Wednesday 9th September

Lesson Objectives 10y

- Welcome and Register
- Course Overview/Books used
- Syllabus Content
- Introduction to using Teams and Class Notebook
    - Collaboration Space
    - Content Library
    - Personal Space
        - Class Notes
        - Hand Outs
        - Homework
        - Quizzes
- Introduction to Chapter 1 notes (new book)

Content overview

Sections	Topics
Section 1 **Theory of computer science**	1.1 Data representation 1.1.1 Binary systems 1.1.2 Hexadecimal 1.1.3 Data storage 1.2 Communication and Internet technologies 1.2.1 Data transmission 1.2.2 Security aspects 1.2.3 Internet principles of operation 1.3 Hardware and software 1.3.1 Logic gates 1.3.2 Computer architecture and the fetch-execute cycle 1.3.3 Input devices 1.3.4 Output devices 1.3.5 Memory, storage devices and media 1.3.6 Operating systems 1.3.7 High- and low-level languages and their translators 1.4 Security 1.5 Ethics
Section 2 **Practical problem-solving and programming**	2.1 Algorithm design and problem-solving 2.1.1 Problem-solving and design 2.1.2 Pseudocode and flowcharts 2.2 Programming 2.2.1 Programming concepts 2.2.2 Data structures; arrays 2.3 Databases

Support for Cambridge IGCSE Computer Science

Our School Support Hub **www.cambridgeinternational.org/support** provides Cambridge schools with a secure site for downloading specimen and past question papers, mark schemes, grade thresholds and other curriculum resources specific to this syllabus. The School Support Hub community offers teachers the opportunity to connect with each other and to ask questions related to the syllabus.

Assessment overview

All candidates take two papers.

All candidates take:

Paper 1 1 hour 45 minutes
Theory 60%

75 marks

Short-answer and structured questions

Questions will be based on section 1 of the subject content

All questions are compulsory

No calculators are permitted

Externally assessed

and:

Paper 2 1 hour 45 minutes
Problem-solving and Programming 40%

50 marks

Short-answer and structured questions

Questions will be based on section 2 of the subject content

All questions are compulsory

20 marks are from questions set on the pre-release material[1]

No calculators are permitted

Externally assessed

1 The pre-release material for Paper 2 Problem-solving and Programming is made available to centres before the examination. It is also reproduced in the question paper. Candidates must not bring any prepared material into the examination.
Teachers should check the *Cambridge Handbook* for the year of assessment for information on when the pre-release materials will be available.

Chapter 1: Data Representation

Learning objectives

By the end of this chapter you will:

- understand how binary numbers are used by computer systems
- be able to convert to and from positive denary and binary numbers
- understand the use of hexadecimal notation
- be able to convert to and from positive denary and hexadecimal notation
- be able to convert to and from positive binary and hexadecimal notation
- understand how different types of data are stored as binary digits
- understand how data is compressed using lossy and lossless compression methods.

1.01 Binary number system

discrete vs analogue

SYLLABUS CHECK

Recognise the use of binary numbers in computer systems.

Data is information coded in a format ready for processing. Data is raw facts and figures and can be in the form of numbers, symbols or alphanumeric characters. As humans we use **analogue** data, such as sound or light waves and impulses on our skin. Everything we see and hear is a continuous transmission of analogue data to our senses. Analogue data is great for us as we can process and understand it. Computers cannot process analogue data, they are only capable of processing **digital** data. Any data that we want a computer to process must first be converted into digital data.

Information = Data ready for processing

Data = Raw facts figures

Computers cannot understand Analogue data

KEY TERM

Data – numbers, symbols or alphanumeric characters in their raw format before processing.

Analogue – this is the smooth stream of data that our senses process on a daily basis, such as a sound wave.

Digital – data represented in the values 1 and 0 that a computer can process.

Analogue ∩ Digital

TIP

Analogue data is data that is transmitted or received continually and it varies over a wide range. An example of analogue data is a sound wave. A sound wave is sent or received over a period of time and the frequency of the wave may differ greatly.

Digital data is data that consists of individually recognisable binary digits. A sound wave would be sampled at set time intervals and converted into a stream of binary digits.

Converting between denary and binary

SYLLABUS CHECK

Convert positive denary integers into binary and positive binary integers into denary.

KEY TERM

Denary – a system of numbers with a base of 10. Each unit used increases by the power of 10.

Binary – a system of numbers with a base of 2. Each unit used increases by the power of 2.

In our daily lives we use a **denary** number system. This system uses the digits 0–9 and it is called a base-10 number system. This means that the units it uses increase by the power of 10. If we take the denary number 123 we can calculate its value. Units begin at the right-hand side of the number and increase by powers of 10 as we move left.

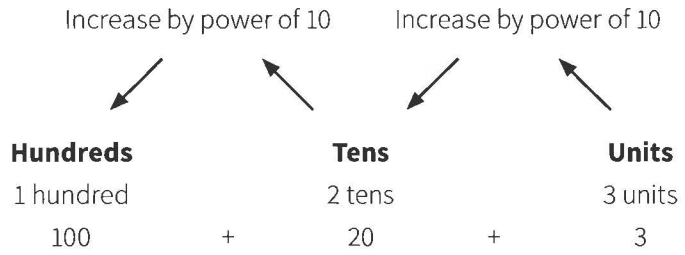

Figure 1.01 Denary number system

So we know the denary number 123 is one hundred and twenty-three.

Computers use a **binary** number system. This system uses the digits 0 and 1 and is called a base-2 number system. This means that the units it uses increase by the power of 2. The binary value represents the current flowing through a circuit: 1 means current is flowing, 0 means it is not.

If we take the binary number 1010 we can calculate its value as a denary number. Units begin at the right hand side and increase by the power of 2 as we move left. 1 indicates we use that unit, 0 indicates we do not.

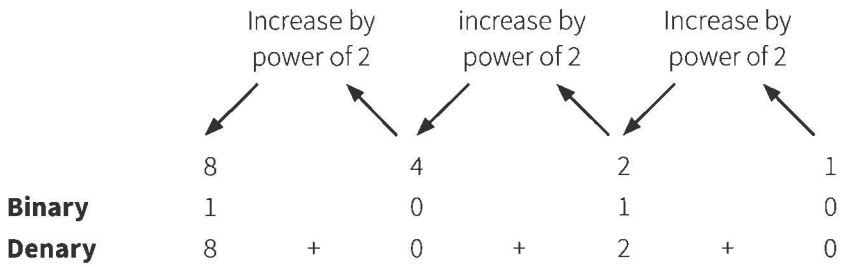

Figure 1.02 Converting a binary number into a denary number

So we know that the binary number 1010 as a denary number is 10.

When we are converting denary numbers to 8-bit binary, the units we use are 1, 2, 4, 8, 16, 32, 64 and 128. We indicate with a 1 or a 0 whether that number is required to create the denary number. For example, if we convert 150 denary to a binary number we could use the following steps:

1 Note down the binary units we can use from 128 down to 1, as in Table 1.01.

Table 1.01

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

2 Compare the denary number (150) to the first binary unit (128) and look to see if it is equal to or larger than it. 150 is larger than 128 so we do need 128 and a 1 can go beneath it (see Table 1.02). We can then deduct the 128 from 150 and we are left with 22.

Table 1.02

128	64	32	16	8	4	2	1
1							

3 Compare 22 to the next unit to the right (64) and look to see if it is equal to or larger than it. 22 is not larger than 64 so we do not need 64 and a 0 can go beneath it (see Table 1.03).

Table 1.03

128	64	32	16	8	4	2	1
1	0						

4 Compare 22 to the next unit to the right (32) and look to see if it is equal to or larger than it. 22 is not larger than 32 so we do not need 64 and a 0 can go beneath it (see Table 1.04).

Table 1.04

128	64	32	16	8	4	2	1
1	0	0					

5 Continue this process till you get to the end of the units and deduct the value of any unit that you use.

Table 1.05

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

So the denary number 150 is 10010110 as an 8-bit binary number (see Table 1.05).

If we calculate the denary number by adding together all the binary units that are used then the largest number we can make with eight binary bits is 255:

$$128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255$$

TIP

A quick check that you can do on your calculation is to look at the last unit on the right of your binary number. If the denary number you are converting is odd, this number should be 1. If the denary number you are converting is even, this number should be 0.

TEST YOURSELF

Convert the following denary numbers into binary:

55 248 13 191 250

To convert 8-bit binary numbers back into denary we need to add together all of the binary units that are marked as required by the digit 1. If we take the binary number 10010100 we can draw a table to see what units are needed:

Table 1.06

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

↑Indicates an even number

simple rule to apply

This means that the denary conversion of this binary number is:

$$128 + 16 + 4 = 148$$

You can use the quick check again to see if your denary number is correct, by using the rule that if the number to the far right in the binary number is 1, the denary number should be odd, otherwise it should be even.

TEST YOURSELF

Convert the following binary numbers into denary:

11001100	00011000	11110001
10100111	01010101	

32 16 2 1
11001100
255 − 51
204

Beyond 8-bit binary

We have looked at examples that convert 8-bit binary numbers into denary. We can easily apply the same principles to convert beyond eight bits, by simply using further binary units beyond 128. For example, the 12-bit binary number 101100101101 would be converted as shown in Table 1.07:

If your number is bigger than 8 bits

Table 1.07

2^9	2^8	2^7 (8)	2^6 (7)	2^5 (6)	2^4 (5)	2^3 (4)	2^2 (3)	2^1 (2)	2^0 (1)
2048	1024	512	256	128	64	32	16	8	4	2	1
1	0	1	1	0	0	1	0	1	1	0	1

$$2048 + 512 + 256 + 32 + 8 + 4 + 1 = 2861$$

You can go beyond 8 bits by just increasing your powers of 2

$2^8 = 256$

When we add our numbers we get 255 what is missing

The 16 bit binary number 1001100110011001 would be converted as shown in Table 1.08.

Table 1.08

32768	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1

$$32768 + 4096 + 2048 + 256 + 128 + 16 + 8 + 1 = 39321$$

Measuring memory size

SYLLABUS CHECK

Show understanding of the concept of a byte and how the byte is used to measure memory size.

Most computer systems have storage which is measured in bytes. A byte is a unit of data that is eight binary digits long. A byte can be used to represent a character, for example a letter, a number or a symbol. It can also hold a string of bits that can be used to build an image, for example.

byte is the unit of data used for storage
byte = B = 8 bits
bit = b = 1/8 bit

Computer storage is usually measured in multiples of bytes. A byte is abbreviated with the capital letter 'B'. A bit is one eighth of a byte and is abbreviated with a lower-case 'b'. The measurements for storage are shown in Table 1.08.

Table 1.09

Unit of measurement	Abbreviation	Conversion
Byte	B	8 bits
Kilobyte	kB	1024 bytes
Megabyte	MB	1024 kB
Gigabyte	GB	1024 MB
Terabyte	TB	1024 GB

Exact representation

The conversion units are commonly rounded down to reflect the standard definition of the units of measurement. For example kilo means one thousand and mega means one million. Therefore they can be represented as shown in Table 1.09.

Table 1.10

Unit of measurement	Abbreviation	Conversion
Byte	B	8 bits
Kilobyte	kB	1000 bytes
Megabyte	MB	1000 kB
Gigabyte	GB	1000 MB
Terabyte	TB	1000 GB

Easy to remember
Approximated representation

Do you know the storage capacity of your computer or phone

My Computer 512 GB
My phone 512GB

Storage capacity is continuously growing and is now expanding beyond the terabyte into 1000 (1024) TB, known as the petabyte. However, the terabyte is ample for what most people require a computer for on a daily basis.

can the storage capacity of a computer match that of a human brain

Using binary in computer registers

SYLLABUS CHECK

Use binary in computer registers for a given application (such as in robotics, digital instruments and counting systems).

KEY TERM

Register – small piece of memory where values can be held.

A **register** is a small piece of memory built into the central processing unit (CPU) of a computer system where values and instructions are temporarily held. They are not part of primary memory or secondary storage. Although they are small in capacity, registers have an extremely fast read and write rate, meaning data can be written to and read from a register much quicker than from primary memory or secondary storage. Therefore computer systems use registers to hold values and instructions for processing, to increase the speed at which they can be processed. If these values and instructions were processed straight from primary memory, processing would be much slower.

CPU = Brain of the computer

CPU holds data in registers

There are different types of register, such as processor registers and hardware registers. Processor registers, for example the program counter (PC), the accumulator and the memory

address register (MAR), are used to process data. These registers are part of the CPU and can be written to and read from extremely quickly. The fast speed of access makes registers very suitable for situations where small amounts of data need to be accessed quickly, such as performing calculations.

Hardware registers are specific to different types of hardware and are used to convey a signal. Consider a robot arm that has various motors to perform different operations, for example, raise the arm, open the grip and close the grip. Each motor works via a signal, 1 for on, 0 for off. A register is used for each motor to convey the signal.

The register within the CPU controlling this robot arm has a bit value of 1 or 0 to move the arm.

Figure 1.03 A robot arm uses hardware registers to control each motor. The register conveys the signal that turns the motor on or off

In a digital instrument, a register might be used to convey whether the device is sending a signal to the computer, 1 or 0. The instrument will have various registers to convey signals either way.

1.02 Hexadecimal number system

SYLLABUS CHECK

Identify current uses of hexadecimal numbers in computing, such as defining colours in Hypertext Markup Language (HTML), Media Access Control (MAC) addresses, assembly languages and machine code, debugging.

Hexadecimal is another number system that is used. Computers do not actually process hexadecimal, they convert it into binary before processing it. Programmers work with hexadecimal as it is easier for humans to read than binary. This is because it is a much shorter way of representing a byte of data, as reading and understanding lots of binary 1s and 0s can be difficult. In the same way, programs that are written in hexadecimal are easier to **debug** than those written in binary. Computers convert hexadecimal data into binary before processing it.

Hexadecimal numbers are used by humans to help us read binary digits

KEY TERM

Hexadecimal – a system of numbers with a base of 16. Each unit used increases by the power of 16.

Debug – finding and fixing problems and errors in a program.

Represents 65k different colors

#000000 number represents black

#FFAA33 represents orange

Hexadecimal (hex) is used as a notation for colour in HTML. Hex colour notations are normally six digits and each hex notation represents a different colour, for example #FFAA33 is orange and #000000 is black. In the hex code #FFAA33 the first two digits are the red component, the second two the green component and the last two the blue component. All three together create the colour orange.

FFFFFF	000000	333333	666666	999999	CCCCCC	CCCC99	9999CC	666699
660000	663300	996633	003300	003333	003399	000066	330066	660066
990000	993300	CC9900	006600	336666	0033FF	000099	660099	990066
CC0000	CC3300	FFCC00	009900	006666	0066FF	0000CC	663399	CC0099
FF0000	FF3300	FFFF00	00CC00	009999	0099FF	0000FF	9900CC	FF0099
CC3333	FF6600	FFFF33	00FF00	00CCCC	00CCFF	3366FF	9933FF	FF00FF
FF6666	FF6633	FFFF66	66FF66	66CCCC	00FFFF	3399FF	9966FF	FF66FF
FF9999	FF9966	FFFF99	99FF99	66FFCC	99FFFF	66CCFF	9999FF	FF99FF
FFCCCC	FFCC99	FFFFCC	CCFFCC	99FFCC	CCFFFF	99CCFF	CCCFF	FFCCFF

Figure 1.04 Hex colour codes

Standard Windows error message codes are given in hexadecimal notation, for example error code 404 (meaning 'File not found') is a hexadecimal notation.

HTTP Error 404

404 Not Found

The Web server cannot find the file or script you asked for.
Please check the URL to ensure that the path is correct.

Please contact the server's administrator if this problem persists.

Have you seen this before?
#404
Hexadecimal notation

Figure 1.05 404 error code

Media Access Control (MAC) addresses are 12-digit hexadecimal numbers that uniquely identify each different device in a network. An example of a MAC address would be 00-1B-63-84-45-E6. You can learn more about MAC addresses in Chapter 2.

MAC Address?
Machine code?
All represented in hexadecimal

Machine code consists of simple instructions that are directly executed by the CPU. Hexadecimal is used for machine code as each byte can be coded as two hexadecimal symbols. You can learn more about machine code in Chapter 3.

SYLLABUS CHECK

Represent integers as hexadecimal numbers.

Hexadecimal is a base-16 number system. This means that the units it uses increase by the power of 16.

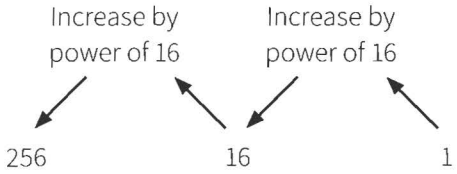

Figure 1.06 Hexadecimal base-16 number system

Hexadecimal uses 16 symbols, the numbers 0–9 and the letters A–F. The notation for each denary number between 0 and 15 is shown in Table 1.10.

Table 1.11

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

The reason for using the symbols A–F for the decimal numbers 10–15 is because only one symbol can be used per unit required in hexadecimal. For example, the denary number 10 has two symbols and hexadecimal only uses one symbol for each unit, so we use the letter 'A'. Similarly 14 has two symbols, so we use the letter 'E' instead. This means that we can use a single symbol for each unit. The same system applies in base ten: each unit is represented by one digit.

Converting between denary and hexadecimal

SYLLABUS CHECK

Convert positive hexadecimal integers to and from denary.

To convert the denary number 55 into hexadecimal we need to calculate how many of the unit 16 and how many of the unit 1 are required. As hexadecimal is a base-16 system it increases in unit by the power of 16. We can use up to fifteen 1s before we use a 16. We can then use up to a further fifteen 1s before we use the next 16.

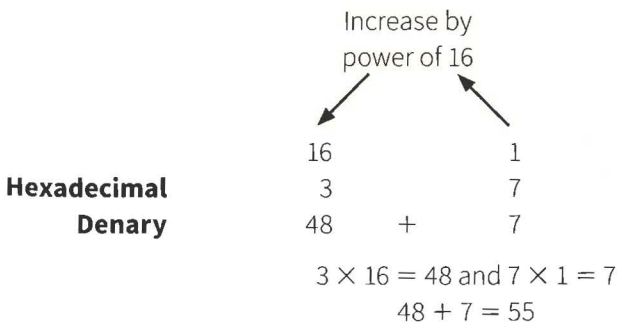

Figure 1.07 Converting a denary number into hexadecimal

The denary number 55 in hexadecimal notation is 37. This means we use three 16s and seven 1s to create the denary number 55. If the denary number is larger than fifteen 16s (the largest amount of 16s that can be used) then we use the next unit in the base-16 system. If we were to convert the denary number 1080 into hexadecimal we do what is shown in Figure 1.08.

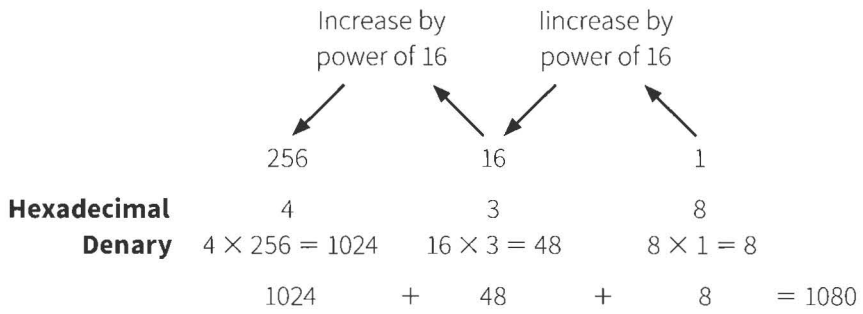

Figure 1.08 Hexadecimal notation for denary number 1080

Therefore the hexadecimal notation for the denary number 1080 is 438.

TEST YOURSELF

Convert the following denary numbers into hexadecimal:	Convert the following hexadecimal notations into denary numbers:
101	15
1551	1AB
65	E9
168	2F2
20	23

Converting between hexadecimal and binary

SYLLABUS CHECK

Convert positive hexadecimal integers to and from binary.

To convert binary into hexadecimal we take each binary number and separate it into 4-digit nibbles. If we take the binary number 1011 1100 1001 we would split it into three 4-digit nibbles, 1011, 1100 and 1001. We can then convert each nibble into a hexadecimal symbol. To do this we use the first four binary places 1, 2, 4 and 8 to calculate each hexadecimal symbol as shown in Figure 1.09.

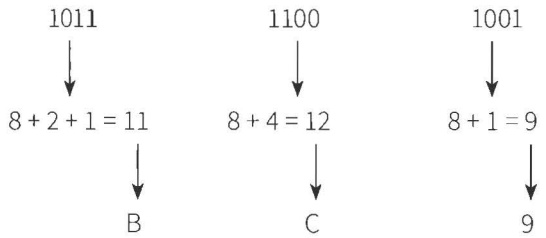

Figure 1.09 Converting a binary number into hexadecimal

Therefore 1011 1100 1001 in hexadecimal notation is BC9.

To convert hexadecimal notation back into binary we reverse the process. We take the hexadecimal notation BC9 and convert the denary value of each hexadecimal unit into a 4-bit binary number. We then join the binary numbers together:

Hexadecimal:	B	C	9
Denary:	11	12	9
Binary conversion:	1011	1100	1001

We can convert a 16 digit binary number in the same way. If we take the binary number 1001001110001010 we would split it into four 4-digit nibbles 1001, 0011, 1000 and 1010.

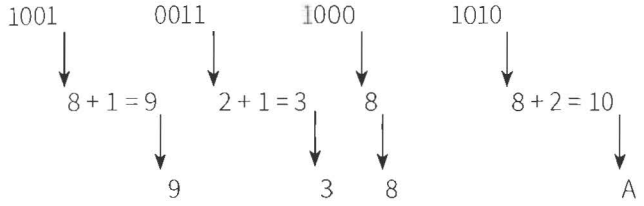

Figure 1.10 Converting a 16 digit binary number into hexadecimal

Therefore 1001001110001010 in hexadecimal notation is 938A.

TEST YOURSELF

Convert the following binary numbers into hexadecimal notations:

1010 1100 1100 1010

1101 1011 0111 0001 1111 0011

1001 1001 1000

1.03 Data storage

As computers can only process the 1s and 0s that we know to be binary, all data stored on a computer is in binary form. If we had to carry out our daily tasks on a computer using only binary it would be extremely time consuming and challenging.

Imagine having to write an email to your friends using only binary. Typing the 1s and 0s that represent each **character** would take you much longer than typing out the characters the binary numbers represent. Also, imagine having to type in each binary digit needed to create your favourite images – it would be extremely difficult. Depending on its resolution, a character might require 100 bits of data. An image might require millions of bits.

A number of systems and software were developed to do this for users and they help a computer store different data such as text, images, video and audio.

KEY TERM

Character – text, numbers and symbols, for example each key on a keyboard.

Text, numbers and symbols

There are two systems commonly used for character (text, numbers and symbols) representation, namely ASCII and Unicode. ASCII uses only 8 bits, giving a possible 256 characters. It is suitable for Standard English but does not contain a large enough character set for some other languages. This is when Unicode is used as it contains many more characters. Unicode uses 16 bits, giving 65 536 possible characters. In ASCII and Unicode each character is represented as a binary number. For example, the letter A is 01000001.

In ASCII each character will take 1 byte of storage space as it is made up of 8 bits. In Unicode a character takes up 2 bytes as it is made up of 16 bits. Capital letters and small case letters have individual binary codes, as do numbers and punctuation.

The word 'Computer' would be as shown in Figure 1.11.

C – 01000011	u – 01110101
o – 01101111	t – 01110100
m – 01001101	e – 01100101
p – 01110000	r – 01110010

Computer =

01000011 01101111 01001101 01110000
01110101 01110100 01100101 01110010

Figure 1.11 The word 'Computer' in ASCII

TEST YOURSELF

Search for an ASCII table on the internet and try and decode this message:

0100001101101111010011010111000001110101011101000110010101110010010100110
1000110110100101000101011011100110001101100101011010010111001101100110011
0101011011000100001

Using the same ASCII table write a message for your friend to decode.

Pictures

We regularly use our computers to store and view images. An image as we see it is analogue data, but a computer will only understand it if it is digital data. We need to convert analogue data into digital data for a computer to process it. This can be done using various devices, such as a scanner or a digital camera.

Images are made up of pixels. A pixel is a tiny dot on the screen. If an image was simply black and white, each pixel would either be black or white. A 1 would represent a black pixel and a 0 would represent a white pixel. Using this we can look simply at how an image is created.

If we are given a series of binary data we can use this in a grid. We need to know the dimensions of the grid to be able to create the image. This data is called metadata. Metadata is what tells the computer how many pixels wide and how many pixels high an image should be. This is the 'resolution' of an image. The grid in Figure 1.12 is 11 × 12.

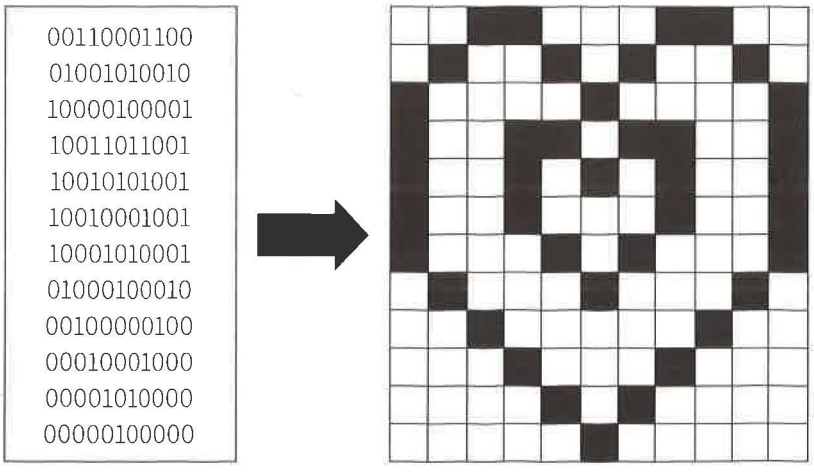

Figure 1.12 An image created using binary data

We have taken the binary data and created an image from it, a lovely heart! If we want an image to have more than just two colours, a computer needs further binary data to represent the colour of each pixel. Most colours are made up of red, green and blue in the RGB colour system. In this system the shade of each colour R, G and B will be represented in a byte for each. This is why images can become large files, especially when they are high in quality. To get the correct colour depth in each pixel to create a detailed image, the amount of binary data needed is much greater.

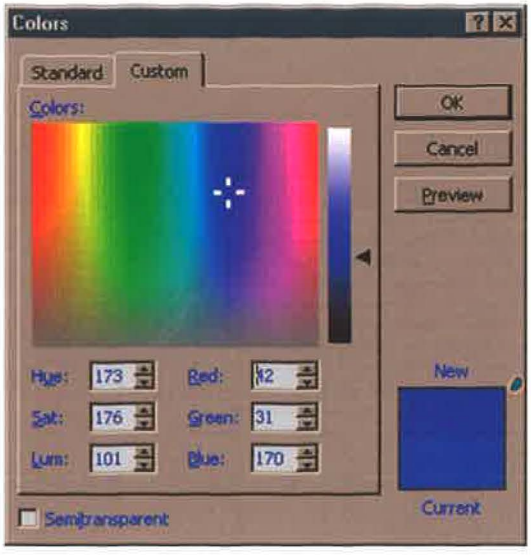

Figure 1.13 RGB colour scale

Sound

The sound we hear is also analogue but, as we know, computers work digitally and can only process binary. We need to convert the analogue sound into binary for a computer to be able to process it. Sound is recorded using a microphone and converted to binary by software.

Sound is recorded at set timed intervals; this process is known as sampling. The samples are then converted into binary. If the set timed intervals are closer together, the sound track will be higher in quality. Simply with more samples the sound can be more accurately captured.

Sample sizes are measured in hertz. 1 hertz equals 1 sample per second. A telephone communication samples a voice at 8000Hz but a higher quality recording, such as a CD, samples music at 44 100Hz.

Take the simple sound wave in Figure 1.14.

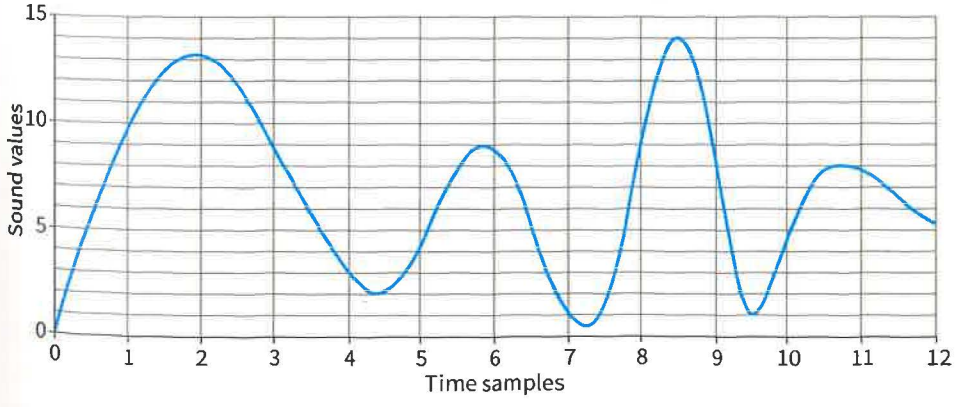

Figure 1.14 Sound wave

If we record the sound value at each time sample we would have the values in Table 1.11.

Table 1.12

Time sample	1	2	3	4	5	6	7	8	9	10	11	12
Sound value	9	13	9	3.5	4	9	1.5	9	8	5	8	5.5

If we then play back the recording the system will use the value at each sample interval to do this. The resultant wave is shown in Figure 1.15.

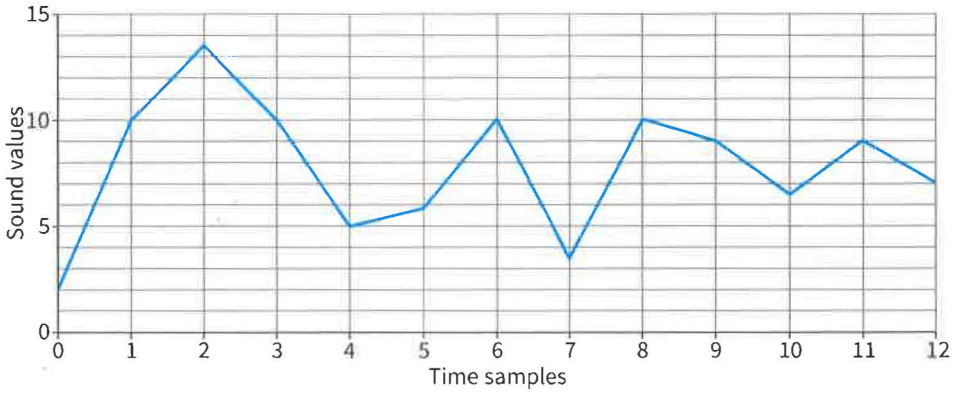

Figure 1.15 Sound wave created by playing back a recorded sound

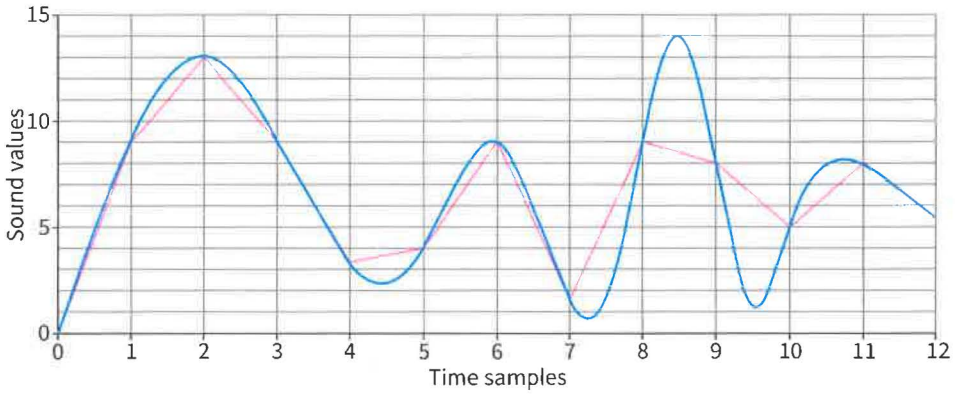

Figure 1.16 Quality of recorded sound

This is what affects the quality of recorded sound. The closer together the time samples that are taken, the higher the quality of the recorded sound. This is because when the sound is played back it has more data to help create a more accurate sound. If the time samples were closer together in Table 1.11, when the recorded sound is played back it may not miss so many of the peaks and troughs it has with the current time samples.

Data compression

Data compression is when the bit structure of a file is manipulated in such a way that the data in a file will become smaller in size. This means that less storage space will be needed to store the file and the file will be easier to transmit from one device to another.

Data compression is done by using compression **algorithms** that manipulate the data. These algorithms normally manipulate the data so that repeating data is removed, either

on a temporary or a permanent basis, depending on the method used. You can learn more about algorithms in Chapter 13.

KEY TERM

Algorithm – a step-by-step set of instructions.

There are two main methods for compressing data: lossy and lossless.

SYLLABUS CHECK

Show understanding of the principles of data compression (lossless and lossy compression algorithms) applied to music/video, photos and text files.

Lossy compression

Lossy is derived from the word 'loss' and this refers to the way this method of compression works. With lossy compression, data that is deemed redundant or unnecessary is removed in the compression process. The data is removed permanently, so it is effectively 'lost'. This way the size of the file is reduced.

Lossy compression is mostly used for multimedia such as audio, video and image files. This is mostly done when streaming these files, as a file can be streamed much more effectively if it is smaller in size.

If a lossy compression method is used on a music file it will try to remove all background noise and noises that may not be heard by the human ear. This data isn't hugely necessary for playing the track; removing it will mean the track will not sound exactly as it did when recorded, but it will be a very close representation.

Lossless compression

Lossless refers to a method of compression that loses no data in the process. In lossless compression, the compressed data can be reversed to reconstruct the data file exactly as it was. Lossless compression is used when it is essential that no data is lost or discarded during the compression process. There are many different lossless compression algorithms; most work using a shorthand to store the data that can be then reconstructed when the file is opened.

If a lossless compression method is used on a music file it will not lose any of the data from the file. A possible way to compress the data would be to look for repeating patterns in the music. It would store this pattern once along with how many times it is repeated. This way repeating data is reduced. When the music track is played, the full track, exactly as it was recorded, can be reconstructed and listened to. People may use lossless compression when downloading a music track if they want the highest quality possible and to hear the track exactly as it was recorded. Lossless compression can also be used when storing text files.

Consider the following message:

WHEN IT IS SNOWING HEAVILY LOOK OUTSIDE.
LOOK OUTSIDE IT IS SNOWING HEAVILY.

Excluding the spaces between the words and the full stops, the message has a total of 62 characters. 1 character requires 1 byte of storage, so we would need 62 bytes of memory to store this message.

When we examine the message we can see that it consists of words that are mostly repeated. Instead of storing all 62 characters individually, we could instead store the words and the positions at which they occur in the message, in a lookup table:

Table 1.13

Word	Position(s) in the message
WHEN	1
IT	2 10
IS	3 11
SNOWING	4 12
HEAVILY	5 13
LOOK	6 8
OUTSIDE	7 9

When we store the message this time we need 1 byte for each character in each word and 1 byte for each position the word occurs in the message:

Table 1.14

Word	Position(s) in the message
WHEN	1
IT	2 10
IS	3 11
SNOWING	4 12
HEAVILY	5 13
LOOK	6 8
OUTSIDE	7 9
TOTAL BYTES: 33	TOTAL BYTES: 13

Therefore we would need 33 bytes to store the words and 13 bytes to store the positions, giving a total of 46 bytes. This is much less than the 62 bytes we required with our original method. No data has been lost and we have reduced our storage requirements by 26%, quite a saving! To recreate the message, the computer simply retrieves the words and places them in the positions allocated. We should note that the amount of compression we can achieve varies depending on the data we wish to store.

Consider a second message:

ASK NOT WHAT YOUR FRIEND CAN DO FOR YOU.
ASK WHAT YOU CAN DO FOR YOUR FRIEND.

Excluding spaces and full stops, this message has 59 characters, therefore requiring 59 bytes for storage. This message is shorter than that in our first example, which required 62 bytes. If we were to apply our lossless compression algorithm, we would end up with:

Table 1.15

Word	Position(s) in the message
ASK	1 10
NOT	2
WHAT	3 11
YOUR	4 16
FRIEND	5 17
CAN	6 13
DO	7 14
FOR	8 15
YOU	9 12
TOTAL BYTES: 31	TOTAL BYTES: 17

We need 31 bytes to store the words and 17 bytes to store the positions, giving a total of 48 bytes. This again is less than the 59 bytes required with our original method. This time we have reduced our storage requirements by a lesser amount though of 19%. This message, even though it was originally shorter than our first message, would after compression require more storage space than the first message.

Uncompressed image files can potentially be huge. This can make adding downloading, uploading or emailing them very difficult and time consuming. Many emails limit the size of the file that can be attached. This means that most uncompressed images, especially high quality ones, cannot be attached to an email without being compressed.

Both the lossy and lossless methods of compression reduce the size of an image by looking for repeating colour patterns within the image. For example, for an image that has a main background colour that is the same throughout the image, a compression method will recognise that there are a lot of pixels that all have the same value and collate them. This means they will be stored as a single data value with further data that records the pattern.

Lossy compression will reduce the file size further by removing detail from the image that should go unnoticed and will not affect the quality too much. One issue with using some lossy compression methods on images, is that the method will remove a little bit of detail each time the image is saved in the compression method e.g. JPEG. This means that there will be a small loss in quality each time it is saved.

1.04 File formats

SYLLABUS CHECK

Show understanding that sound (music), pictures, video, text and numbers are stored in different formats.

A file format is the method that we choose to store different data on a computer. Different file formats encode data in different ways. This means that they organise the data for storage in different ways. It is important for software to recognise the file format used to save the data in order to access it.

There are many different types of file format. Some are specific to software and some are more generic or standard. Certain file formats are designed for a particular type of data, for example text, images or multimedia. The file format for a file will mostly depend on the type of data it will be storing. The file format is normally three or four characters, separated from the file name with a dot, and is known as the file extension. These are the most common file extensions:

Table 1.16

File type	File extension	Use
Text	.doc	Microsoft Word document
	.rtf	Rich Text Format file
	.pdf	Portable Document Format
Data	.csv	Comma Separated Values file
	.xls	Microsoft Excel Spreadsheet
	.mbd	Microsoft Access Database
Audio	.mp3	MP3 audio file
	.mid	MIDI file
	.wav	Wave audio file
Video	.mp4	MPEG-4 video file
	.flv	Flash video file
	.wmv	Windows Media Video file
Image	.bmp	Bitmap file
	.gif	Graphical Interchange Format file
	.jpg	JPEG Photo
	.png	Portable Network Graphic

SYLLABUS CHECK

Show understanding of the concept of Musical Instrument Digital Interface (MIDI) files, JPEG files, MP3 and MP4 files.

Users often need to import and export data in and out of different software. In order to do this users need the different files to be compatible with each other so that data can be effectively imported and exported. This prompted the development of standard file formats that different software applications can understand.

There are four multimedia standard file formats that you should be aware of:

- Musical Instrument Digital Interface (MIDI) uses a series of protocols and interfaces that allow lots of different types of musical instrument to connect and communicate. MIDI also allows one computer, or instrument, to control other instruments. The controlling device instructs the others on which notes to play and when, whilst specifying the pitch, duration and velocity of each note. MIDI files are, therefore, not a musical recording, but a series of instructions for an instrument to carry out.
- Joint Photographic Experts Group (JPEG) is a standard format for lossy compression of images. It can reduce files down to 5% of their original size.
- MP3 is a standard format for lossy compression of audio files.
- MP4 is a standard format for lossy compression of video files. It can also be used on other data such as audio and images.

MP3 and MP4 have developed from the original file format Motion Picture Experts Group (MPEG). This is a lossy compression method for video files dating back to 1991.

JPEGs, MP3s and MP4s are used in a wide variety of devices, such as computers, digital cameras, DVD/Blu-Ray players and smartphones to store content.

Summary

- As humans we process analogue data but computers process digital data.
- Computers use a binary number system. This consists of a series of 1s and 0s to represent data.
- Computer storage systems are measured in multiples of bytes.
- A register is a small piece of memory where values can be held. Computers use them to hold values for processing.
- Computers do not process hexadecimal notation, they convert it into binary first. Programmers work with hexadecimal notation as it is shorter and easier to read than binary.
- Computers convert text, images and sound into binary to process and store.
- Data compression reduces the size of a file. The two main methods of data compression are lossy and lossless.
- File formats are the method chosen to store data on a computer.

Exam-style Questions

1 A stopwatch uses six digits to display hours, minutes and seconds: (3 marks)

0 1 : 5 4 : 2 3

It uses a nibble of binary data for each digit displayed on the stopwatch. What time is the stopwatch stopped at when the binary nibbles show the following values?

0000 0010 : 0011 1000 : 0101 1001

2 What is meant by the term 'byte'? (2 marks)

3 The following instruction, written in machine code, is stored in computer memory: (3 marks)

110000011110

Convert the code into hexadecimal notation.

4 Explain why hexadecimal notation is sometimes used to represent binary numbers. (2 marks)

5 Tom wants to send an image by email to his friend Nadia. He needs to reduce the size of the image in order to be able to send it via email. Describe two methods he could use to do this. (4 marks)

6 Tick (✓) the most suitable file extension for the following tasks:

	.jpg	.mp4	.mp3	.csv
Storing holiday photos				
Storing holiday videos				
Storing a favourite song				
Storing data to be imported into other files				

(4 marks)

7 Explain how ASCII is used to represent text so it can be processed by a computer. (2 marks)

8 Kamil and his band want to record the new track they have written. How could the quality of their sound recording on a computer be improved? (2 marks)