OCR GCSE Computer Science

Your notes

Data Storage & Compression

Contents

Units of Data Storage

Your notes

Units of Data Storage

- Computers use binary numbers to **represent data**
- Data such as characters, images and sound must be **stored as binary**
- The smallest unit of data a computer can store is 1 **b**inary dig**it,** otherwise expressed as **1 bit**
- 1 bit can only hold one value (2^1), this is **not big enough** to store all kinds of data, so computers have different 'Units of Data'

What are units of data?

- A unit of data is a **term** given to describe **different amounts of binary digits** stored on a digital device
- These are the units you need to know for GCSE:

Unit	Symbol	Binary	Written as	Example
Bit	b	1 or 0		
Nibble		4 b		
Byte	B	8 b		A single character
Kilobyte	KB	1000 B (2^{10})	Thousand bytes	A small text file
Megabyte	MB	1000 KB (2^{20})	Million byes	A music file
Gigabyte	GB	1000 MB (2^{30})	Billion bytes	A high definition movie
Terabyte	TB	1000 GB (2^{40})	Trillion bytes	A large hard drive
Petabyte	PB	1000 TB (2^{50})	Quadrillion bytes	A large data centre

Examiner Tips and Tricks

Your notes

Binary only contains two digits (1 and 0) so technically larger multiples would be calculated as $2^{\text{number of bytes}}$

For example, a kilobyte is 2^{10} = 1024 bytes not 1000 bytes

In GCSE we approximate all larger units of storage as multiples of 1000 to make calculations easier

Converting between units

- It is often a requirement of the exam to be able to convert between different units of data, for example bytes to megabytes (larger) or kilobytes to bytes (smaller)
- This process involves **division**, **moving up in size** of unit and **multiplication**, **moving down in size** of unit
- When dealing with all units **bigger than a byte** we use multiples of **1000**
- For example, 2000 kilobytes in megabytes would be 2000 / 1000 = **2 MB** and 2 terabytes in gigabytes would be 2 * 1000 = **2000 GB**
- When dealing with **bits and bytes** the same process is used with the value **8** as there are 8 bits in a byte
- For example, 24 bits in bytes would be 24 / 8 = **3 B** and 10 bytes in bits would be 10 * 8 = **80 b**

	Unit	
Multiply by 8 ⬆	Bit	Divide by 8 ⬇
	Byte	
Multiply by 1000 ⬆	Kilobyte	Divide by 1000 ⬇
	Megabyte	
	Gigabyte	
	Terabyte	
	Petabyte	

Worked Example

Computers represent data in binary form.

Tick one box in each row to identify the binary unit equivalent of each of the given file sizes **[4]**

File size	4 megabytes	24 bits	5 kilobytes	10 bytes	2 terabytes
2000 gigabytes					
5000 bytes					
6 nibbles					
3 bytes					

Answer

File size	4 megabytes	24 bits	5 kilobytes	10 bytes	2 terabytes
2000 gigabytes					
5000 bytes					
6 nibbles					
3 bytes					

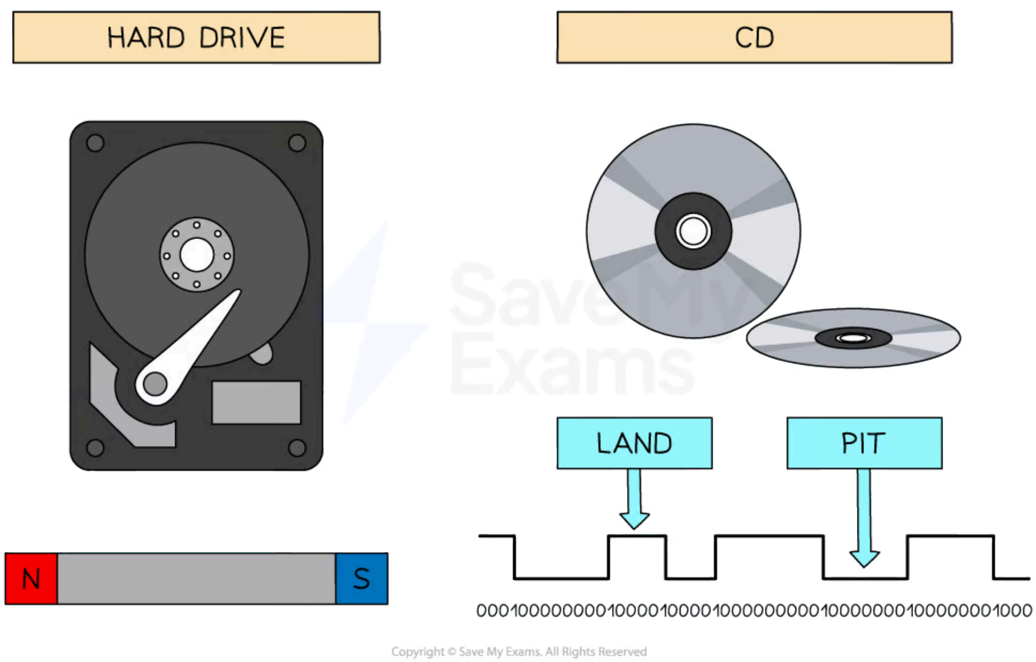
Your notes

Your notes

Processing Binary Data

Why does data have to be converted to binary to be processed by a computer?

- A computer is built using **switches** that can either be **on or off**, this fits the binary number system which only has two digits (1/0), which means off can be represented by a 0 and on can be represented by a 1
- This means all data must be c**onverted to binary** before a computer can **understand** and **process** it
- Converting data to binary allows computers to process it at an incredible **speed**, perform complex **calculations** and **store** vast amounts of **data efficiently**



- Examples of where you see this process is in secondary storage, in **magnetic hard drives** they use **North** and **South polarity** to represent a 1 or a 0 and in **optical disks** light is reflected back to the surface or not, 1 and 0
- Take an example of driving a car
  - When driving a car the accelerator pedal is used to increase the cars speed

- If a car was accelerating from 50mph to 100mph the increase would be gradual
- In a computer system, the car is doing either 50mph (0) or 100mph (1), there is no in-between
- Trying to change the computer system so that it has more options would be less efficient and require more complex parts for the computer to understand

Worked Example

Explain why computers process data in binary format **[2]**

Answer

- Computers consist of switches/transistors **[1]**
- 1 is represented as switch/transistor being on/open // 0 is represented as switch/transistor being off/closed **[1]**

Your notes

Your notes

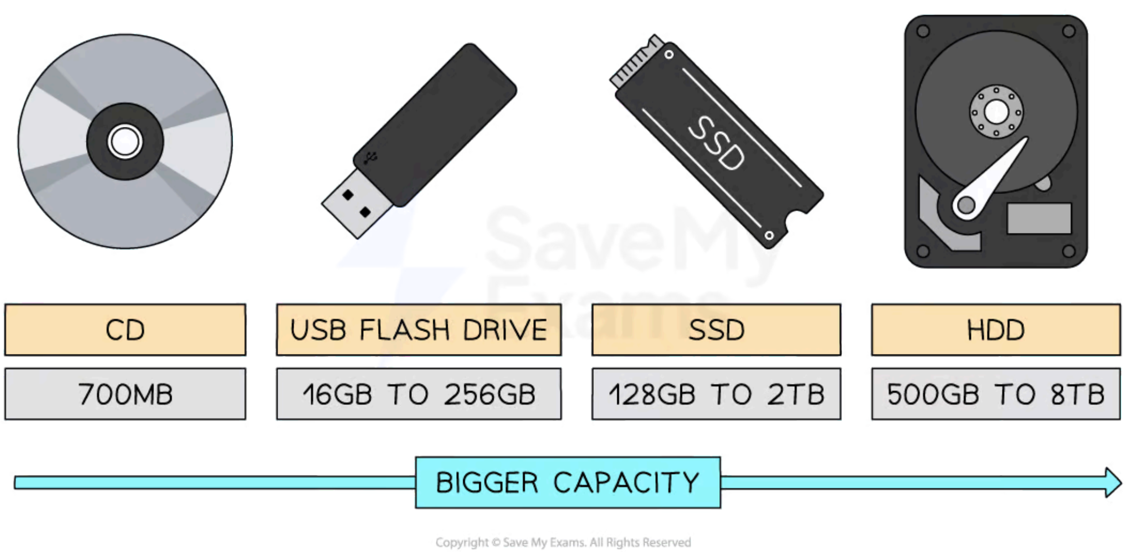
Data Capacity & Calculating Capacity Requirements

What is data capacity?

- Data capacity is the **maximum amount of information** that a **storage device can hold**
- If you know the capacity of a storage device, you can calculate how much of different data types can be stored
- Examples of data that could be stored include text, images and sound files

How do you calculate capacity requirements?

- To calculate capacity requirements you need to:
  - Know the **capacity** of the storage device
  - Calculate the **size of a file** (formula)
  - Ensure the same units of data storage are used for **capacity** and **size of the file**, else convert between units to get them to be the same
  - **Divide** the **capacity** by **size of the file** being stored



- The image above shows common capacities of storage devices such as hard drives (HDD), USB flash drives, solid state drives (SSD) and optical disks (CD)

Your notes

Calculating file sizes

Text Files
(Bytes per character) **x** (Number of characters)

Size of text file =		
Bytes per character	**1**	Based on Extended ASCII character set being used (**8 bits**)
Number of characters	**2000**	Including spaces!
1 x 2000	**=**	**2000 bytes** = **2 KB**

Image Files
(Resolution) **x** (Colour Depth)

Size of image file =		
Resolution	**500 x 500**	Resolution = Width x Height
Colour Depth	**24 bits**	24 bits = 3 bytes
(500×500) x 24 **(500×500) x 3**	**=** **=**	**6,000,000 bits** **750,000 bytes** = **75 KB**

Sound Files
(Sample Rate) **x** (Duration in seconds) **x** (Bit Depth)

Size of sound file =		
Sample Rate	**10**	Samples per second
Duration	**30**	Seconds
Bit Depth	**4**	Number of bits stored per sample
10 x 30 x 4	**=**	**1200 bits = 150 bytes**

Your notes

Worked Example

Lyla is a social media influencer. She creates images to be shared on her social media accounts.

Each image has a fixed size of 3 MB. She is storing the images on a USB flash drive which has a capacity of 6 GB.
Calculate how many images can be saved on the storage device. Show your working **[2]**

How to answer this question:

- Convert 6 GB to MB
- Divide the capacity in MB by the files size in MB

Answer

- 6 GB = 6 * 1000 = **6000 MB**
- 6000 / 3 = **2000 images**

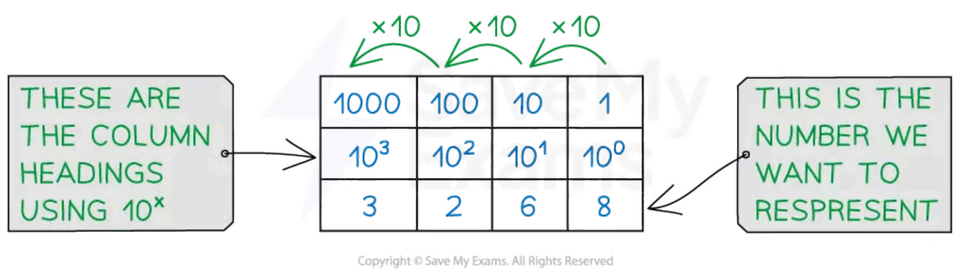
Your notes

Converting Between Denary & Binary

Denary to Binary Conversion

What is denary?

- Denary is a number system that is made up of **10 digits (0-9)**
- Denary is referred to as a **Base-10** number system
- Each digit has a weight factor of **10 raised to a power**, the rightmost digit is 1s (10^0), the next digit to the left 10s (10^1) and so on
- Humans use the denary system for **counting**, **measuring** and **performing maths calculations**
- Using combinations of the 10 digits we can represent any number

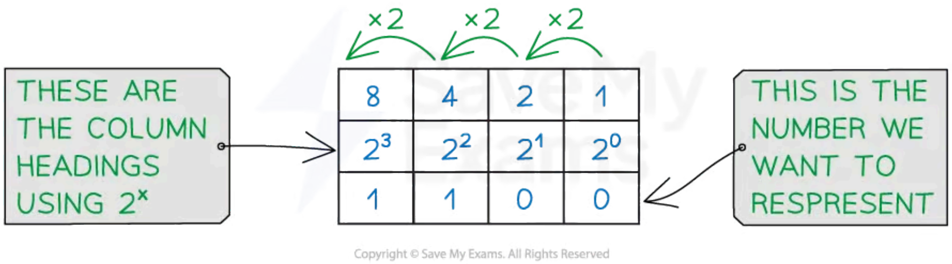


- In this example, (3 x 1000) + (2 x 100) + (6 x 10) + (8 x 1) = 3268
- To represent bigger number we add more digits

What is binary?

- Binary is a number system that is made up of **two digits (1 and 0)**
- Binary is referred to as a **Base-2** number system
- Each digit has a weight factor of **2 raised to a power**, the rightmost digit is 1s (2^0), the next digit to the left 2s (2^1) and so on
- Using combinations of the 2 digits we can represent any number

Your notes



- In this example, (1 x 8) + (1 x 4) = 12
- To represent bigger numbers we add more **b**inary dig**its** (bits)

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

Why do computers use binary?

- The CPU is made up of billions of tiny **transistors**, transistors can only be in a state of on or off
- Computers use binary numbers to **represent data** (1 = on, 0 = off)

Examiner Tips and Tricks

Don't forget to show your working! Data conversion questions will often be worth 2 marks, 1 for the answer and 1 for your working

Denary to binary conversion

- It is important to know the process of converting from denary to binary to understand how computers interpret and process data

Example 1

- To convert the denary number 45 to binary, start by **writing out the binary headings** from right to left

128	64	32	16	8	4	2	1

Your notes

- Start at the **leftmost empty column heading** (128). Is the **denary number > column heading?** (45 > 128) No, put a 0 in the 128 column. Repeat until you put a 1 under a heading. In this example it would be 32

128	64	32	16	8	4	2	1
0	0	1					

- Next **subtract column heading from denary value**, 45–32 = 13
- Repeat previous two steps until you have a value under each column heading

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

- 32 + 8 + 4 + 1 = 45
- Denary 45 is **00101101** in Binary

Examiner Tips and Tricks

At GCSE you will only be asked to convert from/to binary up to and including 8 binary digits (8 bits). That means you are working with a denary range of 0–255 (00000000–11111111)

Example 2

- To convert the denary number 213 to binary, start by **writing out the binary headings** from right to left

128	64	32	16	8	4	2	1

- Start at the **leftmost empty column heading** (128). Is **denary number > column heading?** (213 > 128) Yes, put a 1 under the heading.

128	64	32	16	8	4	2	1
1							

Your notes

- Next **subtract column heading from denary value**, 213–128 = 85
- Repeat process until you have a value under each column heading

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

- 128 + 64 + 16 + 4 + 1 = 213
- Denary 213 is **11010101** in Binary

Binary to Denary Conversion

Example 1

- To convert the binary number 1011 to denary, start by **writing out the binary headings** from right to left

8	4	2	1

- Write in the binary digits under the headings from left to right

8	4	2	1
1	0	1	1

- Add together **any value with a 1 under it**
- (1 x 8) + (1 x 2) + (1 x 1) = 11
- Binary 1011 is **11** in Denary

Your notes

Examiner Tips and Tricks

If you are converting from binary to denary and the binary number ends in 1, the denary answer must be an odd number!

Example 2

- To convert the binary number 01100011 to denary, start by **writing out the binary headings** from right to left

128	64	32	16	8	4	2	1

- Write in the binary digits under the headings from left to right

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

- Add together **any value with a 1 under it**
- (1 x 64) + (1 x 32) + (1 x 2) + (1 x 1) = 99
- Binary 01100011 is **99** in Denary

Your notes

Binary Addition

Binary Addition

What is binary addition?

- Binary addition is the process of **adding together two binary integers** (up to and including 8 bits)
- To be successful there are **5 golden rules** to apply:

<table>
<tr><th>Binary Addition</th><th>Binary Answer</th><th>Working</th></tr>
<tr><td>0 + 0 =</td><td>0</td><td><table><tr><td>1s</td><td></td></tr><tr><td>0</td><td>= 0</td></tr></table></td></tr>
<tr><td>0 + 1 =</td><td>1</td><td><table><tr><td>1s</td><td></td></tr><tr><td>1</td><td>= 1</td></tr></table></td></tr>
<tr><td>1 + 0 =</td><td>1</td><td><table><tr><td>1s</td><td></td></tr><tr><td>1</td><td>= 1</td></tr></table></td></tr>
<tr><td>1 + 1 =</td><td>10</td><td><table><tr><td>2s</td><td>1s</td><td></td></tr><tr><td>1</td><td>0</td><td>= 2</td></tr></table></td></tr>
<tr><td>1 + 1 + 1 =</td><td>11</td><td><table><tr><td>2s</td><td>1s</td><td></td></tr><tr><td>1</td><td>1</td><td>= 3</td></tr></table></td></tr>
</table>

Your notes

- Like denary addition, start from the rightmost digit and move left
- Carrying over occurs when the **sum of a column is greater than 1**, passing the excess to the next left column

Example 1

- Add together the binary values 1001 and 0100

8	4	2	1	+
1	0	0	1	
0	1	0	0	
				C

- Starting from right to left, add the two binary values together **applying the 5 golden rules**
- If your answer has **2 digits**, place the rightmost digit in the column and **carry the remaining digit** to the next column on the left
- In this example, start with 1+0, 1+0 = 1, so place a 1 in the column

8	4	2	1	
1	0	0	1	+
0	1	0	0	
				C
			1	

- Repeat until **all columns have a value**

8	4	2	1	
1	0	0	1	+
0	1	0	0	
				C
1	1	0	1	

Your notes

- The sum of adding together binary 1001 (9) and 0100 (4) is **1101** (13)

Examiner Tips and Tricks

Make sure any carried digits are clearly visible in your answer, there are marks available for working. Carries can be put above or below in the addition

Example 2

- Add together the binary values 00011001 and 10001001

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

- Starting from right to left, add the two binary values together **applying the 5 golden rules**
- If your answer has **2 digits**, place the rightmost digit in the column and **carry the remaining digit** to the next column on the left
- In this example, start with 1+1, 1+1 = 10, so place a 0 in the column and carry the 1 to the next column

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

- Repeat until **all columns have a value**

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

- The sum of adding together binary 00011001 (25) and 10001001 (137) is **10100010** (162)

What is an overflow error?

- An overflow error occurs when the result of a binary addition exceeds the available bits
- For example, if you took binary 11111111 (255) and tried to add 00000001 (1) this would cause an overflow error as the result would need a 9th bit to represent the answer (256)

256	128	64	32	16	8	4	2	1	
	1	1	1	1	1	1	1	1	+
	0	0	0	0	0	0	0	1	
1	1	1	1	1	1	1	1		C
	0	0	0	0	0	0	0	0	

Examiner Tips and Tricks

When starting a binary addition question, always look at the question that comes after. If it asks you to name what problem has been caused, you know the binary addition question must cause an overflow error and therefore mean a carried bit that does not fit into the answer

Your notes

Converting Between Denary & Hexadecimal

Denary to Hexadecimal Conversion

What is hexadecimal?

- Hexadecimal is a number system that is made up of **16 digits**, 10 numbers (0–9) and 6 letters (A-F)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

- Hexadecimal is referred to as a **Base-16** number system
- Each digit has a weight factor of **16 raised to a power**, the rightmost digit is 1s (16^0), the next digit to the left 16s (16^1)
- In GCSE you are required to work with up to and including **2 digit hexadecimal** values

16s	1s	
1	**3**	
1 x16	3 x 1	= 19

- A quick comparison table demonstrates a **relationship** between **hexadecimal and a binary nibble**
- One hexadecimal digit can represent **four bits** of binary data

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

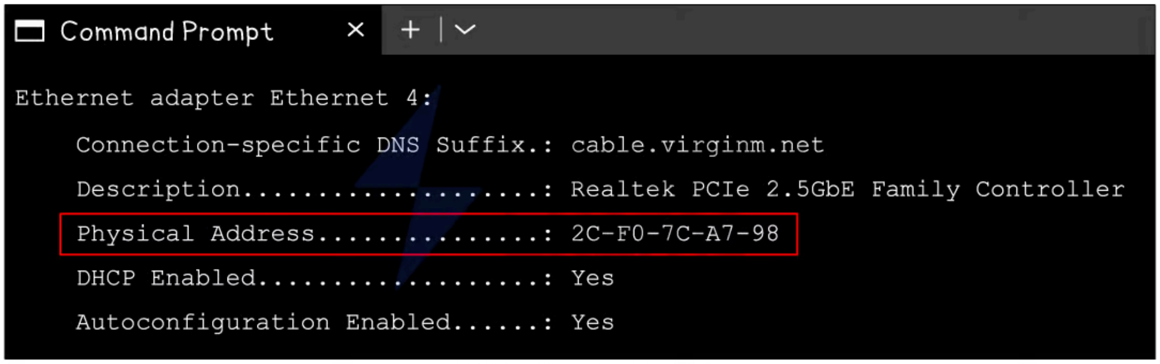
Your notes

Examiner Tips and Tricks

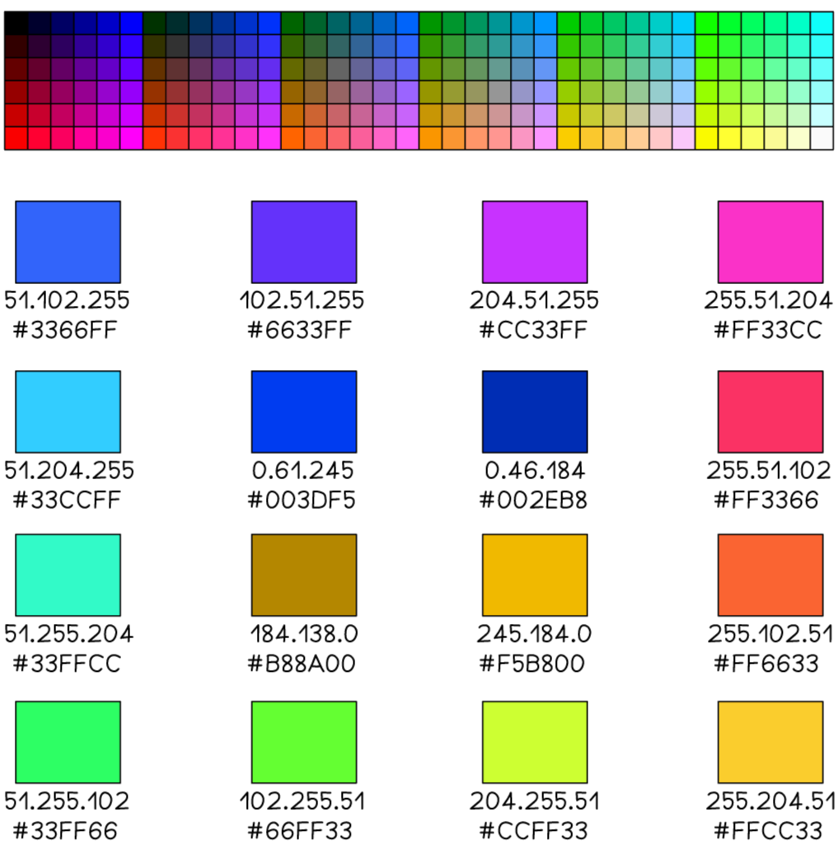
A common exam mistake is mixing up which letter matches with what number, write out the 16 hexadecimal digits at the start of the exam!

Why is hexadecimal used?

- In Computer Science hexadecimal is often preferred when working with **large values**
- It takes fewer digits to represent a given value in hexadecimal than in binary
- It is beneficial to use hexadecimal over binary because:
  - The more bits there are in a binary number, the **harder it is to read**
  - Numbers with more bits are more **prone to errors** when being copied
- Examples of where hexadecimal can be seen:
  - MAC addresses



- Colour values



How do you convert denary to hexadecimal?

Your notes

Your notes

Method 1 (denary to binary to hexadecimal)

- To convert the denary number 28 to hexadecimal, start by converting the denary number to binary

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

- **Split** the 8 bit binary number into **two nibbles** as shown below

8	4	2	1	8	4	2	1
0	0	0	1	1	1	0	0

- Convert each nibble to its denary value
- **0001 = 1** and **1100 = 12**
- Using the comparison table, the denary value 1 is also 1 in hexadecimal whereas denary value 12 is represented in hexadecimal as C
- Denary **28** is **1C** in hexadecimal

Method 2 (divide by 16)

- To convert the denary number 163 to hexadecimal, start by **dividing the denary** value by 16 and recording the **whole times** the number goes in and the **remainder**
- 163 $\div$ 16 = 10 remainder 3
- In hexadecimal the **whole number = digit 1** and the **remainder = digit 2**
- Digit 1 = 10 (A)
- Digit 2 = 3
- Denary 163 is **A3** in hexadecimal

Hexadecimal to Denary Conversion

How do you convert hexadecimal to denary?

Method 1 (hexadecimal to binary to denary)

- To convert the hexadecimal number B9 to denary, take each hexadecimal value and convert it as **denary to 4 bit binary**

Your notes

B (11)				9			
8	4	2	1	8	4	2	1
1	0	1	1	1	0	0	1

- **Join** the two nibbles to make an **8 bit number** (byte)
- Convert from binary to denary

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

- (1 x 128) + (1 x 32) + (1 x 16) + (1 x 8) + (1 x 1) = 185
- Hexadecimal B9 is **185** in denary

Method 2 (multiply by 16)

- To convert the hexadecimal number 79 to denary, start by **multiplying the first hexadecimal digit by 16**
- 7 $\times$ 16 = 112
- **Add digit 2** to the result
- 112 + 9 = 121
- Hexadecimal 79 is **121** in denary

Examiner Tips and Tricks

Remember that the exam is non-calculator, if you are not confident multiplying and dividing by 16 then use method 1 on both conversions

Your notes

Converting Between Binary & Hexadecimal

Binary to Hexadecimal Conversion

- It is important before revising how to convert from binary to hexadecimal and vice versa that you fully understand the binary and hexadecimal number systems.

How do you convert from binary to hexadecimal?

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Example 1

- To convert the binary number 10110111 to hexadecimal, first split the 8 bit number into 2 binary **nibbles**

8	4	2	1		8	4	2	1
1	0	1	1		0	1	1	1

- For each nibble, convert the binary to it's denary value
- (1 x 8) + (1 x 2) + (1 x 1) = 11 (B)
- (1 x 4) + (1 x 2) + (1 x 1) = 7
- Join them together to make a 2 digit hexadecimal number
- Binary 10110111 is **B7** in hexadecimal

Example 2

- To convert the binary number 00111001 to hexadecimal, first split the 8 bit number into 2 binary nibbles

8	4	2	1		8	4	2	1
0	0	1	1		1	0	0	1

- For each nibble, convert the binary to it's denary value
- (1 x 2) + (1 x 1) = 3

- (1 x 8) + (1 x 1) = 9
- Join them together to make a 2 digit hexadecimal number
- Binary 00111001 is **39** in hexadecimal

Hexadecimal to Binary Conversion

How do you convert from hexadecimal to binary?

Example 1

- To convert the hexadecimal number 5F to binary, first **split the digits apart** and convert each to a binary **nibble**

8	4	2	1	
0	1	0	1	**= 5**

8	4	2	1	
1	1	1	1	**= 15 (F)**

- Join the 2 binary nibbles together to create an 8 bit binary number

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

- Hexadecimal 5F is **01011111** in binary

Example 2

- To convert the hexadecimal number 26 to binary, first **split the digits apart** and convert each to a binary **nibble**

8	4	2	1	
0	0	1	0	**= 2**

Your notes

8	4	2	1	
0	1	1	0	**= 6**

- Join the 2 binary nibbles together to create an 8 bit binary number

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

- Hexadecimal 26 is **00100110** in binary

Your notes

Your notes

Binary Shifts

Binary Shifts

What is a binary shift?

- A binary shift is how a computer system performs **basic multiplication and division**
- Binary digits are **moved left** or **right** a set number of times
- A left shift **multiplies** a binary number by 2 (x2)
- A right shift **divides** a binary number by 2 (/2)
- A shift can move **more than one place at a time**, the principle remains the same
- A **left shift of 2 places** would **multiply the original binary number by 4** (x4)
- Binary shifts can cause **a loss of precision by discarding bits**, which can lead to changes in the numerical value

How do you perform a left shift of 1?

- Here is the binary representation of the denary number 40

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

- To perform a left binary shift of 1, we move each bit 1 place to the left
- The digit in the 128 column will move left causing an **overflow error**
- The 1 column becomes empty so is filled with a 0

128	64	32	16	8	4	2	1	
	0	1	0	1	0	0	0	**= 40**
0	1	0	1	0	0	0	0	**= 80**

- The original binary representation of denary **40** (32+8) has **multiplied by 2** and became **80** (64+16)

How do you perform a left shift of 2?

- Here is the binary representation of the denary number 28

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

- To perform a left binary shift of 2, we move each bit 2 place to the left
- The digit in the 128 and 64 column will move left causing an **overflow error**
- The 1 and 2 column become empty so are filled with a 0

128	64	32	16	8	4	2	1	
		0	1	1	1	0	0	**= 28**
0	1	1	1	0	0	0	0	**= 112**

- The original binary representation of denary **28** (16+8+4) has **multiplied by 4** and became **112** (64+32+16)

How do you perform a right shift of 1?

- Here is the binary representation of the denary number 40

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

- To perform a right binary shift of 1, we move each bit 1 place to the right
- The digit in the 1 column will move right causing an **underflow error**
- The 128 column becomes empty so is filled with a 0

128	64	32	16	8	4	2	1	
0	0	1	0	1	0	0		**= 40**

Your notes

0	0	0	1	0	1	0	0	**= 20**

- The original binary representation of denary **40** (32+8) has **divided by 2** and became **20** (16+4)

How do you perform a right shift of 2?

- Here is the binary representation of the denary number 200

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

- To perform a right binary shift of 2, we move each bit 2 places to the right
- The digits in the 1 and 2 columns will move right causing an underflow error
- The 128 and 64 columns become empty so are filled with a 0

128	64	32	16	8	4	2	1	
1	1	0	0	1	0			**= 200**
0	0	1	1	0	0	1	0	**= 50**

- The original binary representation of denary **200** (128+64+8) has **divided by 4** and became **50** (32+16+2)

Worked Example

1. Perform a binary shift of 2 places left on the binary number 00001110 **[1]**

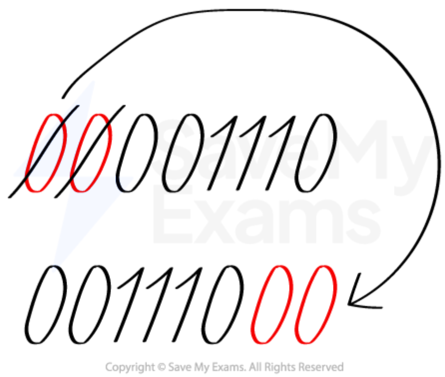
2. Explain the effect of performing a 2 place shift to the left on the binary number 00001110 **[2]**

Answers

Q1

Your notes

Your notes



- Cross out the first 2 digits from the left
- Write down the binary digits left and add 2 zeros to the end

Q2

- Multiplies the number by 4
- Overflow errors can cause loss of precision

Representing Characters

Your notes

Character Sets

- Computers represent all data in binary, including characters that are input using a keyboard
- 1 binary digit (bit) would allow us to represent only two possible characters, for example 1=A and 0=B
- Using more bits allows more characters to be represented, 2 bits or 22 = 4 and so on

What is a character set?

- A character set is **a defined list of characters** that can be understood by a computer
- Each character is given a **unique** binary code
- Character sets are **ordered logically**, the code for 'B' is one more than the code for 'A'
- A character set provides a **standard** for computers to communicate and send/receive information
- Without a character set, one system might interpret 01000001 differently from another
- 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

What is ASCII?

- ASCII is a character set and was an accepted standard for information interchange
- ASCII uses **7 bits**, providing 27 unique codes (128) or a maximum of **128 characters** it can represent

Your notes

Letter	ASCII Code	Binary	Letter	ASCII Code	Binary
a	097	01100001	A	065	01000001
b	098	01100010	B	066	01000010
c	099	01100011	C	067	01000011
d	100	01100100	D	068	01000100
e	101	01100101	E	069	01000101
f	102	01100110	F	070	01000110
g	103	01100111	G	071	01000111
h	104	01101000	H	072	01001000
i	105	01101001	I	073	01001001
j	106	01101010	J	074	01001010
k	107	01101011	K	075	01001011
l	108	01101100	L	076	01001100
m	109	01101101	M	077	01001101
n	110	01101110	N	078	01001110
o	111	01101111	O	079	01001111
p	112	01110000	P	080	01010000
q	113	01110001	Q	081	01010001
r	114	01110010	R	082	01010010
s	115	01110011	S	083	01010011
t	116	01110100	T	084	01010100
u	117	01110101	U	085	01010101
v	118	01110110	V	086	01010110
w	119	01110111	W	087	01010111
x	120	01111000	X	088	01011000

x	120	01111000	X	088	01011000
y	121	01111001	Y	089	01011001
z	122	01111010	Z	090	01011010

- ASCII only represents basic characters needed for **English**, limiting its use for other languages

Extended ASCII

- Extended ASCII uses **8 bits**, providing 256 unique codes (28 = 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 ©

Limitations of ASCII & extended ASCII

- ASCII has a **limited** number of characters which means it can only represent the **English alphabet**, **numbers** and some special characters
    - A, B, C, ………, Z
    - a, b, c ,.............,z
    - 0, 1, 2,........, 9
    - !, @, #, ...
- ASCII **cannot** represent characters from languages other than English
- ASCII **does not** include modern symbols or emojis common in today's digital communication

Examiner Tips and Tricks

The binary representation of ASCII in the exam will be 8 bits (Extended ASCII)

UNICODE

What is UNICODE?

- UNICODE is a **character set** and was created as a solution to the limitations of ASCII
- UNICODE uses a minimum of **16 bits**, providing 216 unique codes (65,536) or a minimum of **65,536 characters** it can represent

Your notes

- UNICODE can represent characters from **all the major languages** around the world

Your notes

Examiner Tips and Tricks

Exam questions often ask you to compare ASCII & UNICODE, for example the number of bits, number of characters and what they store

ASCII vs UNICODE

	ASCII	UNICODE
Number of bits	7-bits	16-bits
Number of characters	128 characters	65,536 characters
Uses	Used to represent characters in the English language.	Used to represent characters across the world.
Benefits	It uses a lot less storage space than UNICODE.	It can represent more characters than ASCII. It can support all common characters across the world. It can represent special characters such as emoji's.
Drawbacks	It can only represent 128 characters. It cannot store special characters such as emoji's.	It uses a lot more storage space than ASCII.

Worked Example

The computer stores text using the ASCII character set.

Part of the ASCII character set is shown:

Character	**ASCII Denary Code**
E	69
F	70
G	71
H	72

Identify the character that will be represented by the ASCII denary code 76 **[1]**

L (must be a capital)

Identify a second character set **[1]**

UNICODE

Your notes

Your notes

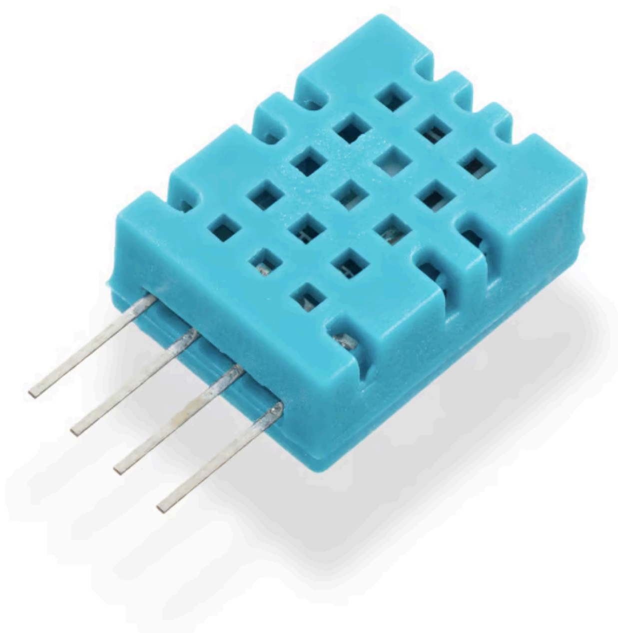
Representing Images

Vectors & Bitmaps

- Computers represent all data in binary, including images that are seen on a screen, TV or other output device
- Images can be stored in binary as **Bitmap** or **Vector**

What is a bitmap?

- A bitmap image is made up of squares called **pixels**
- 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

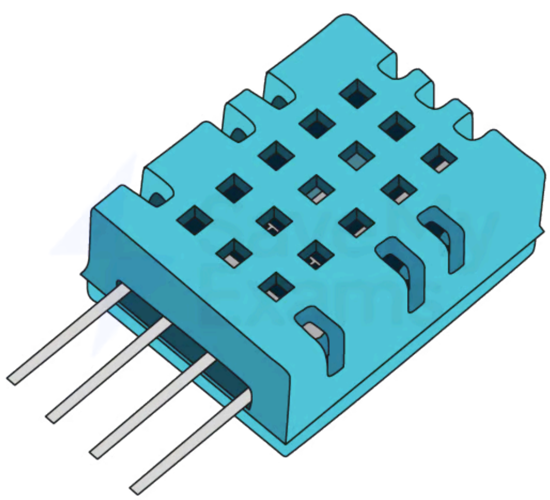


- The more colours and more detail in the image, the higher the quality of the image and the more binary that needs to be stored

Your notes

What is a vector?

- A vector image is created from **mathematical equations** and **points**
- Only the mathematics used to create the image are stored
- For example, to create a circle the data stored would be:
  - Centre point (x, y coordinates)
  - Radius
- Typical examples of vector images are logos and clipart



- Vector images are infinitely **scalable**
- Ideal for situations where the same image **will be made bigger and smaller** and a **loss of quality is unacceptable**. For example, the same logo used on both a pencil and a billboard

Examiner Tips and Tricks

In the exam, the focus will be on bitmap images. You need to know the basics of a vector but after that all questions will relate to bitmaps only!

Your notes

Resolution & Colour Depth

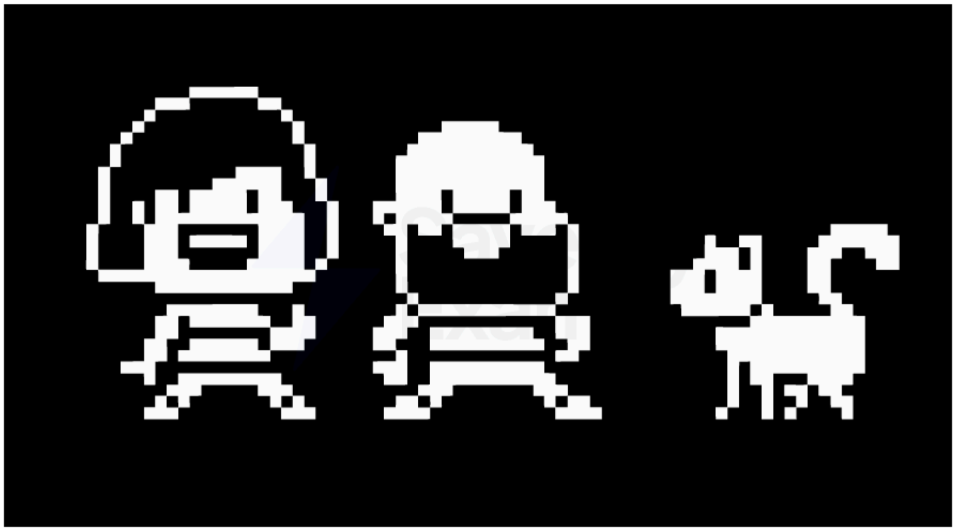
What is resolution?

- Resolution is the **total amount of pixels** that make up a bitmap image
- The resolution is calculated by **multiplying the height and width** of the image (in pixels)
- In general, the **higher the resolution the more detail in the image** (higher quality)
- Resolution can also refer to the total amount of pixels horizontally in a display, such as:
  - **Computer monitors** - 1440p means 1440 pixels horizontally compared to 4K which is 3840 pixels (roughly 4 thousand)
  - **TVs** - HD (high definition) channels have a resolution of 1080p, 1080 pixels horizontally compared to newer UHD (ultra high definition) channels with 3840 pixels (4K)
  - **YouTube** - The quality button allows a user to change the video playback resolution from 144p (144 pixels horizontally) up to 4K

What is colour depth?

- Colour depth is the **number of bits stored per pixel** in a bitmap image
- The colour depth is dependent on the number of colours needed in the image
- In general, the **higher the colour 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)

Your notes



- In an image with a colour depth of 2, you would have 00, 01, 10 & 11 available binary codes, so 4 colours

- As colour depth increases, so does the amount of colours available in an image
- The amount of colours can be calculated as 2^n (n = colour depth)

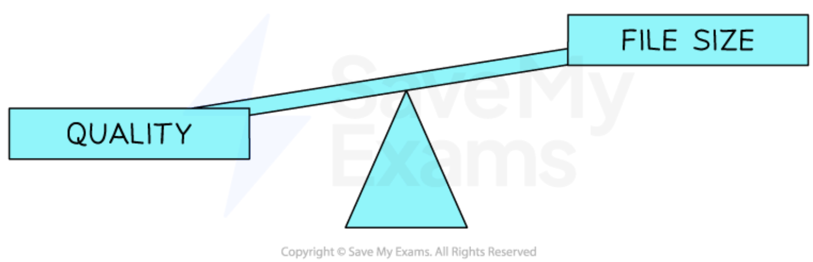
Colour Depth	Amount of Colours
1 bit	2 (B&W)
2 bit	4

4 bit	16
8 bit	256
24 bit	16,777,216 (True Colour)

Your notes

What is the impact of resolution and colour depth?

- 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
- Striking a balance between quality and file size is always a consideration



Worked Example

1. Define the term Pixel **[1]**

2. If an image has a colour depth of 2 bits, how many colours can the image represent? **[1]**

3. Describe the impact of changing an images resolution from 500×500 to 1000×1000 **[2]**

Answers

1. The smallest element of a bitmap image (1 square)
2. 4
3. The image quality would be higher **[1]** the file size would be larger **[1]**

Your notes

Metadata

What is metadata?

- Metadata is **data about data**
- Metadata is **additional information stored with the image**, although not required to display the image it provides context and information
- Examples of metadata that can be stored are:
  - **Author** - Who created the image?
  - **Date/Time** - When and what time was the image created/taken?
  - **Location** - Where was the image taken?
  - Width & height of the image (**resolution**)
  - **Colour depth**

Worked Example

A parent takes a photograph of their family whilst on holiday. The image file includes metadata.

Identify three pieces of metadata that is often stored with an image **[3]**

Answer

1. Location
2. Author
3. Resolution

Representing Sound

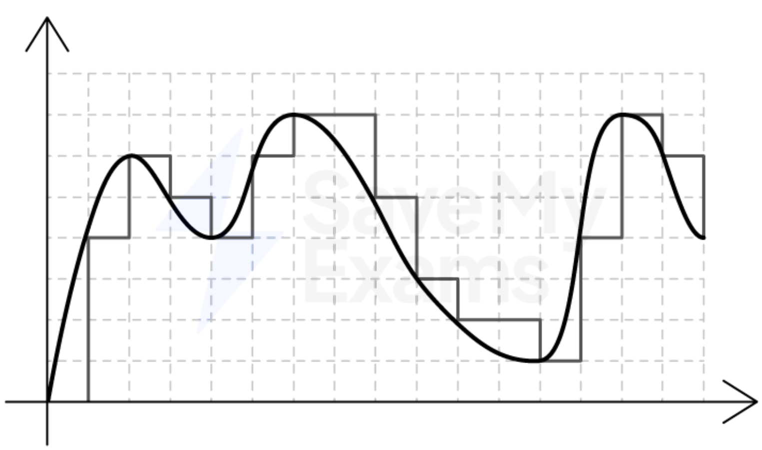
Your notes

How Sound is Sampled & Stored

- Computers represent all data in binary, including sound that we record using a microphone (input) or sound that we playback from a speaker (output)
- For this to happen, sound must be sampled and stored

How is sound sampled & stored?

- Measurements of the original sound wave are **captured** and **stored as binary** on secondary storage
- Sound waves begin as **analogue** and for a computer system to understand them they must be converted into a **digital** form
- This process is called **Analogue to Digital conversion** (A2D)
- It may be useful to understand the science of '*Sound Waves in the Ear*' to help with this concept
- The process begins by **measuring the height** (amplitude) of the **analogue sound wave**, these are called **samples**
- 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



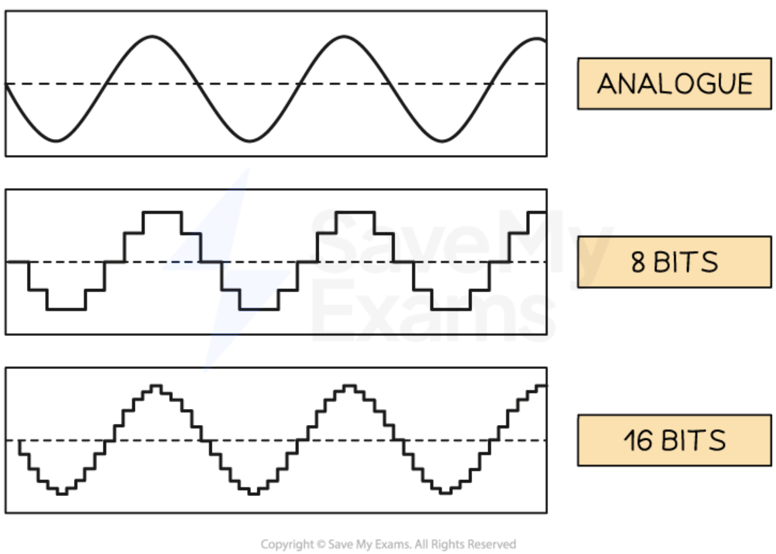
Your notes

- In this example, the grey line represents the digital wave that has been created by taking samples of the original analogue wave
- In order for the digital wave to look more like the analogue wave the sample rate and bit depth can be changed

Sample Rate, Duration & Bit Depth

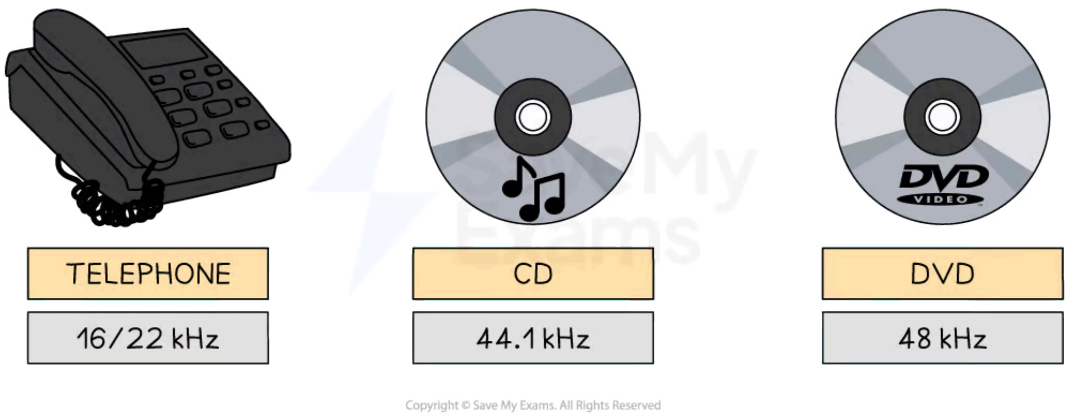
What is sample rate?

- Sample rate is the amount of **samples taken per second** of the analogue wave
- Samples are taken each second for the **duration** of the sound
- The sample rate is measured in **Hertz** (Hz)
- 1 Hertz is equal to 1 sample of the sound wave



- In the example above, the higher the sample rate, the closer to the original sound wave the digital version looks

Your notes



- The sampling rate of a typical audio CD is 44.1kHz (44,100 Hertz or 44,100 samples per second)
- Using the graphic above helps to answer the question, "*Why does telephone hold music sound so bad?*"

What is bit depth?

- Bit depth is the **number of bits stored per sample** of sound
- Bit depth is closely related to the colour depth of a bitmap image, they measure the same thing in different contexts

What effect do sample rate and bit depth have?

Factor	Effect on playback quality	Effect on file size
Sample rate	↑**higher** = more detail, better sound quality	↑**higher** = more data, larger file size
Bit depth	↑**higher** = bigger range, better sound quality	↑**higher** = more data per sample, larger file size

Worked Example

A student records a podcast about computer science for a school project.

Describe how an analogue sound wave is converted into digital form **[3]**

Answer

1 mark per bullet to max 3

- (analogue) sound wave is sampled
- ...amplitude/height (of wave) is measured
- ...at set/regular intervals
- Each sample/measurement is stored as a binary value

Your notes

Your notes

Compression

The Need For Compression

What is compression?

- Compression is **reducing the the size of a file** so that it **takes up less space on secondary storage**
- 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
- Compression can be achieved using two methods, **lossy** and **lossless**

Lossy Compression

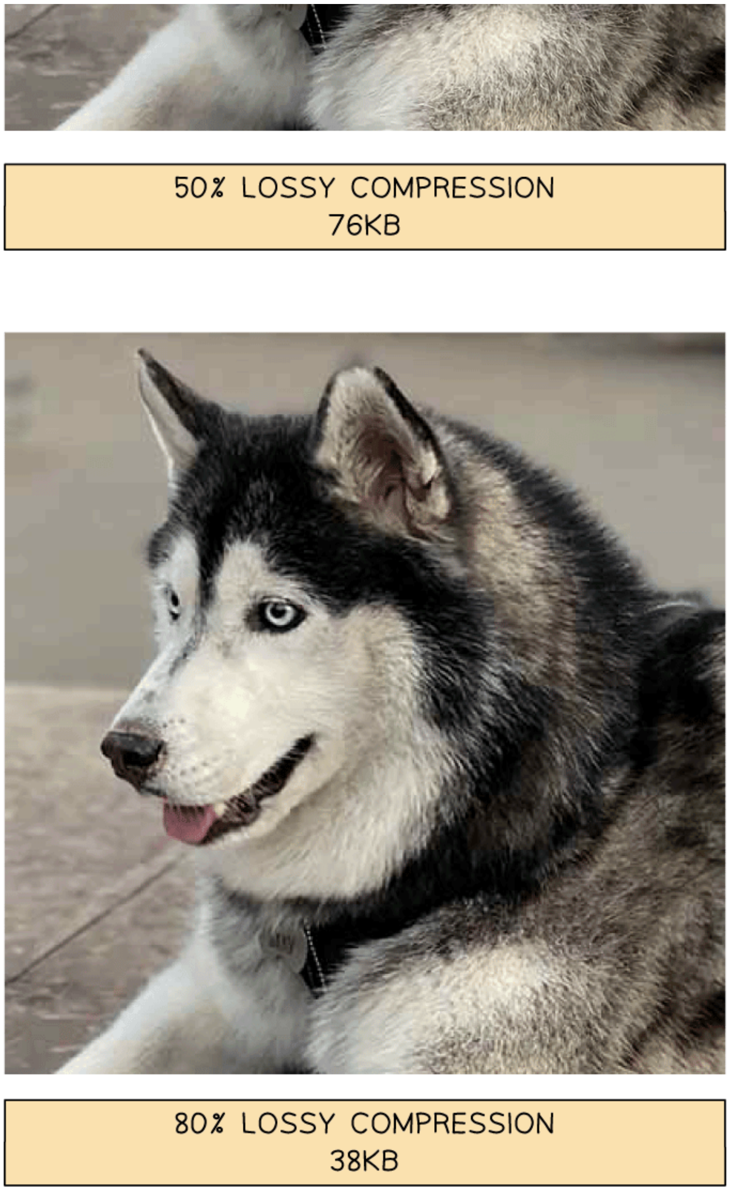
What is lossy compression?

- 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

Your notes

ORIGINAL JPG
824KB

Your notes



- In the images above, lossy compression is applied to a photograph and dramatically reduces the file size
- Data has been removed and the overall quality has been reduced, however it is acceptable as it is difficult to visually see a difference
- Lossy compressed photographs take up less storage space which means you can store more and they are quicker to share across a network

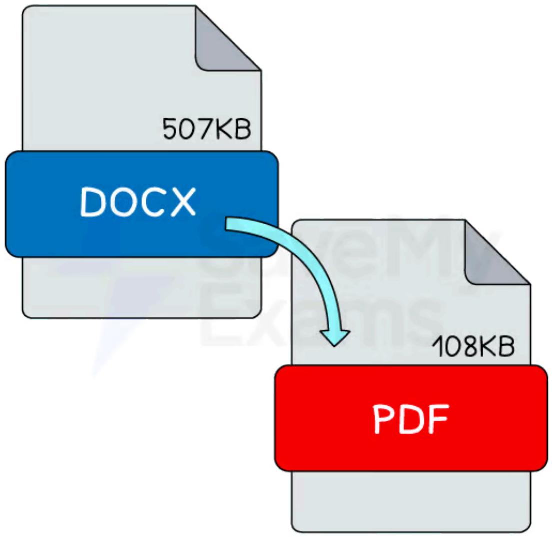
Lossless Compression

Your notes

What is lossless compression?

- Lossless compression is when data is **encoded** in order to **reduce the size on secondary storage**
- Lossless compression is **reversible**, the file **can be returned to its original state**
- Lossless can reduce the size of a file but not as dramatically as lossy
- 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**. For example, repeating characters are replaced with a single character and the number of occurrences in the document ("EEEEE" becomes "E5")

Your notes



- In the image above, lossless compression is automatically applied to document formats such as DOCX and PDF with a different rate of success
- When you open a lossless compressed document the decompression process reverses the algorithms and returns the data back to its original state
- Lossless compressed documents take up less storage space which means you can store more and they are quicker to share across a network

Worked Example

Sarah uses her computer to record an audio file of herself narrating a video for work.

She emails her recording to friend for proofing. She uses lossy compression to produce the sound file.

Explain two reasons why using **lossy** compression is beneficial. **[4]**

How to answer this question

- What are the differences between lossy and lossless?
- Can you state two differences? [2 marks]
- Can you say why each point is a benefit? [2 marks]

Answer

- Lossy will decrease the file size **[1]**
- ...so it can sent via email quicker **[1]**
- Lossy means data is lost **[1]**
- ...the difference is unlikely to be noticed by humans **[1]**

Your notes