1. Data Representation
Number Systems

Using binary to represent data in a computer system
<u>Why</u> computers use binary to represent all forms of data
- All data needs to be converted to binary to be processed by a computer.
- Data is processed in a computer by using logic gates/ switches/ transistors.
- These can only store/process data in two states: high-low / on-off / 1 and 0.
- Data is stored in a computer using registers

<u>How</u> computers use binary to represent all forms of data.
- All data needs to be converted to binary to be processed by a computer.
- For conversion, a binary value is assigned to each character/pixel in the data.
- Data is processed using logic gates and stored in registers.

<u>Bit (binary digit):</u> The smallest unit of data in a computer; can use (0 or 1).
<u>Logic gates:</u> electronic devices that perform logical operations on binary data.
- Processes binary data: applies Boolean logic to input values to produce binary output.

Explain what is meant by binary number system
- It has a base of 2
- It only uses two values
- ... that are 1 and 0

Binary	**Denary**	**Hexadecimal**
Base 2 system	Base 10 system	Base 16 system
Values: 0,1	Values: 0-9	Values: 0-9, A-F
Units / placeholders / column headings increase by power of 2. 2^0, 2^1, 2^2, 2^3...	Units / placeholders / column headings increase by power of 10. 10^0, 10^1, 10^2, 10^3...	Units / placeholders / column headings increase by power of 16. 16^0, 16^1, 16^2, 16^3...
more digits for same value.	less digits for same value.	

NOTE
1 nibble = 4 bits
1 byte = 8 bits

1. **Denary to Binary**

   <u>Example: 188</u>

   - Write down the powers of 2 from right to left, starting with 2^0, 2^1, 2^2 and so on (until 128 - as answers must be given in 8 bits)

128	64	32	16	8	4	2	1

   - Starting from the leftmost column, write 1 if the corresponding power of 2 is less than or equal to the number you're converting; otherwise write 0.

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

   - Check your work by adding together all the colum headings with 1. underneath; 128+32+16+8+4 = 188
   - Write the binary digits from left to right to get binary equivalent of 188: 10111100

2. **Binary to Denary**

   <u>Example: 10111100</u>

   - Write down the powers of 2 from right to left, starting with 2^0, 2^1, 2^2 and so on (until 128 - as answers must be given in 8 bits)

128	64	32	16	8	4	2	1

   - Starting from the leftmost column, write the binary digits in the columns.

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

   - Add up the values in each column where the binary digit is 1: 128+32+16+8+4 = 188
   - Thus 188 is the denary equivalent

3. **Binary to hexadecimal**

   <u>Example: 10111100</u>

   - Groups the binary digits into groups of 4, starting from the rightmost digit. (if there are not enough digits to make a group of 4, add leading zeroes as needed)

- Add column headings to work out the value of each nibble (1 nibble = 4 bits)

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

- 8+2+1 = 11 8+4 = 12
- Replace each nibble with its corresponding hexadecimal value:

Denary	Hex value
10	A
11	B
12	C
13	D
14	E
15	F

NOTE: numbers from 1-9 remain the same in hexadecimal.

- 11 = B; 12 = C
- Thus the corresponding hexadecimal value = BC

4. Hexadecimal to binary

Example: BC

- Separate each hex digit into groups of 4 bits (nibble)

8	4	2	1		8	4	2	1

- Convert each hex digit to appropriate form: B = 11; C = 12
- Fill up the binary digits under the appropriate columns

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

- Thus the binary equivalent = 10111100

5. **Denary to hexadecimal**
    - Turn your denary number to binary, and then turn the binary number to hexadecimal

6. **Hexadecimal to denary**
    - Turn your hexadecimal number to binary, and then turn the binary number to denary.

Binary addition

- 0+0=0
- 0+1=1
- 1+1=10 (The 1 is carried into the next column on the left)
- 1+1+1=11 (The 1 is carried into the next column on the left)

<u>Add binary numbers 11101110 and 00110001 using binary addition</u>

- One mark for each correct nibble
- One mark for correct carries
- One mark for identification of overflow error

```
     1 1
1    0001 1111
```

Overflow

<u>Explain why overflow error occurs</u>

- A computer/device has a predefined limit it can represent or store, eg. 16-bit.
- An overflow error occurs when a value outside this limit should be returned.
- Overflow error occurs when result of a calculation is greater than 255 (in 8-bit register)

- Smallest number of bits that can be used to store the denary value 2000: 11
- Least number of bits that can be used to store the denary number 301: 9
- Least number of bits that can be used to store the hexadecimal value A4D: 12

Binary shifts

- Term used for multiplying/ dividing in binary.
- Binary shift moves all bits in a binary number a certain number of positions to left/ right.
- Bits shifted from the end of the register are lost and zeros are shifted in at the opposite end of the register.
- Most significant bit(s) or least significant bit(s) are lost, according to the shift performed.
- The positive binary integer is multiplied or divided according to the shift performed: Left shift = multiplication by a power of 2; Right shift = division by a power of 2.

NOTE: If we lose 1 bit during logical shift, we have exceeded maximum number of (left / right) shifts possible using that register.

Effect that logical right shift of three places has on binary number

- The value becomes incorrect/inaccurate as the rightmost bits are lost
- (It is divided by 8) - only give this point if it's for 2 marks

Most significant bits = leftmost bits = lost during logical left binary shift
Least significant bits = rightmost bits = lost during logical right binary shift

Two's Complement

- A method of representing signed integers (positive & negative) in binary form
- The leftmost bit represents the sign (0 for positive and 1 for negative)

To convert a negative number to a two's complement 8-bit integer

- Represent the positive equivalent of the number in binary form
- Invert all the bits (flip all the 1's to 0's and all the 0's to 1's).
- Add 1 to the result

To convert a positive number to a two's complement 8-bit integer

- Represent the number in binary form with leading zeros until it is 8 bits long.
- If the number is positive, the leftmost bit should be 0

Complete the binary register for the denary number –78 (two's complement)

- Binary register for +78 = 01001110
- Flip the 1s & 0s = 10110001
- Add 1: 10110010

Benefits of using hexadecimal

Why programmer may use hexadecimal to represent binary numbers // Benefits of converting binary to hexadecimal

- Easier/quicker to read/write/understand
- Easier/quicker to debug
- Less chance of making an error
- Shorter representation // Takes up less screen/display space // Fewer digits to represent the same value in hexadecimal than in binary

Uses of hexadecimal in computers

- IP address (IPv6) & MAC address
- URL // web address

- Error messages/codes
- Colour codes // colour in HTML/CSS
- Assembly language // low-level language
- ASCII // Unicode
- Locations in memory // memory addresses
- Memory dumps

<u>Why error codes are represented in hexadecimal, instead of binary</u>
- It is easier for user to read/recognise/understand
- It takes up less space on a display

HTML values
Red: FF 00 00
Green: 00 FF 00
Blue: 00 00 FF
Cyan: 00 FF FF
Magenta: FF 00 FF
Yellow: FF FF 00

How are other shades formed?
- hex values between 0 to F are combined together to create a hex code
- different combinations in hex codes will create different shades/tones/colours

1.2 Text, Sound, Images

Representing text

Text is converted to binary to be processed by a computer.

Character set
- All the characters and symbols that can be represented by a computer system.
- Each character and symbol is assigned a unique value.

American Standard Code for Information Interchange (ASCII)
- Assigns a unique 7-bit binary code to each character
- Includes uppercase/ lowercase letters, digits, punctuation marks, control characters

<u>Disadvantages of using ASCII</u>
- Has limitations in terms of the number of characters it can represent
- Does not support characters from languages other than English

Unicode
- Allows for a greater range of characters & symbols than ASCII: including different languages and emojis
- Uses variable-length encoding scheme: assigns a unique code to each character, which can be represented in binary form using multiple bytes

Disadvantage of using unicode
- Requires more bits per character than ASCII // each character is encoded using more bits
- Text stored takes up more storage space
- Larger file size/ slower processing times

Differences between ASCII and Unicode
- ASCII has limited/fewer characters // Unicode has a more characters
- ASCII covers a limited set of languages/fewer languages
- Unicode includes many/more languages/emojis
- ASCII requires 7/8 bits per character
- Unicode requires up to 16/32 bits per character
- ASCII has 128/256 characters
- Unicode has 65 536/4 294 967 296 characters // approx. 60/70 thousand/4 billion characters

Representing sound
- Sound is a type of analog signal.
- Sound wave is sampled, for sound to be converted to binary & processed by a computer.
- **Sampling** involves taking measurements of the sound wave at regular intervals.
- These measurements are converted into binary data.

Factors affecting sound quality
- **Sample rate:** number of samples taken in a second.
- **Sample resolution:**
    - The number of bits that are used per sample
    - … that provides the variation in amplitude that can be stored for each sample // defines the number of different amplitudes that can be recorded
    - … that determines how quiet/loud the sounds are that can be recorded
    - Example e.g. 16-bit

Accuracy of recording and file size increases as the sample rate and resolution increase.

Advantages of using larger sample rate/ sample resolution
- Better quality & accuracy of recording

- Less sound distortion

Benefit of using larger sample rate

- The recording of the song is more accurate/closer to original

Disadvantages of using larger sample rate/ sample resolution

- Larger file size
- Fewer number of files can be stored (eg. on hard drive)
- Takes longer to download sound files (from the internet)
- Takes longer to transfer sound files (from device to device)
- Requires greater processing power

Drawback of using larger sample rate

- The file size will be increased
- – The file will require more storage space

How analogue sound is recorded and converted into digital

- (The analogue sound is) recorded using a microphone
- The sound wave is sampled
- … measuring the height/amplitude
- Each amplitude has a unique binary value
- The sample rate is set
- … that is the number of samples taken per second
- The sample resolution is set
- … that is the number of bits used for each sample
- Each sample taken is converted to binary

Why a musician would choose to use lossless compression instead of lossy

- They want to be able to edit the original sound file
- They want the highest sound quality for the file // They want the sound to be closest to the original recording
- … using lossy would reduce the sound quality
- … using lossy will permanently remove some of the data // no data will be permanently removed with lossless

Representing images

Image: series of pixels that are converted to binary, which is processed by a computer.

- Each pixel can be represented by a binary code, which is processed by a computer

Factors affecting image quality

- **Resolution:** dimensions of the image // number of pixels wide by number of pixels high
- **Colour depth:** number of bits used to represent each colour

File size and quality of the image increases as the resolution and colour depth increase.

Benefit of using high resolution image/ high colour depth

- Sharper and more detailed image
- More colours can be represented
- More realistic image

Benefit of increasing colour depth

- A greater range of colours can be seen/used
- Image will be closer to the actual content of the image/real life
- The image will have more detail

Drawback of using a high resolution image

- Larger file size
- Fewer number of images can be stored (eg. on a hard drive).
- Takes longer to download images (from the internet).
- Takes longer to transfer images (from device to device).
- Requires greater processing power

How digital image file is stored by computer

- Image is made of pixels
- Each pixel stores one colour
- Image has a set number of pixels wide by pixels high
- Each colour has a unique binary value // Each colour has a unique colour code
- The colour/binary value of each pixel is stored in sequence
- File contains metadata to identify how the file should be displayed
- … metadata can be the colour depth / resolution

Data storage

bit	smallest unit of data in a computer (0 or 1).
1 nibble	4 bits
1 byte	8 bits
1 kibibyte (KiB)	1024 bytes
1 mebibyte (MiB)	1024 kibibytes
1 gibibyte (GiB)	1024 mebibytes

1 tebibyte (TiB)	1024 gibibytes
1 pebibyte (PiB)	1024 tebibytes
1 exbibyte (EiB)	1024 pebibytes

Calculating file size: NOTE: Calculations must use the measurement of 1024, NOT 1000.

Size of image file

- Determine image resolution in pixels (width x height)
- Determine colour depth in bits (e.g. 8 bits for 256 colours)
- Multiply number of pixels by colour depth to get the total number of bits
- Divide total number of bits by 8 to get file size in bytes
- If necessary, convert to larger units like kibibytes, mebibytes, etc

● **File size of image (in bits)** = image resolution (in pixels) × colour depth (in bits)

Eg. An image measures 100 by 80 pixels and has 128 colours.

  - Resolution = 100 x 80
  - Colour depth = 7 bits for 128 colours
  - Total number of bits = 100 x 80 x 7
  - File size in bytes = (100 x 80 x 7) / 8 = 7000 bytes
  - 7000 bytes = 6.84 kibibytes

Size of sound file

- Determine sample rate in Hz
- Determine sample resolution in bits
- Determine length of track in seconds
- Multiply sample rate by sample resolution to get number of bits per second
- Multiply number of bits per second by length of track to get total number of bits
- Divide total number of bits by 8 to get file size in bytes
- If necessary, convert to larger units like kibibytes, mebibytes, etc

● **Size of a mono sound file (in bits)** =
  Sample rate (in Hz) × sample resolution (in bits) × length of the sample (in seconds)

● **Size of a stereo sound file (in bits)** =
  2[Sample rate × sample resolution × length of the sample]

Eg. A sound clip uses 48KHz sample rate, 24 bit resolution and is 30 seconds long.

  - Sample rate = 48×10^3 Hz
  - Sample resolution = 24 bit

- Length of track = 30s
- No of bits = $48 \times 10^3 \times 24 \times 30$
- No of bytes = $(48 \times 10^3 \times 24 \times 30) / 8 = 4320000$ bytes
- 4320000 bytes = 4.12 mebibytes

A company advertises its backup memory device as having 500 GB of storage. A customer wishes to know how many 8 MB files could be stored on the device. The company claimed that up to 62 500 files (assuming each file is 8 MB) could be stored. The customer calculated that 64 000 files could be stored. Explain the difference between these two storage values. Show any calculations you use in your explanation.

- company calculation is based on 1 GByte = 1000 MByte
- so $(500 \times 1000)/8 = 62\,500$ files
- customer calculation based on 1 GByte = 1024 MByte
- so $(500 \times 1024)/8 = 64\,000$ files
- giving the difference of 1500 file

Data Compression

Reducing the file size.

Purpose of data compression

- Less storage space required
- Less bandwidth required for transmission/streaming/upload/download
- Shorter transmission time // faster transmission/streaming/upload/download speed
- To make files small enough to attach to an email: sending/receiving email accounts may have restricted file size for attachments
- Less data usage is taken (for mobile clients) // Reduces costs when using cloud storage

Benefits of compressing image

- Quicker to transmit/upload/download
- Not as much bandwidth needed to transmit file
- To fit in limitation of file size on e.g. email

Lossy compression

Reduces the file size by permanently removing data, e.g. reducing resolution or colour depth, reducing sample rate or resolution.

How lossy compression compresses sound file

- Compression algorithm is used (eg. MP3)

- Removes unnecessary/ redundant data
- « like background noise / sounds humans can't hear
- « using perceptual musical shaping
- Reduces sample size / resolution
- Reduces sample rate
- Sound is clipped
- Some data will be lost/deleted permanently // original file cannot be restored

How lossy compression compresses image files

- Compression algorithm is used (eg. JPEG)
- Removes unnecessary/ redundant data
- « like details that human eye can't see
- Reduces colour depth/ range of colours used/ bits per pixel
- Reduces image resolution/ the number of pixels
- Some data will be lost/deleted permanently // original file cannot be restored

How lossy compression compresses video files

- Compression algorithm is used (MP4)
- Removes unnecessary/ redundant data
- « details/sound that human eye/ear cannot see/hear // perceptual music shaping
- Reduces colour depth/ range of colours used/ bits per pixel
- Reduces image resolution/ the number of pixels
- Reduce sample rate
- Reduce sample resolution
- Removes repeated frames (only stores what changes between frames)
- Some data will be lost/deleted permanently // original file cannot be restored

Benefits of using lossy compression instead of lossless

- Lossy decreases the file size more
- Take up less storage space on web server/users' computer
- Quicker transmission speed for upload/download/send
- Takes up less bandwidth to download/upload
- No requirement for high quality; can still be a similar quality
- No requirement for files to be exactly same as original
- Uses less data allowance

Drawbacks of lossy compression

- Quality of the file will be reduced
- Original file cannot be restored

Lossless compression

Reduces file size without permanent loss of data, e.g. run length encoding (RLE).

<u>How lossless compression compresses text file</u>

- Compression algorithm is used
- ... such as RLE/run length encoding
- Repeated patterns are identified: eg. repeated words/characters/phrases
- ... and indexed
- ... and replaced with their index
- ... their positions are stored
- ... their number of occurrences is stored
- Files are downloaded as zip files/ may be converted to pdf
- No data is removed in the process // original file can be restored

<u>How lossless compression compresses sound files</u>

- Compression algorithm is used (eg. RLE)
- Repeated patterns in the music are identified
- ... and indexed/grouped
- No data will be removed // original file can be restored

<u>How lossless compressions compresses video file</u>

- Compression algorithm used (e.g. RLE)
- Repeating frames/pixels are identified
- ... and are indexed
- No data will be removed // original file can be restored

<u>Benefit of lossless compression</u>

- No loss of quality
- No loss of data
- Original file can be restored

<u>Drawback of lossless compression</u>

- Larger file size
- Take up more storage space on web server/users' computer
- Slower transmission speed for upload/download/send: may cause buffering
- Takes up more bandwidth to stream/download/upload: more expensive
- Uses less more allowance

Applications of lossy & lossless compression

- Lossy compression used for media files: minor data loss is acceptable

- Lossless compression used for text/ code/ program files
    - No data can be lost
    - Will not run correctly if any other compression method is used