COMPUTER SCIENCE

O Level Notes

Syllabus 2210 / 0478

Prepared by

Khaled Bin Aziz

01756922708

Chapter 1: Data Representation

1.1 Number Systems

1. Data Representation

1.1. Definition

Data Representation is the method of storing data inside a computer using **binary (0 and 1).**

> **Important:** Computers understand only **binary.**

2. Why Do Computers Use Binary?

A computer contains millions of tiny electronic switches. Each switch has only two states:

Switch	Binary
ON	1
OFF	0

Therefore computers use the **Binary Number System (Base 2).**

> **Exam Tip: Q: Why do computers use binary?**
> - Electronic switches have only two states.
> - Binary has only two digits (0 and 1).
> - Therefore binary is reliable and easy for computers to process.

3. Number Systems

System	Base	Digits
Binary	2	0, 1
Denary	10	0–9
Hexadecimal	16	0–9, A–F

4. Binary Place Values

An 8-bit binary number has 8 columns. Each column represents a power of 2.

128	64	32	16	8	4	2	1

Example 1: Convert 11101110_2 to denary

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

Only add the place values where the binary digit is 1:

$$128 + 64 + 32 + 8 + 4 + 2 = \mathbf{238}$$

Therefore:

$$11101110_2 = 238_{\{10\}}$$

> **Exam Tip:** Only add the place values where the binary digit is 1.

5. Denary → Binary

There are two methods to convert a denary number into binary.

5.1. Method 1 — Subtraction

Convert 142_{10} to binary.

Subtract the largest possible powers of 2:

```
142 - 128 = 14
14  - 8   = 6
6   - 4   = 2
2   - 2   = 0
```

Binary: `10001110`

5.2. Method 2 — Division by 2

1. Divide by 2.
2. Write the remainder.
3. Continue until the quotient becomes 0.
4. Read remainders from **bottom to top**.

```
59 ÷ 2 = 29 R1
29 ÷ 2 = 14 R1
14 ÷ 2 = 7  R0
7 ÷ 2  = 3  R1
3 ÷ 2  = 1  R1
1 ÷ 2  = 0  R1
```

Bottom → Top: `111011` → 8-bit answer: `00111011`

> **Exam Tip:** Read remainders from **bottom to top.**

6. Hexadecimal System

Hexadecimal is **Base 16.**

Digits: `0 1 2 3 4 5 6 7 8 9 A B C D E F`

Hex	Denary

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

7. Why is Hexadecimal Used?

Binary numbers become very long.

```
Binary:      1101001010101111
Hexadecimal: D2AF
```

Advantages:
- Easier to read
- Easier to remember
- Easier to write
- Less chance of errors

8. Binary ↔ Hexadecimal

> **Exam Tip: Rule:** 1 Hex digit = 4 Binary bits

8.1. Binary → Hex

Split into groups of four bits from the right.

```
101111100001
1011  1110  0001
 B     E     1
```

Answer: `BE1`

8.2. Hex → Binary

Replace each hexadecimal digit with four binary bits.

```
4 → 0100
5 → 0101
A → 1010
```

Answer: `010001011010`

9. Binary–Hex Table

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

> **Exam Tip:** Memorise this table.

10. Hexadecimal → Denary Conversion

To convert a hexadecimal number into denary, multiply each hexadecimal digit by its place value and add the results.

Example: Convert `45A` to denary.

Hexadecimal place values:

Digit	4	5	A
Place Value	256	16	1

Step-by-step calculation:

$$4 \times 256 = 1024$$

$$5 \times 16 = 80$$

$$A \times 1 = 10$$

Remember:
- `A` in hexadecimal = `10` in denary

Adding all values:

$$1024 + 80 + 10 = \mathbf{1114}$$

Therefore:

$$45A_{\{16\}} = 1114_{\{10\}}$$

11. Denary → Hexadecimal

Use successive division by 16.

1. Divide by 16.
2. Record the remainder.
3. Continue until the quotient becomes 0.
4. Read remainders from bottom to top.

```
10 = A · 11 = B · 12 = C · 13 = D · 14 = E · 15 = F
```

12. Uses of Hexadecimal

12.1. Error Codes

Operating systems often display error codes in hexadecimal because it is:

- Shorter than binary
- Easier for programmers
- Helpful for identifying memory locations
- Useful during debugging

12.2. Other Uses

- MAC Addresses
- IPv6 Addresses
- HTML Colour Codes

13. Common Exam Questions

- Convert Binary → Denary
- Convert Denary → Binary
- Convert Binary → Hexadecimal
- Convert Hexadecimal → Binary
- Convert Hexadecimal → Denary
- Convert Denary → Hexadecimal
- State why hexadecimal is used instead of binary.
- State why computers use binary.

14. Memory Tricks

14.1. Binary Place Values

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

14.2. Hex Values

```
A=10  B=11  C=12  D=13  E=14  F=15
```

14.3. Conversion Rules

- Binary → Hex → Group into 4 bits.
- Hex → Binary → Each digit = 4 bits.
- Binary → Denary → Add place values.

- Denary ⟶ Binary ⟶ Divide by 2.
- Denary ⟶ Hex ⟶ Divide by 16.
- Hex ⟶ Denary ⟶ Multiply by place values.

15. Quick Revision

> - Binary = Base 2 (0,1)
> - Denary = Base 10
> - Hexadecimal = Base 16 (0–9, A–F)
> - Computers use binary because switches have only ON/OFF states.
> - 1 Hex digit = 4 Binary bits.
> - Binary ⟶ Denary = Add place values.
> - Denary ⟶ Binary = Divide by 2.
> - Denary ⟶ Hex = Divide by 16.
> - Hex ⟶ Denary = Multiply by place values.
> - Hexadecimal is easier to read, write and remember than binary.
> - Common uses: Error Codes, MAC Addresses, IPv6 Addresses, HTML Colour Codes.

O Level Computer Science

Chapter 3: Networking, Binary Arithmetic & Data Storage

MAC/IP Addresses, HTML Colours, Two's Complement, ASCII, Sound & Images

---

16. Media Access Control (MAC) Addresses

A **Media Access Control (MAC) address** is a unique number that identifies a device on a network.

The MAC address belongs to the **Network Interface Card (NIC)** inside the device.

Important points:

- Each device has a unique MAC address.
- It usually does not change.
- It allows a device to be identified on a network.

16.1. MAC Address Format

A MAC address is usually:

- 48 bits long.
- Written as 6 groups of hexadecimal digits.

`NN-NN-NN-DD-DD-DD`

`or`

`NN:NN:NN:DD:DD:DD`

The first three groups identify the **manufacturer**. The last three groups identify the **device serial number**.

Example: `00-1C-B3-4F-25-FE`

- Manufacturer code: `00-1C-B3`
- Device serial number: `4F-25-FE`

Example manufacturers:

Code	Company
00-14-22	Dell
00-40-96	Cisco
00-A0-C9	Intel

17. Internet Protocol (IP) Addresses

Every device connected to a network is assigned an **Internet Protocol (IP) address**.

IP addresses allow devices to communicate across networks.

17.1. IPv4 Addresses

IPv4 is a **32-bit** address, normally written using denary values.

Example: `109.108.158.1`

IPv4 can also be represented using hexadecimal: `77.76.9E.01`

17.2. IPv6 Addresses

IPv6 was introduced because IPv4 addresses were limited.

IPv6 uses:

- 128 bits
- Hexadecimal numbers
- Groups separated by colons

Example: `A8FB:7A88:0110:00FF:3D21:2085:66FB:F0FA`

Version	Format
IPv4	Uses dots (.)
IPv6	Uses colons (:)

18. HTML Colour Codes

HTML is a markup language used to create web pages.

HTML uses tags to define how information appears.

Example:

```
<h1 style="color:#FF0000;">
This is a red heading
</h1>
```

18.1. Hexadecimal Colour Representation

Computer screens create colours using three primary colours:

- Red
- Green
- Blue

These are known as **RGB values**.

HTML colour codes use the format `#RRGGBB`, where each colour component uses two hexadecimal digits.

Examples:

Hex Code	Colour
#FF0000	Red
#00FF00	Green
#0000FF	Blue
#FF00FF	Fuchsia
#FF8000	Orange
#B18904	Tan

18.2. RGB Colour Range

Each RGB value has **256 possible values** (`00` – `FF`) for Red, Green, and Blue.

Total possible colours:

$$256 \times 256 \times 256 = \mathbf{16,777,216}$$

19. Addition of Binary Numbers

Binary addition follows the same rules as denary addition. The main difference is that a carry happens when the total is greater than 1.

19.1. Binary Addition Rules

Calculation	Carry	Sum
0 + 0	0	0
0 + 1	0	1
1 + 0	0	1
1 + 1	1	0

19.2. Adding Three Binary Digits

Calculation	Carry	Sum
0+0+0	0	0
0+0+1	0	1
0+1+0	0	1
0+1+1	1	0
1+0+0	0	1
1+0+1	1	0
1+1+0	1	0
1+1+1	1	1

19.3. Example Binary Addition

```
  00100111
+ 01001010
-----------
  01110001
```

Answer: `01110001`

20. Overflow

Overflow happens when the result of a binary addition is too large to fit inside the available number of bits.

Example: adding two 8-bit numbers:

```
  01101110
+ 11011110
```

The result creates a 9th bit, which the computer cannot store. This causes an **Overflow Error.**

20.1. Maximum Values

Bits	Maximum Value
8-bit	$2^8 - 1 = \mathbf{255}$
16-bit	$2^{16} - 1 = \mathbf{65,535}$
32-bit	$2^{32} - 1 = \mathbf{4,294,967,295}$

21. Logical Binary Shifts

A logical shift moves binary digits left or right. Empty positions are filled with **0**.

Shift	Effect
Left shift	Multiply by 2
Right shift	Divide by 2

21.1. Left Shift Example

`00010101` (denary 21)

- Shift left once ⟶ `00101010` (value 42)
- Shift left twice ⟶ `01010100` (value 84)

21.2. Right Shift Example

`11001000` (denary 200)

- Shift right once ⟶ `01100100` (value 100)
- Shift right twice ⟶ `00110010` (value 50)

22. Two's Complement Representation

Two's complement is used to represent negative binary numbers.

For an 8-bit number, the place values are:

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

If the left-most bit is 1, the number is negative.

Example: `10010011`

$$-128 + 16 + 2 + 1 = \mathbf{-109}$$

23. Converting Negative Denary to Binary

Method:

1. Convert the positive number into binary.
2. Invert all bits.
3. Add 1.

Example: convert `-67`

```
Positive binary: 01000011
Invert:          10111100
Add 1:           10111101
```

Answer: `10111101`

> **Exam Tip:** To convert a negative denary number to binary: convert the positive value, invert every bit, then add 1.

24. ASCII Code

ASCII stands for **American Standard Code for Information Interchange.**

ASCII represents:

- Letters
- Numbers
- Symbols
- Control characters

Type	Bits	Range
Standard ASCII	7 bits	0 – 127
Extended ASCII	8 bits	0 – 255

25. Unicode

Unicode is a character encoding system that supports languages worldwide.

Advantages:

- Supports many languages.
- Supports thousands of symbols.
- Used by modern operating systems and browsers.

ASCII only supports a limited number of characters. Unicode can use **16-bit** or **32-bit** encoding.

26. Representation of Sound

Sound is created by vibrations in the air. Sound waves are **continuous** and **analogue.**

Computers cannot store analogue data directly — sound must be converted into digital form using an **Analogue to Digital Converter (ADC).**

26.1. Sampling

Sampling measures the amplitude of a sound wave at regular intervals.

1. Measure sound amplitude.
2. Convert measurements into binary.
3. Store binary values.

26.2. Sampling Resolution

Sampling resolution is the number of bits used to represent each sound sample.

> Higher resolution → better sound quality, but larger file size.

26.3. Sampling Rate

Sampling rate is the number of samples taken every second, measured in **Hz.**

Example — CD quality:

- 16-bit resolution
- 44.1 kHz sampling rate

27. Bitmap Images

Bitmap images are made from **pixels.** Each pixel is stored as binary data.

27.1. Colour Depth

Colour depth is the number of bits used to represent each pixel.

Bits	Colours
1 bit	2 colours
2 bits	4 colours
8 bits	256 colours
24 bits	Over 16 million colours

27.2. Image Resolution

Resolution is the number of pixels in an image.

Example: `4096 × 3072` pixels

> Higher resolution means more detail, but a larger file size.

28. Data Storage Units

A **bit** is the smallest unit of data (`0` or `1`).

- A **byte** contains 8 bits.
- A **nibble** contains 4 bits.

28.1. Storage Measurements

Unit	Bytes
KB	1,000 bytes
MB	1,000,000 bytes
GB	1,000,000,000 bytes
TB	1,000,000,000,000 bytes

29. Quick Revision

- MAC addresses identify network devices.
- IPv4 uses 32 bits.
- IPv6 uses 128 bits.
- HTML colours use hexadecimal RGB values.
- Binary addition uses carry values.
- Overflow happens when a number exceeds available bits.
- Logical shifts multiply or divide by 2.
- Two's complement represents negative numbers.
- ASCII represents characters.
- Unicode supports worldwide languages.
- Sound is stored using sampling.
- Images are stored using pixels.
- Colour depth affects image quality.
- Resolution affects image detail.
- Storage is measured in bytes and larger units.

30. Calculation of File Size

File size calculations are used to find the amount of storage required for:

- Bitmap images
- Sound files

30.1. Bitmap Image File Size

Formula:

```
Image resolution × Colour depth
```

Steps:

1. Calculate total number of pixels. Width × Height
2. Multiply by colour depth.
3. Convert bits into bytes. Bits ÷ 8

30.1.1. Example 1

A photograph is:

1024×1080 pixels

Colour depth = 32 bits

Step 1:

Total pixels:

1024×1080

$= 1,105,920$ pixels

Step 2:

Multiply by colour depth:

$1,105,920 \times 32$

$= 35,389,440$ bits

Convert into bytes:

$35,389,440 \div 8$

$= 4,423,680$ bytes

Memory stick:

64 GiB

Convert:

$64 \times 1024 \times 1024 \times 1024$

$= 68,719,476,736$ bytes

Number of photographs:

$68,719,476,736 \div 4,423,680$

$= 15,534$ photographs

30.1.2. Example 2

Camera:

2048×2048 pixels

Colour depth = 16 bits

Total pixels:

2048×2048

$= 4,194,304$ pixels

Multiply by colour depth:

$4,194,304 \times 16$

$= 67,108,864$ bits

Convert into bytes:

$67,108,864 \div 8$

$= 8,388,608$ bytes

Convert into MiB:

$8,388,608 \div 1,048,576$

$= 8$ MiB

31. Sound File Size Calculation

Formula:

```
Sample rate × Sample resolution × Length
```

For stereo sound:

Multiply by 2

Convert:

Bits ÷ 8 = Bytes

31.0.1. Example 3

Audio CD:

- Sample rate = 44,100 Hz
- Sample resolution = 16 bits
- Length = 60 minutes
- Channels = 2

Convert time:

60×60

$= 3,600$ seconds

Calculate size:

$44,100 \times 16 \times 3,600$

$= 2,540,160,000$ bits

Stereo:

$2,540,160,000 \times 2$

$= 5,080,320,000$ bits

Convert into bytes:

$5,080,320,000 \div 8$

$= 635,040,000$ bytes

Convert into MiB:

$635,040,000 \div 1,048,576$

$= 605$ MiB

32. Data Compression

Large image and sound files require compression.

Reasons for compression:

1. Saves storage space.
2. Reduces upload and download time.
3. Improves streaming speed.
4. Reduces bandwidth usage.
5. Lowers storage costs.

Compressed files contain fewer bits compared to the original file.

33. Lossy and Lossless Compression

There are two types of compression:

- Lossy compression
- Lossless compression

34. Lossy Compression

Lossy compression removes unnecessary data from a file.

The original file cannot be completely restored.

Advantages:
1. Smaller file size.
2. Faster transfer.
3. Requires less storage.

Disadvantages:
1. Some quality is lost.

Examples:

1. MP3
2. MP4
3. JPEG

34.1. MP3 Compression

MP3 is used for music files.

It reduces file size by removing sounds that humans cannot hear easily.

Methods:

1. Removes sounds outside the human hearing range.
2. Removes quieter sounds when louder sounds are played at the same time.

This is called:

Perceptual music shaping

34.2. MP4 Compression

MP4 stores multimedia files.

It can contain:

1. Audio
2. Video
3. Images
4. Animation

MP4 uses lossy compression while maintaining acceptable quality.

34.3. JPEG Compression

JPEG is used for bitmap images.

JPEG reduces file size by:

1. Removing unnecessary colour information.
2. Separating brightness from colour information.
3. Dividing images into blocks.

The original image cannot be fully recovered after compression.

35. Lossless Compression

Lossless compression allows the original file to be reconstructed completely.

Used when no data loss is acceptable.

Examples:

1. Computer applications.
2. Large spreadsheets.
3. Important documents.

A common lossless method:

Run-Length Encoding (RLE)

36. Run-Length Encoding (RLE)

RLE reduces repeated data.

It stores:

```
Number of repetitions + Data value
```

Example:

Original:

```
aaaaabbbbccddddd
```

RLE:

```
05 97
04 98
02 99
05 100
```

Explanation:

- `05 97` = five characters with ASCII value 97
- `04 98` = four characters with ASCII value 98

Original size:

16 bytes

Compressed size:

8 bytes

37. RLE Flags

RLE is not effective when data does not repeat.

Example:

`cdcdcdcdcd`

A flag can be used to identify compressed data.

Example:

`255 05 97`

Meaning:

- `255` = flag
- `05` = number of repeats
- `97` = data value

38. RLE With Images

RLE can compress bitmap images with repeated colours.

Example:

A black and white image:

- White = W
- Black = B

Compressed form:

`9W 6B 2W 1B`

Using binary values:

- W = 1
- B = 0

Compressed:

`91 60 21 10`

Benefits:
1. Reduces storage size.
2. Works well with repeated patterns.

39. Binary Coded Decimal (BCD)

Binary Coded Decimal represents each denary digit using 4 bits.

Table:

BCD	Denary
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9

Example:

Denary:

`3165`

BCD:

```
0011 0001 0110 0101
```

39.1. Uses of BCD

BCD is used in:

1. Digital clocks.
2. Calculators.
3. Number displays.

Each denary digit has its own binary representation.

40. Subtraction Using Two's Complement

Binary subtraction can be performed by:

1. Converting the second number into two's complement.
2. Adding the numbers together.

40.1. Example

Calculate:

$95 - 68$

Convert to binary:

$95 = 01011111$

$68 = 01000100$

Find two's complement of 68:

Invert:

10111011

Add 1:

10111100

Now add:

```
  01011111
+ 10111100
-----------
 100011011
```

Ignore the extra ninth bit:

00011011

Answer:

27

41. Key Terms

Term	Meaning
Bit	A binary digit containing 0 or 1
Binary Number System	Base 2 number system using 0 and 1
Hexadecimal	Base 16 system using 0–9 and A–F
MAC Address	Unique hexadecimal address identifying a network device
IP Address	Address used to identify devices on a network
HTML	Language used to create web pages
Overflow Error	Error caused when a value is too large for available bits
Logical Shift	Moving binary bits left or right
Two's Complement	Method of representing negative binary numbers
ASCII	Character encoding system
Unicode	Character system supporting worldwide languages
Sampling Resolution	Number of bits used for each sound sample
Colour Depth	Number of bits used for each image pixel
Sampling Rate	Number of samples taken per second
Bitmap Image	Image made from pixels
Compression	Reducing file size
Lossy Compression	Compression where some data is removed
Lossless Compression	Compression where original data can be restored
RLE	Lossless compression using repeated data values

42. Quick Revision

- Image file size = Resolution × Colour depth.
- Sound file size = Sample rate × Resolution × Time.
- Stereo sound requires multiplying by 2.
- Compression reduces file size.
- Lossy compression removes data permanently.
- Lossless compression restores the original file.
- MP3, MP4 and JPEG use lossy compression.
- RLE is a lossless compression method.
- BCD represents each denary digit using 4 bits.
- Two's complement represents negative numbers.
- Overflow happens when values exceed available bits.

43. Exam-style Questions

43.1. Question 1

A software developer is using a microphone to collect sounds for a game.

43.1.1. a) Sound Recording

43.1.1.1. i) What is sampling resolution?

Sampling resolution is the number of bits used to represent each sound sample.

43.1.1.2. ii) Effect of sampling resolution

A higher sampling resolution:

- uses more bits for each sample
- allows more possible amplitude values
- produces a more accurate representation of the original sound
- increases file size

43.1.2. b) Bitmap Images

43.1.2.1. i) Image resolution

Image resolution is the number of pixels that make up an image.

Example:

```
1920 × 1080 pixels
```

43.1.2.2. ii) 16-colour bitmap

Formula:

$$\text{Number of colours} = 2^n$$
$$16 = 2^4$$

Answer:

4 bits per pixel

43.1.2.3. iii) Image file size

Image:

- Width = 16,384 pixels
- Height = 512 pixels

Colour depth:

256 colours = 8 bits

Calculation:

$16,384 \times 512 \times 8$

$= 67,108,864$ bits

$\div 8$

$= 8,388,608$ bytes

$\div 1024^3$

$= 0.0078$ GiB

43.1.2.4. iv) File compression techniques

Two types of compression:

Lossy compression:

- removes unnecessary data
- reduces file size
- original file cannot be fully recovered

Examples:

1. JPEG
2. MP3
3. MP4

Lossless compression:

- keeps all original data
- original file can be restored

Example:

1. RLE

43.2. Question 2

A movie editor is sending music files by email.

43.2.1. a) Sampling

The sampling process:

1. Analogue sound waves are measured at regular intervals.
2. The amplitude of each sample is recorded.
3. Each sample is converted into binary values.
4. Binary data is stored as a digital sound file.

43.2.2. b) Type of compression

The most suitable compression is:

Lossy compression

Reason:

- music files are very large
- lossy compression greatly reduces file size
- some sound data removed cannot usually be noticed by humans

43.2.3. c) Run Length Encoding (RLE)

43.2.3.1. i) Definition

RLE is a lossless compression technique.

It replaces repeated data with:

```
number of repetitions + data value
```

Example:

Original:

```
AAAAABBBB
```

Compressed:

```
05A 04B
```

43.2.3.2. ii) Using RLE with images

RLE stores:

```
number of repeated pixels + pixel colour value
```

Example:

- White pixels = 1
- Grey pixels = 0

A sequence:

```
11110000
```

becomes:

```
41 40
```

43.3. Question 3

An 8-bit register contains:

```
00110110
```

43.3.1. a) Denary value

Column values:

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

Working:

$32 + 16 + 4 + 2$

$= \mathbf{54}$

43.3.2. b) Logical shift right

Original:

`00110110`

Shift right:

`00011011`

Denary:

$16 + 8 + 2 + 1$

$= \mathbf{27}$

43.3.3. c) Logical shift left two places

Original:

`00110110`

Shift left:

`11011000`

Effect:

The value is multiplied by:

$2^2 = 4$

Original:

$54 \times 4 = \mathbf{216}$

43.4. Question 4

43.4.1. a) Convert denary to 8-bit binary

43.4.1.1. 123

`01111011`

43.4.1.2. 55

`00110111`

43.4.1.3. 180

`10110100`

43.4.2. b) Binary addition

$123 + 55$

```
  01111011
+ 00110111
-----------
  10110010
```

Answer:

178

$123 + 180$

```
  01111011
+ 10110100
-----------
1 00101111
```

The ninth bit shows **overflow**.

43.4.3. c) Two's complement

Given:

`01110100`

Invert:

`10001011`

Add 1:

`10001100`

Answer:

$\mathbf{-116}$

Convert −112:

112:

`01110000`

Invert:

`10001111`

Add 1:

`10010000`

Answer:

`10010000`

Binary:

`10111001`

Two's complement:

$-128 + 32 + 16 + 8 + 1$

$= \mathbf{-71}$

43.5. Question 5

Bitmap image:

1140×1080 pixels

Colour depth = 24 bits

43.5.1. a) Pixel

A pixel is the smallest element of a bitmap image.

43.5.2. b) Colour depth

Colour depth is the number of bits used to represent the colour of one pixel.

43.5.3. c) Number of images stored

File size:

$1140 \times 1080 \times 24$

$= 29,548,800$ bits

Convert to bytes:

$29,548,800 \div 8$

$= 3,693,600$ bytes

32 GiB:

32×1024^3

$= 34,359,738,368$ bytes

Number of images:

$34,359,738,368 \div 3,693,600$

$= \mathbf{9,302}$

43.5.4. d) Increasing number of images

Methods:

- reduce image resolution
- reduce colour depth
- use compression

43.6. Question 6

43.6.1. a) Converting images into digital files

Process:

1. Camera sensor captures light.
2. Image is divided into pixels.
3. Each pixel colour is converted into binary.
4. Binary data is stored as an image file.

43.6.2. b) Suitable compression

Use:

Lossy compression

Reason:

- photos contain large amounts of data
- JPEG reduces file size significantly
- small quality losses are acceptable

43.7. Question 7

A stopwatch stores time values.

Current display:

```
02 : 31 : 58
```

Convert each pair into 8-bit binary:

Hours:

$2 = 00000010$

Minutes:

$31 = 00011111$

Seconds:

$58 = 00111010$

New register values:

```
00000101
00011010
00110111
```

Convert:

Hours:

5

Minutes:

26

Seconds:

55

Display:

```
05 : 26 : 55
```

43.8. Question 8

Memory stick:

64 GiB

Each photo:

10 KiB

43.8.1. a) Number of photographs

64 GiB:

$64 \times 1024 \times 1024$ KiB

$= 67,108,864$ KiB

Photos:

$67,108,864 \div 10$

$= \mathbf{6,710,886}$ photographs

43.8.2. b) Reducing file size

Method:

Compression

Advantages:

- more files can be stored
- faster transfer
- less storage space needed

Disadvantage:

Lossy compression can reduce image quality.

43.8.3. c) Bitmap images

3 bytes are needed because:

- 1 byte = Red intensity
- 1 byte = Green intensity
- 1 byte = Blue intensity

Total:

3 bytes per pixel

Number of colours:

Each colour uses:

8 bits

Therefore:

- Red = 256 values
- Green = 256 values
- Blue = 256 values

Total:

$256 \times 256 \times 256$

$= \mathbf{16,777,216}$ colours

44. Quick Practice Questions

Convert:

$59,055,800,320$ bytes

into GiB.

Two's complement:

`00101101`

Denary value:

$32 + 8 + 4 + 1$

$= \mathbf{45}$

If:

$2^x = 1,048,576$ bytes

Then:

$2^{20} = 1,048,576$

Answer:

$x = \mathbf{20}$

Hexadecimal:

`3F`

Conversion:

$3 \times 16 + 15$

$= \mathbf{63}$

Binary addition:

```
  00010011
+ 00011011
-----------
  00101110
```

Denary:

46

Denary to hexadecimal:

Denary	Hexadecimal
40	28
57	39
60	3C