Computing GCSE – 2.6a

J276/02 – Data Representation 1

REMEMBER MAXIMUM VALUES!
Max value which can be represented with 8 bits (1 byte) = **255**
Total number of available values = **256 (255 + 0)**

KEY VOCABULARY

Denary	Base 10 number system. Uses digits 0,1,2,3,4,5,6,7,8,9
Binary	Base 2 number system. Uses digits 0 and 1 only.
Hexadecimal (Hex)	Base 16 number system. Uses characters 0-9 and A,B,C,D,E and F
BIT	Contraction of BINARY DIGIT – a single value of 0 or 1
Binary Code	Representation of values using multiple bits
Character Set	A list of unique values, stored in binary, which represent the letters, numbers and symbols a computer can show/use.
ASCII	American Standard Code for Information Interchange. A character set which uses 7 bits to store 128 (2^7) characters
Extended ASCII	A character set which uses 8 bits to store 256 (2^8) characters
UNICODE	A characters set which uses 16 bits to store 65,535 characters (2^{16})
INTEGER	A whole number (value written to 0 decimal places)
FLOAT	A decimal value
Conversion	Moving a value from one data type/representation to another, for example Binary to Hex
Exponent	Mathematical term which tells you how many time to multiply a BASE by itself.

UNITS OF DATA IN COMPUTER SYSTEMS

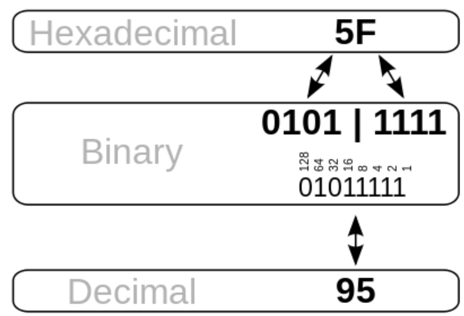
UNIT	VALUE	SIZE
bit (b)	0 or 1	1/8 of a byte
nibble	4 bits	½ a byte (a nibble… get it?!)
byte (B)	8 bits	1 byte
kilobyte (kB)	1000^1 bytes	1,000 bytes
megabyte (mB)	1000^2 bytes	1,000,000 bytes
gigabyte (gB)	1000^3 bytes	1,000,000,000 bytes
terabyte (tB)	1000^4 bytes	1,000,000,000,000 bytes
petabyte (pB)	1000^5 bytes	1,000,000,000,000,000 bytes

BINARY PLACE VALUES

BASE ^{Exponent}	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
PLACE VALUE	128	64	32	16	8	4	2	1

CONVERTING DENARY TO BINARY TO HEX

HEXADECIMAL	
DENARY	HEX
0-9	0-9
10	A
11	B
12	C
13	D
14	E
15	F



There are two methods for converting a HEX value to Denary

OR:

5F = (**5**x16) + **F**
5F = 80 + 15
5F = 95

Computing GCSE – 2.6b

J276/02 – Data Representation 2

KEY VOCABULARY

Overflow Error	Where the denary value cannot be represented with the given number of bits.
Binary Shift	The method for multiplying and dividing numbers in binary. Is not necessarily mathematically correct
Most Significant Bit	The left-most bit in a binary number – it has the highest value
Least Significant Bit	The right-most bit in a binary number – it has the lowest possible value = 0 or 1
Check Digits	Bits used to ensure that the value sent digitally is not corrupted on transfer
Lossy Compression	Data is removed from the file to make it smaller. This data is lost and cannot be regained. Suitable where the loss of data is likely not to be noticed. Eg images
Lossless Compression	No data is lost, but rather rearranged to ensure a perfect version of the data can be returned. Used where exact reproduction is vital. Eg text documents
JPEG / JPG	Joint Photographic Experts Group Compression for images – lossy
GIF	Graphics Interchange Format Lossless bitmapped image format for limited colours.
PDF	Printable Document Format Open standard for reproducing text and graphic documents without editing permissions – lossless
MPEG	Moving Pictures Expert Group Audio-Visual encoding for video Lossy
MP3	Moving Pictures Expert Group Audio Layer 3 Digital music files. Lossy compression, but very good and generally only removes sounds that are beyond human hearing range

BINARY ADDITION

0	1	0	1
+0	+0	+1	+1
00	01	01	10

carried bit

When adding 2 large binary numbers, if there is not enough bits to take the *carried bit* then this results in an **OVERFLOW ERROR**

```
   1 1 0 0 1 1 0 1
 + 0 1 0 1 1 1 1 0
 -----------------
 1 0 0 1 0 1 0 1 1
```

This value is not counted, it is *overflow*.

In 8 bits, this sum reads : 203 + 94 = 43!

BINARY SHIFT

Multiplication	Binary shift to the LEFT
Division	Binary shift to the RIGHT

By ***moving the bits*** to either the left of the right, we can double (x2) or halve (%2) the value with each movement.

8	4	2	1	
1	0	1	1	=11

A 1 place RIGHT SHIFT (divide by 2) →

8	4	2	1		
0	1	0	1	1	=5

The bits which are moved outside of the available value places are **LOST.** They cannot be returned by reversing the shift. The same is true at the highest place value

8	4	2	1	
1	0	1	1	=11

←

A single LEFT SHIFT (multiply by 2) would result in an overflow error (when represented with 4 bits.)

	8	4	2	1	
1	0	1	1	0	=10

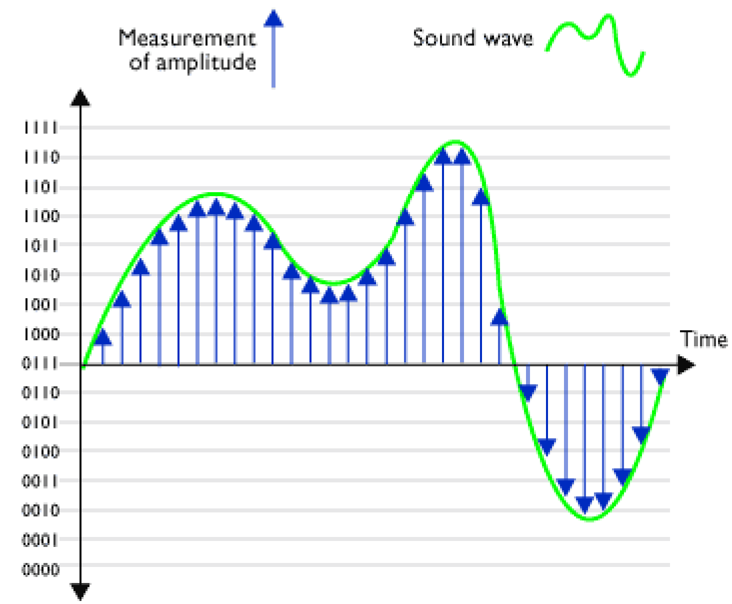
Computing GCSE – 2.6c

J276/02 – Data Representation 3

KEY VOCABULARY

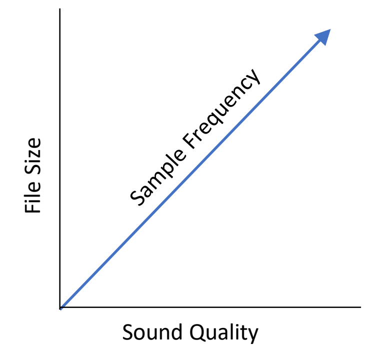
Pixel	Smallest element of an image – the dots that make up an image on a screen
Bitmap	An image where every pixel is ‘mapped’ in binary to show it’s colour, transparency (Alpha) and brightness (Gamma) Increasing size will lower the quality
Vector	An image where the lines are stored as mathematical shapes, so the size can be increased without impacting quality
RGB	Red Green Blue – the order of colour data in a pixel
Colour Depth (bit depth)	The number of bits used to represent each pixel. Shown in bits per pixel (bpp)
Resolution	The number of pixels used per unit eg pixels per inch (ppi)
Metadata	Data about the data – in relation to images, it is the data that allows the computer to recreate the image from it’s binary form.
Analogue	Continuous changing values – no “smallest interval”
Bit Depth	The number of bits used to store the sound
Bit Rate	The number of bits used to store 1 second of sound
Sample Rate	The number of times the sound is sampled in 1 second; measured in kHz (kilohertz or 1000’s per second)

SOUND SAMPLING



Each measurement is assigned a number (byte) according to its amplitude. The end result is a file comprising a string of bytes, eg ...
1001 1110 0001 1010 0111 0100 1111 1101 etc

As the sample rate increases, the quality of the sound goes up – the sound is closer to the analogue original, but the file size also increases. Reduce the sample rate, you reduce quality but also file size.



BIT DEPTH = NUMBER OF COLOURS

Bit depth	Available colours
1 bit (Monochrome)	$2^1 = 2$
2 bits	$2^2 = 4$
3 bits	$2^3 = 8$
8 bits	$2^8 = 256$
16 bits (High Color)	2^{16} = 65,536
24 bits (True Color)	2^{24} = 16.7 million
32 bits (Deep Color)	2^{32} = 4.3 billion

ESTIMATING FILE SIZES

IMAGES:

width X **height** X **colour depth = size**

SOUND:

Nº of channels X **sample rate** X **bit depth**

To get the value into mB, you divide by **1,000,000!**