Data Representation

Converting between number bases

- Bit - is a single binary digit
- Nibble 4 bits
- Byte – is 8 bits

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

What number do you think this is the binary code is for??

Add together 16+8+1

Binary	Decimal	Hex
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

Hexadecimal numbers

(base 16) uses letters A - F for 10 - 15.

Covert HEX A7 to Denary

A 7

10 7

1010 0111

Answer: 167

Exam question: Explain why hexadecimal notation is used.

Hexadecimal is used as shorthand for binary and uses fewer digits, so humans make fewer mistakes and find it easier to read

Exam question: Explain the reason why at least nine bits are needed to store 300 different binary patterns.

29 gives 512 patterns whereas 28 gives 256 patterns
28 does not give enough patterns whereas 29 gives more than enough patterns

Character Encoding

ASCII is a standard that defines how each alphabetical letter is represented by a unique 7 bit binary value (for example 'A' is 100 0001). There are 127 binary values for upper and lowercase letters, the digits and most punctuation.

Extended ASCII uses 8 bits to store 256 characters.

Unicode extends this character set by using more bytes to represent many more characters. (16 to 32 bits)

Binary addition

Binary addition rules:

Addition		Result	Carry
0 + 0	=	0	0
0 + 1	=	1	0
1 + 0	=	1	0
1 + 1	=	0	1

Solution

	0	0	1	1
+	0	0	1	1
	0	1	1	0
Carry		1	1	

Overflow Error – Extra Bit!

```
   1 1 1 1 1 1 1 1
 + 0 0 0 0 0 0 0 1
 -----------------
 1 0 0 0 0 0 0 0 0
   8-bits
```

Binary Shift

Shifts are performed on binary patterns.

Logical shift

Logical Left Shift - doubling the value.
adding zeros on to the right of a binary value and discarding the bits on the left. Multiples the value by 2

Bit	7	6	5	4	3	2	1	0	
	0	0	0	1	1	1	0	1	29
	0	0	1	1	1	0	1	0	58

Logical Right Shift halving the value.

adding zeros on to the left of a binary value and discarding the bits on the right. Divides the value by 2

Bit	7	6	5	4	3	2	1	0	
	0	0	1	1	0	1	0	0	52
	0	0	0	1	1	0	1	0	26

Exam question:

A logical shift right is performed on a pattern.
An arithmetic shift right is performed on the same original pattern.

Describe the reason the results will be different.

An arithmetic shift fills from the left with a copy of the most-significant bit (MSB) whereas a logical shift fills from the left with a 0

An arithmetic shift keeps the most-significant bit (MSB) the same whereas a logical shift always fills the MSB bit with a 0

Arithmetic Shifts

Arithmetic shift left
Exactly the same as a logical shift left

Arithmetic shift right
Copying the most significant bit on to the left of a binary value and discarding the bits on the right. Divides a signed or unsigned number by 2

How to do

To perform an arithmetic shift right of *n* positions, carry out the following steps.

1 Shift each binary digit *n* positions to the right.
2 Discard the rightmost *n* bits.
3 Fill up the empty spaces on the left with a copy of the original MSB.

Worked example

Perform an arithmetic shift right of one position on the binary pattern 1011 0000. Move all the bits one position to the right. Discard the rightmost bit. Put a 1 in the empty space on the left.

Before shift	1	0	1	1	0	0	0	0
After shift	1	1	0	1	1	0	0	0

The result is 1101 1000.

2's Complement

Describe the process of converting a binary number to two's complement.

Copy/keep all the 0s from the right/LSB, up to and including the first 1, then flip the remaining bits

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

2's complement

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

Converting a denary number to binary
Worked example
Convert the denary number –22 to 8-bit binary using two's complement

128 – 22 = 106

106

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

Binary Subtraction

In arithmetic, subtraction can be done by adding a negative number.
Calculate 18 – 8, using 8-bit binary and two's complement.

Step 1 18 converted to 8-bit binary 0001 0010
Steo 2 -8 converted to two's complement 8-bit binary 1111 1000

Step 3 Addition performed:

00010010
11111000
00001010

Result of 0000 1010

Sound

Microphone detects the sound wave and converts it into an electrical (analogue) signal

Sound can be stored as binary data by **sampling** the audio wave **frequency** at regular intervals (the **sample rate**)

Storing the "height" of the wave (the **bit depth**). Typical bit depths are 16 bit and 24 bit. The height of a sound wave is known as the **Amplitude**

Each integer is encoded in binary with a fixed number of bits.

A higher sample rate and a larger bit depth:

Benefit:

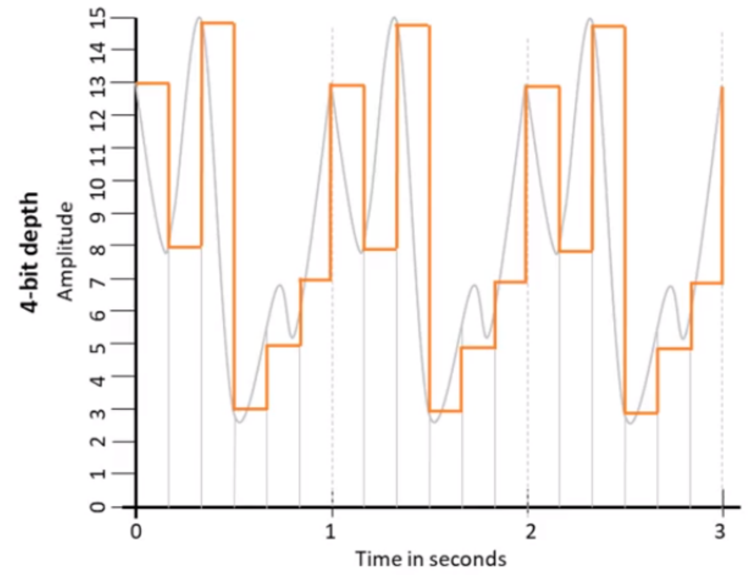
A higher sample rate and a larger bit depth will provide better quality and Increases the accuracy of the representation

Drawback:
• Increases the amount of storage required

• Increases the time it takes to download the audio file

How do you calculate **size of a sound file?**

BIT DEPTH * SAMPLE RATE * LENGTH OF SOUND * CHANNELS



6Hz sample rate

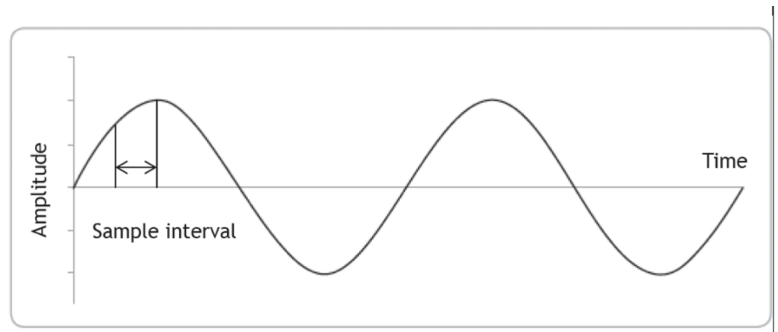
Size of sound file =		
Sample rate	6	Number of samples per second
Duration	3	Length of sample in seconds
Bit depth	4	Number of bits needed to store each sample
6 x 3 x 4	=	**72 bits =** 9 bytes

The **bit rate** of an audio file is the:

sample rate * bit depth * channels

Worked Example:

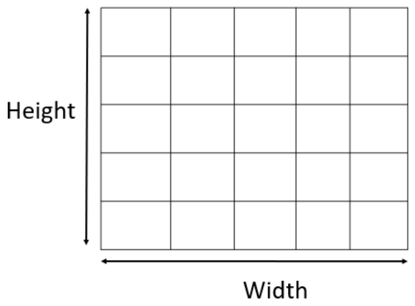
An analogue to digital converter is used to change the sounds received by
a microphone into a form that can be processed by a computer.
Complete the diagram to show a sample interval and label both axes.



Images

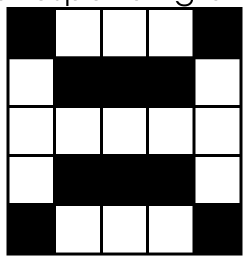
Images are stored as a **bitmap** of **pixels**.

Image Resolution
Total number of pixels in image = width in pixels x height in pixels



Colour depth is the number of bits used to represent each pixel in an image. If we have a black and white image it has two colours. Each pixel can be represented by a single pixel because a bit value of 0 is black and 1 is white.

Image and corresponding binary encoding



0111010001111111000101110

To represent more colours we can use more bits:
2-bits per pixel - 4 colours
8-bits = 256 colours
24 bts = 16 Million

Calculating the size of an image

pixel height * width * colour depth

Larger **resolutions** (the pixel height x width) and **bit depth** (number of bits used to store each colour value) will provide better quality images - but much larger file sizes.

Metadata:

Filename
file format - eg JPG, GIF or PNG
dimensions
resolution
colour depth
time and date the image was last changed
camera settings when the photo was taken

Data Compression

Compression reduces data used to represent the original sound, image or text document.

The purpose of data compression is to make the files smaller which means that:

- Less time / less bandwidth to transfer data
- Take up less space on the disk

Both types require encoding and decoding

Lossless compression means that as the file size is compressed, quality remains the same - the file can be decompressed to its original quality. Text documents (or spreadsheets etc) must only use **lossless compression**.

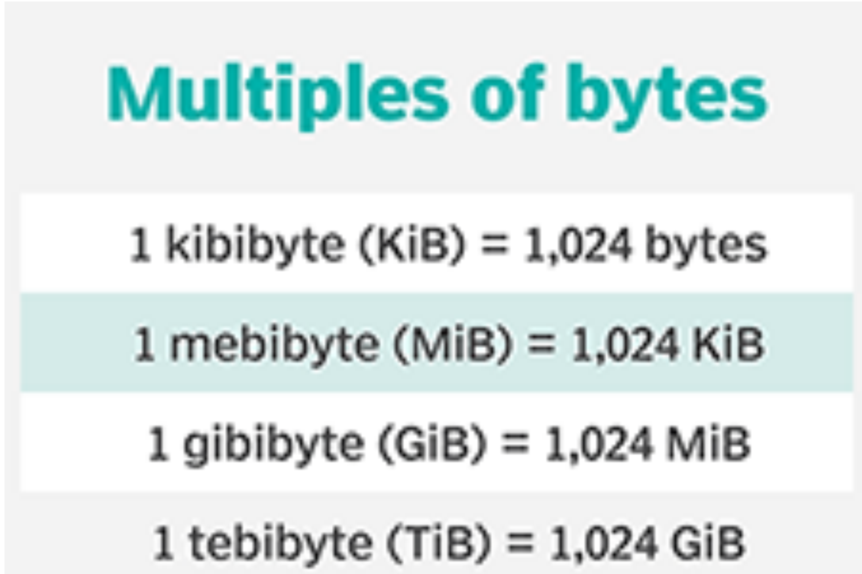
- Lossless better for physical media (CDs)

Lossy compression permanently removes data. This is usually fine for images, movies and audio.

- Lossy compression reduces the accuracy of the representation
- Lossy compression increases the reduction in file size.
- Lossy better for online transmission - streaming technologies as it takes less time to download / can facilitate access by users with low-speed connections.
- Lossy better for cases where limited storage available,
- Lossy audio removes data representing frequencies / visuals that humans cannot hear / see so they cannot tell the difference
- Lossy compression can be variable so that different amounts of compression can be offered depending on a user's bandwidth.

Data Sizes

Data Sizes



Multiples of bytes

1 kibibyte (KiB) = 1,024 bytes
1 mebibyte (MiB) = 1,024 KiB
1 gibibyte (GiB) = 1,024 MiB
1 tebibyte (TiB) = 1,024 GiB

Worked Example:

Construct an expression to show how many bytes there are in 6 tebibytes.

$$\frac{64 \times 48 \times 12}{1024 \times 1024 \times 8}$$

Worked Example:

Construct an expression to calculate the file size, in **mebibytes**, of a CD quality (44.1 KHz, bit depth of 16), two-channel stereo soundtrack that is 4 minutes long.

- Sample rate and bit depth = 44.1 x 1000 x 16
- Channels and time = 2 x 4 x 60
- Unit conversions = 8 x 1024 x 1024

Example of an expression that gains full marks:
((44.1 x 1000 x 16) x (2 x 4 x 60)) / (8 x 1024 x 1024)