0100_2 & \text{(4 in decimal)} \\ \text{--------} & \\ 10010_2 & \end{array}$$

But in a 4-bit system, we can only keep the last 4 bits: 0010_2 = 2 (decimal).

- Actual sum is 18, but 4-bit max is 15 → overflow occurred.

Representation of Signed Numbers

- An unsigned number contains just zero or positive values, whereas a signed number has both positive and negative numbers along with the value zero.
- The Most Significant Bit (MSB)of signed binary numbers is used to indicate the sign of the numbers. Hence, it is also called as sign bit.
- The positive sign is represented by placing '0' in the sign bit.
- Similarly, the negative sign is represented by placing '1' in the sign bit.

There are three different ways the signed binary numbers can be represented.

1. Signed Magnitude Form
2. 1's Complement Form
3. 2's Complement Form

- Signed Magnitude Form

In sign-magnitude representation, the Most Significant bit of the number is a sign bit and the remaining bit represents the magnitude of the number in a true binary form.

Rules of +ve and –ve binary numbers

Positive binary number always starts with 0 in MSB (Most significant bit)

Negative binary number always starts with 1 in MSB (Most significant bit)

E.g: +ve number: 01010010

-ve number: 10010110

Here is the representation of + 34 and -34 in a 8-bit sign-magnitude form.

+ 34 = 0 0 1 0 0 0 1 0

- 34 = 1 0 1 0 0 0 1 0

- 1's complement form
  - The 1's complement of a number is obtained by complementing all the bits (in other words, 0 becomes 1 and 1 becomes 0) of signed binary number.
  - The one's complement of a positive number is same as its binary representation because positive numbers in one's complement are represented directly in binary.

Here's how you can represent a negative number using one's complement:

1. Convert the positive number to its binary representation.
2. Invert (flip) all the bits in the binary representation.

- Example

Consider the denary number -7, let's represent it in 1's complement form.

	8	4	2	1
+7	0	1	1	1
1's complement	1	0	0	0

To represent -34 in 1's complement form

```
+ 34  =  0  0  1  0  0  0  1  0
         ↓  ↓  ↓  ↓  ↓  ↓  ↓  ↓
- 34  =  1  1  0  1  1  1  0  1    (1's complement of + 34)
```

Here is another example which shows how to represent -60 in 8-bit 1's complement form.

To represent -60 in 1's complement form

```
+ 60  =  0  0  1  1  1  1  0  0
         ↓  ↓  ↓  ↓  ↓  ↓  ↓  ↓
- 60  =  1  1  0  0  0  0  1  1    (1's complement of + 60)
```

- 2's complement form
  - In 2's complement representation also, the representation of the positive number is same as 1's complement and sign-magnitude form.
  - The 2's complement of a negative number is obtained by **adding one to the 1's complement** of signed binary number.

  Let's Consider Denary Number -7

```
                   8 4 2  1

+7                 0 1 1  1

1's complement     1 0 0  0

           +              1
          ------------------------

2's complement    1 0 0 1
```

To represent -34 in 2's complement form

```
+ 34  =  0  0  1  0  0  0  1  0
         ↓  ↓  ↓  ↓  ↓  ↓  ↓  ↓
         1  1  0  1  1  1  0  1    (1's complement of + 34)
      +                        1
      ---------------------------
- 34  =  1  1  0  1  1  1  1  0    (2's complement of + 34)
```

- Positive number representation is same as its binary representation

```
+ 60  =  0  0  1  1  1  1  0  0    (Sign Magnitude Representation)
+ 60  =  0  0  1  1  1  1  0  0    (1's Complement)
+ 60  =  0  0  1  1  1  1  0  0    (2's Complement)
```

Two's complement sums:

Using two's complement, the CPU can perform arithmetic using **binary** addition. For example:

-7 + 7 in two's complement binary would be calculated as:

$$\begin{array}{r} 1\ 0\ 0\ 1 \\ +\ {}_1 0_1 1_1 1_1 1 \\ \hline \cancel{1}\ 0\ 0\ 0\ 0 \end{array}$$

In two's complement, if the final result **overflows** the remaining carry number is simply discarded. For example:

-3 + 4 in two's complement binary would be calculated as:

$$\begin{array}{r} 1\ 1\ 0\ 1 \\ +\ {}_1 0_1 1\ 0\ 0 \\ \hline \cancel{1}\ 0\ 0\ 0\ 1 \end{array}$$

Binary subtraction

- Method 1: Binary subtraction using binary numbers

The **binary subtraction table** is as follows:

Binary Number	Subtraction Value
0 – 0	0
1 – 0	1
0 – 1	1 (Borrow 1 from the next high-order digit)
1 – 1	0

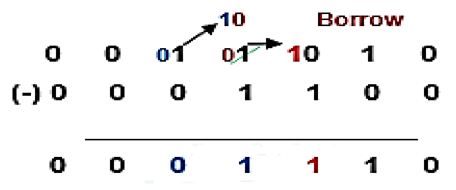
Example 1:

0011010 – 001100

Solution:

2 - 1 = 1 (denary)

after borrow its 10 – 1 = 1 (binary)

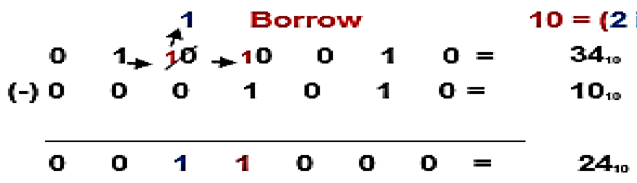


Example 2:

0100010 – 0001010

Solution:

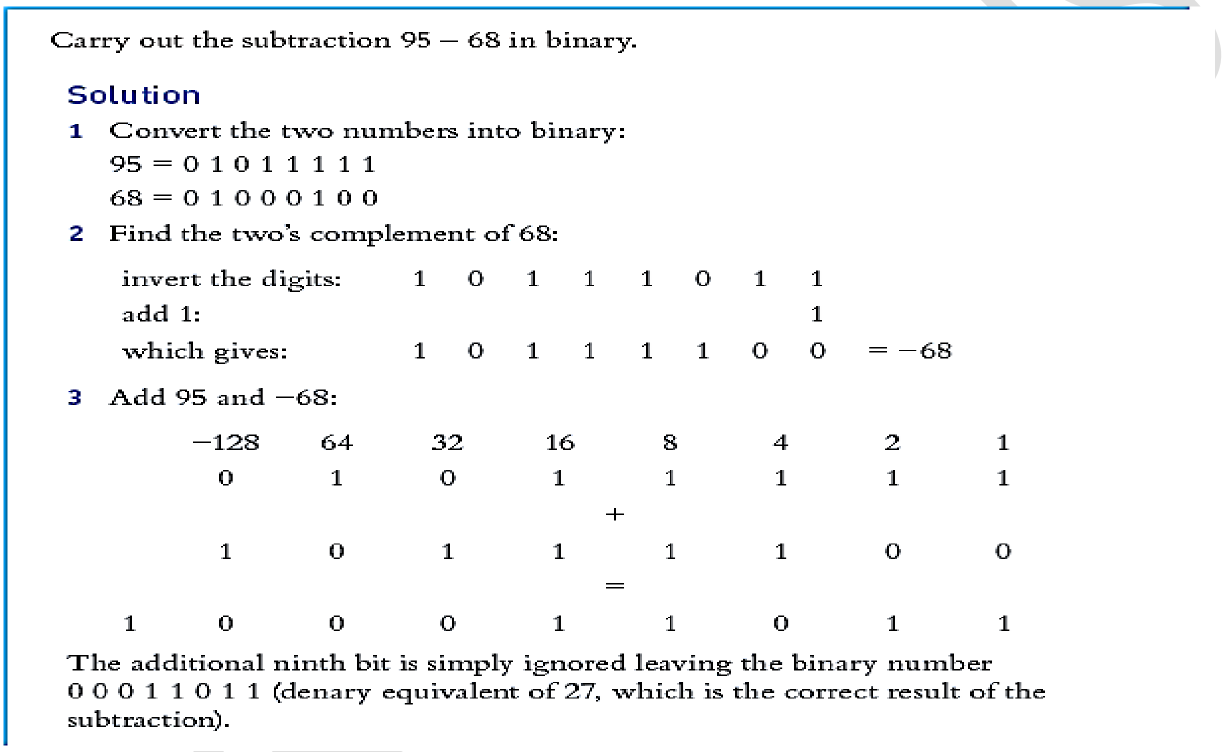
10 = (2 in denary) so 10 - 1 = 1



Method 2: Binary subtraction converting denary numbers to binary numbers

- Steps:

1. Find the 2's complement of the subtrahend (the number to be subtracted).
2. Add the 2’s complement of the subtrahend to the minuend (the number from which you're subtracting).
3. If there's an overflow (carry out of the leftmost bit), discard it.
4. If no overflow occurs, the result is negative and it’s in the 2’s complement form, and you may need to take the 2’s complement of the result for interpretation.



Carry out the subtraction 95 − 68 in binary.

Solution

1 Convert the two numbers into binary:
95 = 0 1 0 1 1 1 1 1
68 = 0 1 0 0 0 1 0 0

2 Find the two’s complement of 68:

invert the digits:	1	0	1	1	1	0	1	1	
add 1:								1	
which gives:	1	0	1	1	1	1	0	0	= −68

3 Add 95 and −68:

	−128	64	32	16	8	4	2	1
	0	1	0	1	1	1	1	1
				+				
	1	0	1	1	1	1	0	0
				=				
1	0	0	0	1	1	0	1	1

The additional ninth bit is simply ignored leaving the binary number 0 0 0 1 1 0 1 1 (denary equivalent of 27, which is the correct result of the subtraction).

- Binary-Coded Decimal (BCD) system
  - The binary-coded decimal (BCD) system uses a 4-bit code to represent each denary digit

0 0 0 0 = 0	0 1 0 1 = 5
0 0 0 1 = 1	0 1 1 0 = 6
0 0 1 0 = 2	0 1 1 1 = 7
0 0 1 1 = 3	1 0 0 0 = 8
0 1 0 0 = 4	1 0 0 1 = 9

Therefore, the denary number 3 1 6 5 would be 0 0 1 1 0 0 0 1 0 1 1 0 0 1 0 1 in BCD format.

The 4-bit code can be stored in the computer either as half a byte or two 4-bit codes stored together to form one byte. For example, using 3 1 6 5 again ...

Method 1: four single bytes

0	0	0	0	0	0	1	1	3
0	0	0	0	0	0	0	1	1
0	0	0	0	0	1	1	0	6
0	0	0	0	0	1	0	1	5

Method 2: two bytes

0	0	1	1	0	0	0	1	3	1
0	1	1	0	0	1	0	1	6	5

- Uses of BCD

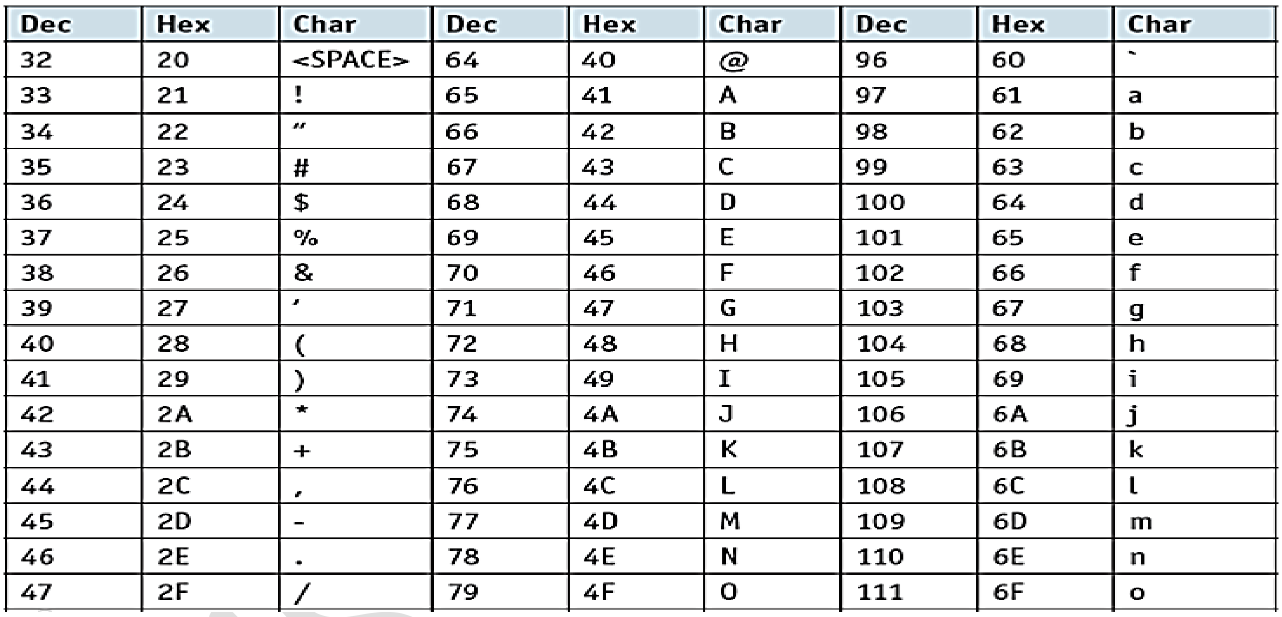
Use Case	Why BCD is used
Electronic calculators	Keeps numbers in decimal format for easier display and accuracy
Digital clocks and watches	Time is naturally decimal (e.g. 12:45), so BCD makes display logic simpler
Banking and financial systems	Avoids rounding errors when doing decimal calculations, especially with money
Old digital systems / embedded systems	Simpler to implement with hardware that displays digits individually

Character sets

- A character set is all the characters and symbols that can be represented by a computer system
- Each character is given a unique binary code
- A character set provides a standard for computers to communicate and send/receive information
- The number of characters that can be represented is determined by the number of bits used by the character set
- Two common character sets are:
  - **American Standard Code for Information Interchange (ASCII)**
  - **Universal Character Encoding (UNICODE)**

ASCII

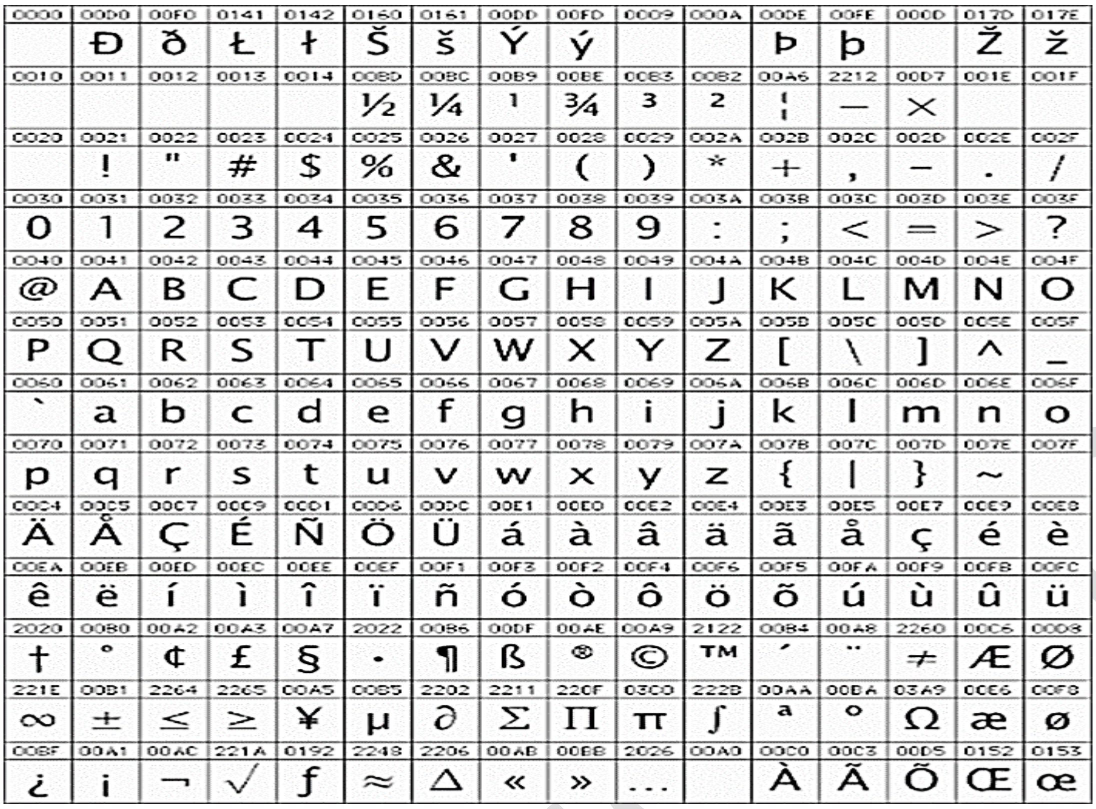
- ASCII uses 7 bits, providing 2^7 unique codes (128) or a maximum of 128 characters it can represent.
- This is enough to represent the letters, numbers, and symbols from a standard keyboard
- ASCII defines a set of 128 characters, which includes the English alphabet (both uppercase and lowercase), numerals (0-9), punctuation marks, control characters, and a few special symbols.
- **Extended ASCII (8-bit)**: Extended ASCII uses an 8th bit, providing 256 unique codes ($2^8 = 256$) or a maximum of 256 characters it can represent.
- Extended ASCII provides essential characters such as mathematical operators and more recent symbols such as ©



Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
32	20	<SPACE>	64	40	@	96	60	`
33	21	!	65	41	A	97	61	a
34	22	"	66	42	B	98	62	b
35	23	#	67	43	C	99	63	c
36	24	$	68	44	D	100	64	d
37	25	%	69	45	E	101	65	e
38	26	&	70	46	F	102	66	f
39	27	'	71	47	G	103	67	g
40	28	(	72	48	H	104	68	h
41	29	)	73	49	I	105	69	i
42	2A	*	74	4A	J	106	6A	j
43	2B	+	75	4B	K	107	6B	k
44	2C	,	76	4C	L	108	6C	l
45	2D	-	77	4D	M	109	6D	m
46	2E	.	78	4E	N	110	6E	n
47	2F	/	79	4F	O	111	6F	o

Unicode

- UNICODE is a character set and was created as a solution to the limitations of ASCII
- UNICODE can represent characters from all the major languages around the world
- UNICODE uses 8 bits, 16 bits or 32 bits based on the encoding formats.
- **Encoding forms:**
  - **UTF-8** → variable (1 to 4 bytes), most common on the web.
  - **UTF-16** → 2 or 4 bytes.
  - **UTF-32** → 4 bytes (fixed size).
- Multilingual Support: Unicode supports a vast range of languages and writing systems, including Latin, Greek, Chinese, Japanese, Arabic, and many others.



0000	00D0	00F0	0141	0142	0160	0161	00DD	00FD	0009	000A	00DE	00FE	000D	017D	017E
	Đ	ð	Ł	ł	Š	š	Ý	ý			Þ	þ		Ž	ž
0010	0011	0012	0013	0014	00BD	00BC	00B9	00BE	00B3	00B2	00A6	2212	00D7	001E	001F
					½	¼	¹	¾	³	²	¦	−	×		
0020	0021	0022	0023	0024	0025	0026	0027	0028	0029	002A	002B	002C	002D	002E	002F
	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0030	0031	0032	0033	0034	0035	0036	0037	0038	0039	003A	003B	003C	003D	003E	003F
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0040	0041	0042	0043	0044	0045	0046	0047	0048	0049	004A	004B	004C	004D	004E	004F
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0050	0051	0052	0053	0054	0055	0056	0057	0058	0059	005A	005B	005C	005D	005E	005F
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
0060	0061	0062	0063	0064	0065	0066	0067	0068	0069	006A	006B	006C	006D	006E	006F
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0070	0071	0072	0073	0074	0075	0076	0077	0078	0079	007A	007B	007C	007D	007E	007F
p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
00C4	00C5	00C7	00C9	00D1	00D6	00DC	00E1	00E0	00E2	00E4	00E3	00E5	00E7	00E9	00E8
Ä	Å	Ç	É	Ñ	Ö	Ü	á	à	â	ä	ã	å	ç	é	è
00EA	00EB	00ED	00EC	00EE	00EF	00F1	00F3	00F2	00F4	00F6	00F5	00FA	00F9	00FB	00FC
ê	ë	í	ì	î	ï	ñ	ó	ò	ô	ö	õ	ú	ù	û	ü
2020	00B0	00A2	00A3	00A7	2022	00B6	00DF	00AE	00A9	2122	00B4	00A8	2260	00C6	00D8
†	°	¢	£	§	•	¶	ß	®	©	™	´	¨	≠	Æ	Ø
221E	00B1	2264	2265	00A5	00B5	2202	2211	220F	03C0	222B	00AA	00BA	03A9	00E6	00F8
∞	±	≤	≥	¥	µ	∂	Σ	Π	π	∫	ª	º	Ω	æ	ø
00BF	00A1	00AC	221A	0192	2248	2206	00AB	00BB	2026	00A0	00C0	00C3	00D5	0152	0153
¿	¡	¬	√	ƒ	≈	∆	«	»	…		À	Ã	Õ	Œ	œ

Similarities and differences between the ASCII and Unicode character sets

- Similarity
  - · both can use 8 bits
  - · both represent each character using a unique code
  - · Unicode will contain all the characters that ASCII contains // ASCII is a subset of Unicode
- Differences:
  - · Unicode can go up to 32 bits per character whereas ASCII is 7 or 8 bits
  - . Unicode can represent a wider range of characters than ASCII
  - · different languages are represented using Unicode, ASCII is only for one language

1.2 Multimedia-Graphics, Sound

Bit-Map Images and Vector Graphic

- Images can be stored in a computer in two common formats:
  - bit-map image
  - vector graphic

Bit-Map Image (Raster Image)

- A bitmap image is made up of squares called pixels (Picture Element)
- A pixel is the smallest element of a bitmap image
- Each pixel is stored as a binary code
- Binary codes are unique to the colour in each pixel
- A typical example of a bitmap image is a photograph
- The more colours and more detail in the image, the higher the quality of the image and the more binary that needs to be stored

Here are some key characteristics and features of bitmap images:

Resolution: The resolution of a bitmap image is determined by the number of pixels it contains, both in width and height. Higher-resolution images have more pixels and, as a result, greater detail and clarity.

Pixel Density: The number of pixels per unit of area is known as pixel density. Higher pixel density results in sharper and more detailed images.

File Formats: Bitmap images can be saved in various file formats, including JPEG, PNG, GIF, BMP, and TIFF, among others. The choice of file format may affect factors like compression and image quality.

Scaling: When bitmap images are scaled up (enlarged) beyond their original resolution, individual pixels become more visible, leading to pixelation and a loss of image quality.

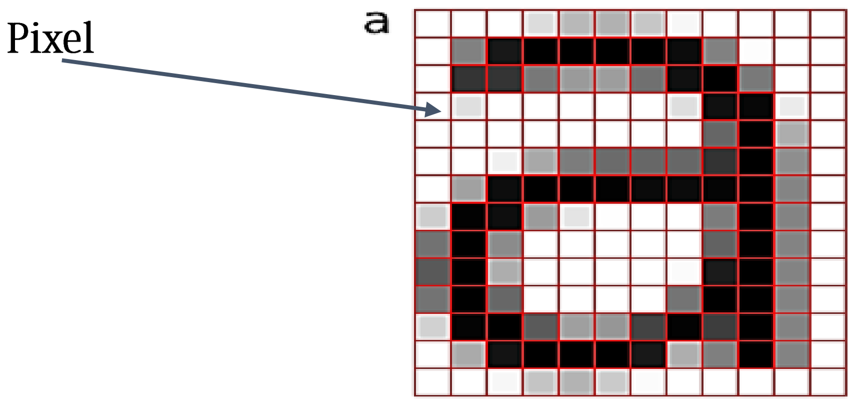
Editing: Bitmap images can be edited using image editing software like Adobe Photoshop. You can manipulate individual pixels to make changes to colour, brightness, contrast, and other attributes.

Digital Photography: Digital cameras capture photographs as bitmap images. The resolution of the camera's sensor and the settings used determine the quality and size of the resulting image.

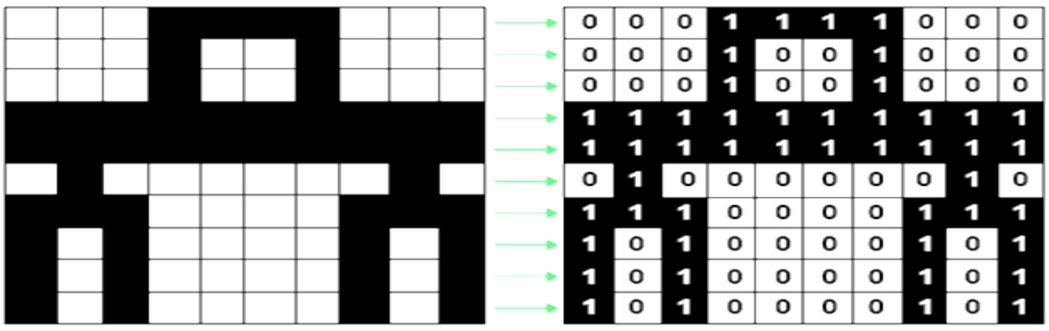
Display: Bitmap images are used in various types of displays, including computer monitors, television screens, and digital projectors. The quality of the displayed image depends on the screen's resolution and pixel density.

Common Use Cases: Bitmap images are commonly used for photographs, web graphics, digital art, and any content where precise control over individual pixels and colours is required.

- Pixels
  - Short for "**picture elements**," are the smallest individual units of a digital image or display.
  - Each pixel is a tiny, square or rectangular area that contains colour information and represents a single point in the image.
  - Pixels are the building blocks of digital images and screens.



- **When storing images as pixels**, we have to consider:
  - **at least 8 bits** (1 byte) per pixel are needed to code a **coloured image** (this gives 256 possible colours by varying the intensity of the blue, green and red elements)
  - true colour requires 3 bytes per pixel (24 bits), which gives more than one million colours
- In the **black-and-white** image below, each pixel is either black or white. As each pixel is either black or white, this image can be encoded with a value of **0 for white and 1 for black**.



Bit Depth/ Colour Depth :

- Bit depth is a measure of the number of bits used to represent each pixel in an image.
- The colour depth is dependent on the number of colours needed in the image
- In general, the higher the colour depth/bit depth the more detail in the image (higher quality)
- In a black & white image the colour depth would be 1, meaning 1 bit is enough to create a unique binary code for each colour in the image (1=white, 0=black)
- An 8-bit image can represent 2^8 (256) different values per pixel, while a 24-bit image can represent 2^24 (over 16 million) different values per pixel.

Image Resolution:

- Image resolution refers to the number of pixels or dots that make up a digital image. It is expressed in terms of the number of pixels in the horizontal (width) and vertical (height) dimensions of the image.
- A higher image resolution means the image contains more pixels, offering greater detail and allowing it to be displayed or printed at larger physical sizes without a loss of quality.

Screen Resolution (Display Resolution):

- Screen resolution, often referred to as display resolution, pertains to the number of pixels a display device, such as a computer monitor, television screen, or smartphone screen, can show at a specific setting.
- It is expressed in terms of the number of pixels in the width and height of the screen.
- Screen resolution determines the level of detail that can be displayed on the screen, and it affects how content appears. Higher screen resolutions provide sharper and more detailed visuals.

Image Resolution Screen Resolution

Pixel
1366 pixels
768 pixels

•

Calculating bit-map image file sizes

- It is possible to estimate the file size needed to store a bit-map image.
- The file size will need to take into account the **image resolution and bit depth**.
- For example, a full screen with a resolution of 1920 x 1080 pixels and a bit depth of 24 requires 1920 x 1080 x 24 bits = 49 766 400 bits for the full screen image.
- In other words, **file size=Total Pixels x Bit Depth**

- **To calculate the number of pixels in an image** from its resolution, you need to multiply the width resolution (in pixels) by the height resolution (in pixels).
  The formula is:

Total Pixels = Width Resolution x Height Resolution

- For example, if you have an image with a resolution of 1920 pixels in width and 1080 pixels in height (commonly known as Full HD resolution), you can calculate the total number of pixels as follows:
- Total Pixels = 1920 pixels (width) x 1080 pixels (height) = 2,073,600 pixels

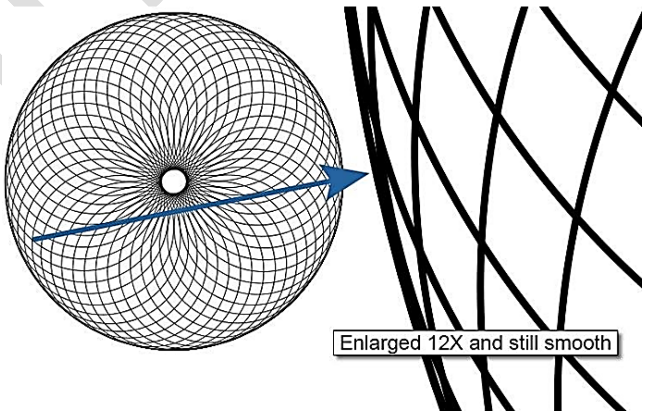
  Note: when saving a bit-map image, it is important to include a **file header;** this will contain items such as **file type** (.bmp or .jpeg), **file size**, **image resolution**, **bit depth** (usually 1, 8, 16, 24 or 32), any type of data compression employed and so on.

What is the relationship between image quality and filesize?

- As the **resolution and/or colour depth increases**, the **bigger the size of the file** becomes on secondary storage
- The **higher the resolution**, the **more pixels** are in the image, the more bits are stored
- The **higher the colour depth**, the **more bits per pixel** are stored
- **Compression** can be used on high quality images to help reduce the file size

2. Vector graphics

- **Vector graphics** are a type of computer graphics that **use geometric shapes**, such as points, lines, curves, and polygons, to represent images and illustrations.
- A vector graphic is created from **mathematical equations** and **points**
- Vector graphics can be designed using computer aided design (**CAD**) software or using an application which uses a drawing canvas on the screen (Eg: Adobe Illustrator)
- This means that they can be scaled to any size without a loss of quality, as they are defined mathematically rather than by individual pixels



- **Here are some key characteristics and advantages of vector graphics**:

➢ **Scalability**: Vector graphics can be resized without loss of quality. This makes them ideal for logos, icons, and images that need to be used in various sizes, from small icons to large billboards.

➢ **Small File Sizes**: Vector files are often smaller in size compared to bitmap images because they store the mathematical representations of shapes rather than individual pixel data.

➢ **Resolution Independence**: Vector graphics look sharp and clear on high-resolution displays as well as low-resolution screens.

➢ **Editability**: Vector graphics are easy to edit and modify.

- **Common vector graphics file formats include**:

➢ **SVG (Scalable Vector Graphics)**: An XML-based format widely used on the web for creating vector graphics that can be scaled and animated within web pages.

➢ **AI (Adobe Illustrator)**: A proprietary vector graphics format used by Adobe Illustrator, a popular vector graphics design software.

➢ **CDR (CorelDRAW)**: A proprietary vector graphics format used by CorelDRAW software.

❖ Drawing Object:

  - A drawing object refers to any individual element or entity within a vector graphic.
  - It is a component created using a formula
  - Drawing objects can encompass various geometric shapes, text, lines, curves, and other design components

❖ Drawing Property:

  - Drawing Properties are attributes or characteristics associated with a drawing object. These attributes describe how the object should be rendered or behave within the drawing.
  - It is an attribute of a drawing object
  - Properties can include details such as colour, size, position, rotation, line thickness, fill pattern, opacity, and more.
  - It defines one aspect of the appearance of a drawing object

❖ Drawing List:

Drawing list is a structured catalogue or inventory of all the drawing objects present in a vector graphic or drawing.

- Contents of a vector graphic drawing list includes:
    - List of objects in the drawing
    - A list that stores the command/description/equation required to draw each object
    - Properties of each object e.g. the fill colour, line weight/colour

Comparison between vector graphics and bit-map images

Vector graphic images	**Bit-map images**
made up of geometric shapes which require definition/attributes	made up of tiny pixels of different colours
to alter/edit the design, it is necessary to change each of the geometric shapes	possible to alter/edit each of the pixels to change the design of the image
they do not require large file size since it is made up of simple geometric shapes	because of the use of pixels (which give very accurate designs), the file size is very large
because the number of geometric shapes is limited, vector graphics are not usually very realistic	since images are built up pixel by pixel, the final image is usually very realistic
file formats are usually .svg, .cgm, .odg	file formats are usually .jpeg, .bmp, .png

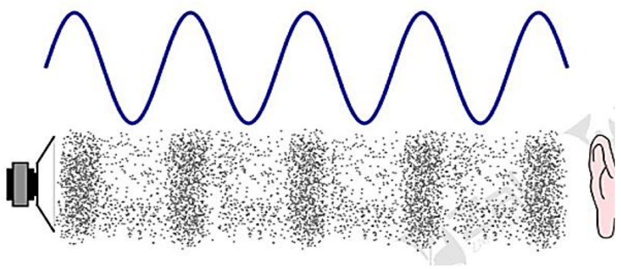
Note:
- Vector graphics are stored as **mathematical equations** that describe shapes, lines, and curves, along with color information (fill, stroke, gradients, etc.).
- Vector graphics are ultimately displayed as pixels on a screen, but their core advantage is that they are **resolution-independent**.
- They are stored as mathematical descriptions of shapes and colors, and the computer calculates the appropriate pixels for whatever resolution or size is needed, allowing the image to be scaled without losing quality.

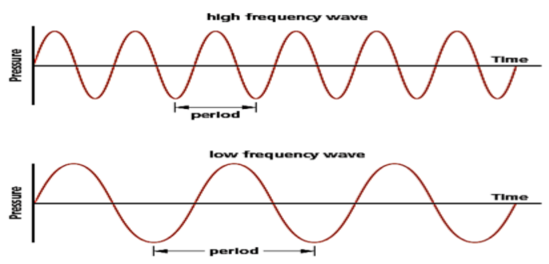
1.3 Sound files

❖ Sound requires a medium in which to travel through (it cannot travel in a vacuum). This is because it is transmitted by causing oscillations of particles within the medium.

❖ The human ear picks up these oscillations (changes in air pressure) and interprets them as sound.

❖ Each sound wave has a frequency and wavelength; the amplitude specifies the loudness of the sound.



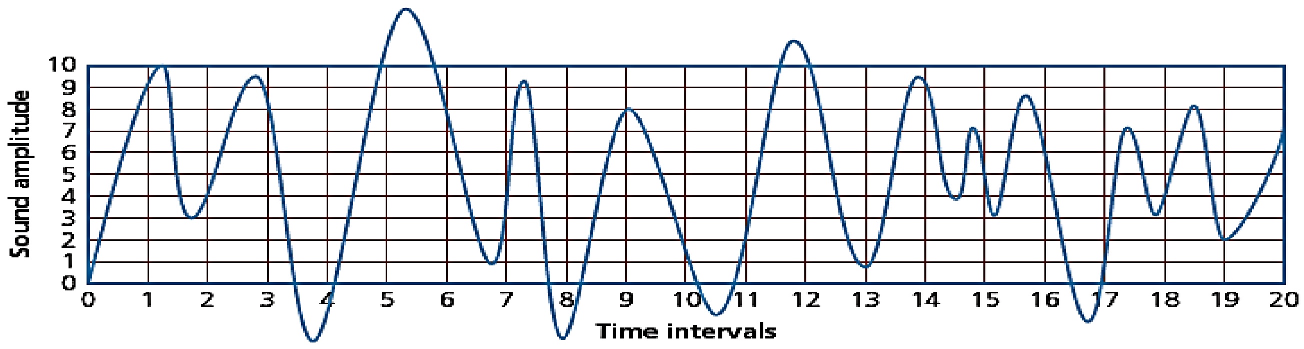


Analogue sound

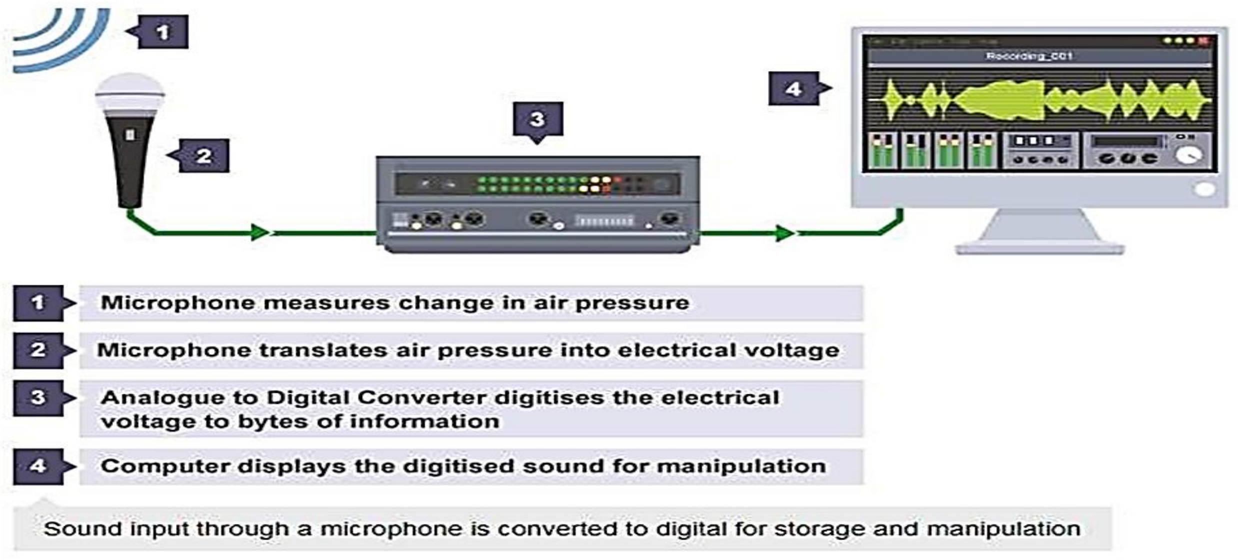
- Sound waves begin as **analogue** and for a computer system to understand them they must be converted into a **digital form**
- Measurements of the original sound wave are **captured** and **stored as binary** on secondary storage
- This process is called **Analogue to Digital conversion** (A2D)
- The process begins by measuring the loudness (**amplitude**) of the analogue sound wave at a point in time, this is called **sampling**
- The higher the amplitude, the **louder** the sound
- Each measurement (**sample**) generates a value which can be represented in **binary** and **stored**
- Using the **samples**, a computer is able to create a **digital version** of the original analogue wave
- The digital wave is **stored** on **secondary storage** and can be played back at any time by reversing the process

Sampling:

- Sampling is the process of taking discrete measurements of an analogue sound wave at regular intervals. These measurements capture the amplitude of the sound wave at specific points in time.



- The x-axis shows the time intervals when the sound was sampled (0 to 20), and the y-axis shows the amplitude of the sampled sound.

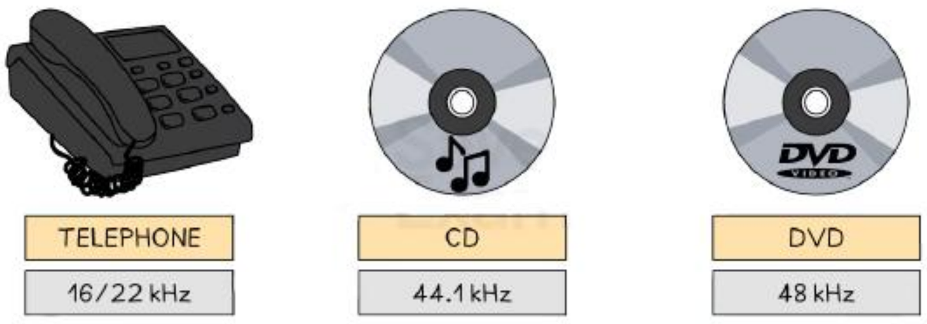


1 Microphone measures change in air pressure
2 Microphone translates air pressure into electrical voltage
3 Analogue to Digital Converter digitises the electrical voltage to bytes of information
4 Computer displays the digitised sound for manipulation

Sound input through a microphone is converted to digital for storage and manipulation

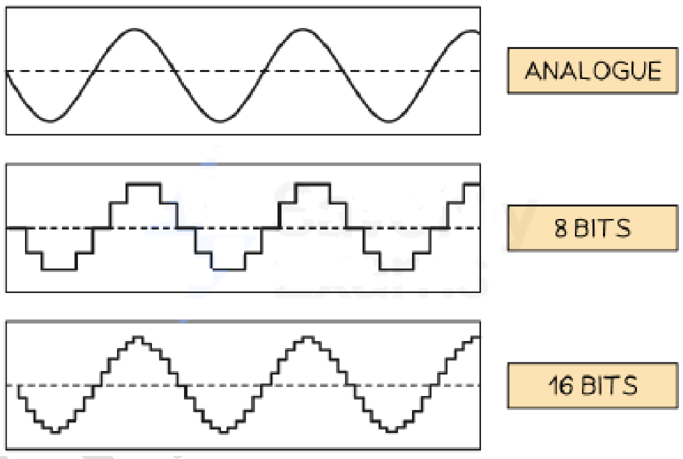
Sampling Rate:

- The sampling rate is the number of **samples taken per second** of the analogue wave
- Samples are taken for the **duration** of the sound
- The sampling rate is measured in **Hertz** (Hz)
- **1 Hertz is equal to 1 sample** of the sound wave
- The sample rate of a typical audio CD is **44.1kHz** (44,100 Hertz or 44,100 samples per second)



Sampling Resolution (Bit Depth):

- Sampling resolution is the **number of bits** used to represent **each sound sample**
- Sampling resolution is closely related to the **bit depth of a bitmap image**, they measure the same thing in different contexts
- Common bit depths are 16-bit and 24-bit. A higher bit depth results in a greater dynamic range and higher audio quality.



What are the impacts of sampling settings?

Factor	Effect of playback quality	Effect on file size
Sampling rate	**higher** = more detail, better sound quality	**higher** = more data, larger file size
Sampling resolution	**higher** = bigger range, better sound quality	**higher** = more data per sample, larger file size

1.3 File Compression

- File compression is the process of reducing the size of a file or a group of files to save storage space.
- There are scenarios where compression may be needed, such as:
  - **Maximise the amount of data you can store** on a digital device such as a mobile phone or tablet
  - **Minimise the transfer time** of data being uploaded, downloaded or streamed across a network such as the Internet
- This reduction in file size is achieved through various compression techniques that remove redundancy and unnecessary data while retaining the essential information needed for reconstruction.
- The two most common forms of file compressions:
  - **lossless file compression**
  - **lossy file compression.**

Lossy File Compression:

- Lossy compression is when **data is lost** in order to **reduce the size** on secondary storage
- Lossy compression is **irreversible**
- Lossy can **greatly reduce the size** of a file but at the **expense of losing quality**
- Lossy is only suitable for data **where reducing quality is acceptable**, for example images, video and sound
- In photographs, lossy compression **will try to group similar colours together**, reducing the amount of colours in the image without compromising the overall quality of the image
- In audio, **frequencies that are outside the human hearing range** can be removed to reduce the file size.
- **MP3** uses **audio compression** to reduce file size.
- **JPEG** is the most common file format for bitmap images

Lossless File Compression:

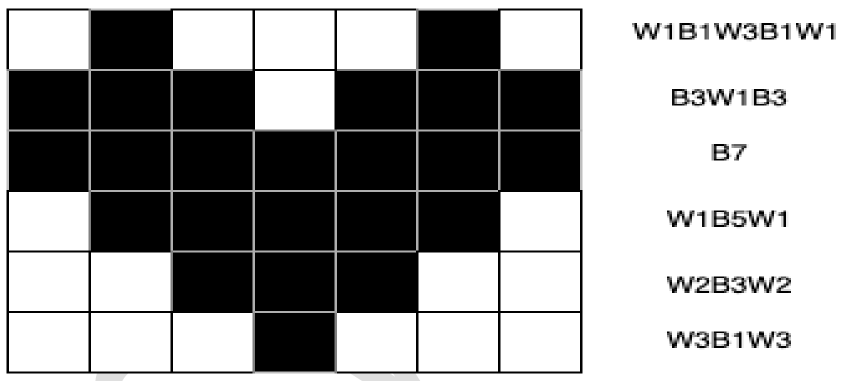
  - Lossless compression is a method for reducing the size of a file while ensuring that no data is lost in the compression process.
  - This means that when a compressed file is decompressed, it is an exact replica of the original file.
  - Lossless compression is **reversible**, the file **can be returned to its original state**
- Lossless can be used on all data but is more suitable for data where **a loss in quality is unacceptable**, for example documents

- In a document, lossless compression uses algorithms to analyse the contents looking for **patterns** and **repetition**
- **Eg**: Run-Length Encoding (RLE)

Run-length encoding (RLE):

- ➢ Run-Length Encoding (RLE) is a simple and efficient lossless compression algorithm used to reduce the size of data by encoding repeated consecutive characters or values as a single character and a count.
- ➢ It is commonly applied to data with long sequences of repeated elements, such as simple graphics, bitmap images, and some types of text data.
- ➢ The key idea behind RLE is to represent sequences of identical characters or values with a shorter representation, which can significantly reduce the storage or transmission size of the data.
- ➢ For example, if you have the string "AAAABBBCCDAA," it can be encoded as "4A3B2C1D2A."

Example of Image compression using RLE


W1B1W3B1W1
B3W1B3
B7
W1B5W1
W2B3W2
W3B1W3