A1.2.1 – Principal Methods of Representing Data

Binary representation

- Computers represent all types of data (numbers, text, sound, images, video) using only **two digits: 0 and 1**.
  Explanation: Inside every computer, the CPU and memory work with electrical signals. A signal can be **on (1)** or **off (0)** — this is why computers use binary.

Binary Digit	Meaning	Electrical State
0	Off	Low voltage
1	On	High voltage

- A **bit** is one binary digit (0 or 1).
- A **byte** is a group of **8 bits**.
  Explanation: One byte can represent 256 possible values (2^8). All information stored on a computer is made up of bytes.
  Example: The letter **'A'** in binary is **01000001**, which equals 65 in decimal.
- **Why binary matters:** Binary is universal. Every computer, from a calculator to a supercomputer, uses the same two-state logic to represent and process information.

Number representation

- Computers use three main systems to represent numbers: **binary**, **decimal**, and **hexadecimal**.

System	Base	Digits used	Example	Used by
Binary	2	0, 1	$1011_2 = 11_{10}$	Computers
Decimal	10	0–9	$472_{10} = 4×100 + 7×10 + 2$	Humans
Hexadecimal	16	0–9, A–F	$11110010_2 = F2_{16}$	Programmers

- **Binary:** The simplest way a computer stores numbers.
  Example: 1011_2 means (1×8) + (0×4) + (1×2) + (1×1) = **11**$_{10}$.

- **Hexadecimal:** Used to shorten long binary numbers.
  Explanation: Each hex digit represents 4 binary bits.
  Example: $1111_2 = F_{16}$.
  Real-world examples:
  - Memory addresses: `0x7FF3A9`
  - Web colors: `#FF0000` (red)

Text representation

- Text is stored using **character encoding** — a system that assigns each symbol a unique binary value.

ASCII (American Standard Code for Information Interchange):

Character	Decimal	Binary
A	65	01000001
a	97	01100001
0	48	00110000

Explanation: ASCII uses 7 bits (0–127) to represent English letters, numbers, and symbols.

- **Limitation:** ASCII cannot represent characters from non-English languages.

Unicode:

- Unicode extends ASCII and can represent **all global characters**, including Chinese, Arabic, emojis, and mathematical symbols.
  Example: The Chinese character “中” = U+4E2D.

Encoding	Bit Length	Characters Supported	Example
ASCII	7 bits	128	A = 01000001
UTF-8	8–32 bits	1.1 million	😃 = U+1F603
UTF-16	16–32 bits	65,000+	中 = U+4E2D

Real-world example: Unicode ensures that your name written in Hindi, Arabic, or Chinese displays correctly on any device.

Image representation

- Images are made up of small dots called **pixels (picture elements)**. Each pixel is stored as a binary value that represents its color or brightness.

Type	Bits per pixel	Description	Example Use
Monochrome	1	Black (0) or white (1)	Simple icons
Grayscale	8	256 shades of gray	Medical imaging
Color (RGB)	24	8 bits for red, green, and blue	Digital photos

- **RGB (Red, Green, Blue) color model:**
  Explanation: Each pixel's color is created by combining red, green, and blue light intensities.
  Example: Pure red = R=255, G=0, B=0 → Binary: 11111111 00000000 00000000.

- **Image size formula:**
  `File size (bits) = width × height × color depth`
  Example: A 1920×1080 image with 24-bit color = 1920 × 1080 × 24 = 49,766,400 bits (≈ 6 MB).

  Real-world examples:

  - Smartphone cameras use RGB color depth to produce realistic photos.
  - MRI scans use grayscale to show subtle differences in tissue density.

Sound representation

- Sound is **analog**, meaning it is continuous. To store it digitally, it must be **sampled** — measured at regular intervals.

Key Terms:

Term	Definition	Unit	Example
Sampling	Measuring the sound wave's amplitude	—	Converts analog to digital
Sampling Rate	Samples taken per second	Hertz (Hz)	CD = 44,100 Hz
Bit Depth	Bits used per sample	bits	16-bit = 65,536 levels
Channel	Independent audio stream	—	Stereo = 2 channels

- **How sound is stored:**
  Each sample's amplitude is stored as binary. The higher the sampling rate and bit depth, the better the quality.

- **File size formula:**
  `File size = Sampling rate × Bit depth × Duration × Channels`
  Example: 44,100 Hz × 16-bit × 60 sec × 2 channels = 84,672,000 bits = 10.6 MB.

- **Real-world examples:**

  - MP3 compresses files by removing inaudible frequencies.

  - WAV format stores uncompressed audio for professional editing.

Video representation

- A video is a **series of images (frames)** shown quickly to create motion, combined with audio.

Factor	Description	Example
Frame rate	Frames per second (fps)	24–60 fps typical
Resolution	Number of pixels per frame	1920×1080 (Full HD)
Color depth	Bits per pixel	24-bit color
Compression	Reduces file size	H.264, H.265 codecs

- **How compression works:**
  Instead of storing every frame completely, only changes between frames are saved.

- **Real-world examples:**
  - YouTube and Netflix use advanced compression (H.265) to stream 4K video efficiently.
  - Smartphones record video at 60 fps for smoother playback.

Measuring data

Unit	Symbol	Conversion	Notes
Bit	b	Smallest unit (0 or 1)	Used for speed (Mbps)
Byte	B	8 bits	Basic storage unit
Kilobyte	KB	1,000 bytes	Decimal system
Kibibyte	KiB	1,024 bytes	Binary system
Megabyte	MB	1,000 KB	Decimal
Mebibyte	MiB	1,024 KiB	Binary
Gigabyte	GB	1,000 MB	Decimal

Example: A “1 TB” drive (1,000,000,000,000 bytes) appears as **931 GiB** in Windows because the computer uses binary (1 GiB = 1,073,741,824 bytes).

Summary

Data Type	Representation	Key Concept	Real-world Example
Number	Binary, Hexadecimal	Base conversion	Memory addresses
Text	ASCII, Unicode	Character encoding	Multilingual names
Image	Pixels, RGB	Resolution, color depth	Photography, MRI
Sound	Sampling, Bit depth	Quality vs file size	MP3, WAV
Video	Frames, Compression	Motion and storage	YouTube, Netflix

Revision Questions

Define bit and byte.
Explain why computers use binary to represent data.
Convert 10011010_2 to decimal.
Convert 56_{10} to binary and hexadecimal.
Explain why hexadecimal notation is used in computer systems.
Define ASCII and Unicode.
Outline two differences between ASCII and Unicode.
Explain why Unicode requires more storage than ASCII.
Describe how color images are represented using binary.
Calculate the file size of a 200×200 image with 8-bit color depth.
Define sampling rate and bit depth.
Explain how increasing sampling rate affects sound quality and file size.
Outline two factors that affect video file size.
Explain why compression is essential in video streaming.
Distinguish between kilobyte and kibibyte.
Explain why a 1 TB hard drive shows less space when connected to a computer.

A1.2.2 – Number Systems and Conversions

Overview
Computers store and process all information in **binary**, but humans commonly use **decimal**, and programmers often use **hexadecimal** for readability.
Understanding how to convert between these systems is essential for data representation, programming, and debugging.

1. Decimal (Base 10)

- **Digits used:** 0–9
- **Base:** 10
- **Place value:** Each digit's value depends on its position, increasing by powers of 10 from right to left.

Position	**10^2**	**10^1**	**10^0**
Number	3	4	7

Example:
$347_{10} = (3 \times 100) + (4 \times 10) + (7 \times 1) =$ **347**

Explanation: This is the number system humans use daily — money, distance, time, etc.

2. Binary (Base 2)

- **Digits used:** 0 and 1
- **Base:** 2
- **Place value:** Each digit represents a power of 2.

Position	**2^7**	**2^6**	**2^5**	**2^4**	**2^3**	**2^2**	**2^1**	**2^0**
Example	1	0	1	1	0	0	1	0

Conversion:
10110010_2 = (1×128) + (0×64) + (1×32) + (1×16) + (0×8) + (0×4) + (1×2) + (0×1) = **178_{10}**

Explanation: Computers use binary because circuits can easily detect **two states — ON and OFF**.
Real-world example: Data stored on a hard drive or transmitted through a network is always in binary form.

3. Hexadecimal (Base 16)

- **Digits used:** 0–9 and A–F
  (A = 10, B = 11, C = 12, D = 13, E = 14, F = 15)
- **Base:** 16
- **Purpose:** Simplifies the representation of long binary strings.

Decimal	Binary	Hexadecimal
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Example:
Binary: 11011110_2 → Group bits into 4s: (1101)(1110) → D and E → **DE_{16}**

Explanation: Each hexadecimal digit represents **four binary bits (a nibble)**, making it much shorter and easier to read.
Real-world example:

- Web colors use hex (e.g. #FF5733 = orange).
- Memory addresses in programming (e.g. 0x7FF4A2).

4. Conversion Methods

Decimal → Binary

Method 1: Repeated Division by 2

1. Divide the decimal number by 2.
2. Record the remainder (0 or 1).
3. Continue dividing the quotient by 2 until it becomes 0.
4. Write the remainders **in reverse order**.

Example: Convert 25_{10} to binary

Step	Division	Quotient	Remainder
1	25 ÷ 2	12	1
2	12 ÷ 2	6	0
3	6 ÷ 2	3	0
4	3 ÷ 2	1	1
5	1 ÷ 2	0	1

Answer: 25_{10} = $\mathbf{11001_2}$

Method 2: Subtract Powers of 2
Find the largest power of 2 ≤ the number, mark as 1, and subtract.
Example: 19_{10} = 16 + 2 + 1 = $\mathbf{10011_2}$

Binary → Decimal

Multiply each bit by its power of 2 and sum the results.

Example: $10101_2 = (1\times16) + (0\times8) + (1\times4) + (0\times2) + (1\times1) = \mathbf{21}_{10}$

Binary → Hexadecimal

Group the binary digits into groups of four (from the right).
Convert each group to its hexadecimal value.

Example:
Binary: $11100101_2 \rightarrow (1110)(0101) \rightarrow E5_{16}$

Hexadecimal → Binary

Convert each hex digit into its 4-bit binary form.
Example:
$A7_{16}$ = A → 1010, 7 → 0111 → $\mathbf{10100111}_2$

Decimal → Hexadecimal

Method: Repeated Division by 16

1. Divide the number by 16.
2. Record the remainder (convert 10–15 into A–F).
3. Continue dividing until quotient = 0.
4. Write remainders in reverse order.

Example: Convert 254_{10} to hex

Step	Division	Quotient	Remainder
1	254 ÷ 16	15	14 (E)
2	15 ÷ 16	0	15 (F)

Answer: 254_{10} = **FE**$_{16}$

Hexadecimal → Decimal

Multiply each digit by powers of 16 and add.
Example: $2F_{16}$ = (2×16) + (15×1) = **47**$_{10}$

5. Binary Arithmetic (Addition)

Binary follows the same rules as decimal addition, except only 0 and 1 are used.

Operation	Result	Carry
0 + 0	0	0
0 + 1	1	0
1 + 0	1	0
1 + 1	0	1 (carry 1)

Example:

```
 1011

+ 1101

------

11000
```

Explanation: 11 (carry 1) at each step, just like normal addition.

Real-world example:
Binary arithmetic is the basis for all CPU operations — addition, subtraction, and logical processing are built using binary circuits.

6. Signed Binary Numbers

Computers must represent both **positive and negative** values.

Sign-and-Magnitude

- The leftmost bit (MSB) shows the sign:
  0 = positive, 1 = negative.
- Remaining bits store the magnitude.

Example:
+5 = 0101, −5 = 1101

Limitation: Two zeros exist (+0 and −0), which complicates calculations.

Two's Complement

- Widely used system to represent negative numbers.
- Steps to find the negative of a number:
  1. Write the binary number.

2. Invert all bits (0→1, 1→0).
3. Add 1.

Example:
+5 = 0101 → invert → 1010 → add 1 → **1011 = −5**

Advantage: Only one zero exists, and binary addition/subtraction work normally.

Real-world example:
CPUs use two's complement to perform arithmetic quickly using the same adder circuits.

7. Floating-Point Representation (Introductory Concept – SL Overview)

Used to store **real numbers (with decimals)** using scientific notation in binary form.

Example:
Decimal: 6.25 → Binary: 110.01 → Represented as **1.1001×2^2**

Component	Description	Example
Sign bit	0 = positive, 1 = negative	0
Exponent	Position of the binary point	2
Mantissa	Actual digits of the number	1.1001

Real-world example:
Used for representing numbers like 3.14, 0.0001, or 9.8 in scientific or engineering calculations.

8. Importance of Number Systems

- Binary simplifies digital design — 2 states = on/off.
- Hexadecimal provides human-friendly representation.
- Conversions are vital in:
  - Programming memory addresses
  - Debugging machine code
  - Networking (e.g., IP and MAC addresses)
  - Graphic design (color encoding)

Revision Questions

Define the term “base” in the context of number systems.
Convert 47_{10} to binary.
Convert 101011_2 to decimal.
Convert 10011100_2 to hexadecimal.
Convert $2A_{16}$ to binary.
Explain one advantage of using hexadecimal instead of binary.
Outline the steps to convert a decimal number to binary.
Explain how two’s complement represents negative numbers.
Convert −6 to its 4-bit two’s complement form.
Describe one advantage of two’s complement over sign-and-magnitude representation.
Perform the binary addition: 1010 + 1101.
Convert 255_{10} to hexadecimal.
Define the term “floating-point representation.”
Explain why floating-point representation is useful for scientific calculations.
Discuss how understanding number systems is essential in computer programming.

A1.2.3 – Data Compression

1. What is Data Compression?

- **Definition:**
  Data compression is the process of **reducing the size of a file or data set** so that it requires less storage space and can be transmitted faster over networks.

- **Explanation:**
  Compression works by removing redundancy — repeated or unnecessary data — while preserving as much of the original information as possible.

- **Real-world examples:**

  - Compressing a large photo before sending via email.

  - Streaming platforms (like Netflix or Spotify) compress videos and audio for smooth playback.

  - File formats like ZIP and RAR compress multiple files into one smaller file.

File Type	Original Size	Compressed Size	Format Used
Image	5 MB	1 MB	JPEG
Audio	10 MB	3 MB	MP3
Video	500 MB	100 MB	MP4 (H.264 codec)

2. Why Data Compression is Important

- **Faster transmission:** Smaller files upload, download, and stream faster.

- **Reduced storage cost:** Saves disk space on servers, mobile devices, and cloud storage.

- **Efficient bandwidth use:** Less data sent over networks, improving performance and reducing costs.

- **Practical applications:**
    - Sending large attachments via email.
    - Faster website loading.
    - Mobile apps running on limited network data.

Real-world example:
When sending an image through WhatsApp, the app automatically compresses it to send faster without noticeably reducing quality.

3. Two Main Types of Data Compression

Type	Description	Data Loss	Example Formats	Typical Uses
Lossless	No data is lost; the original file can be perfectly reconstructed.	No	ZIP, PNG, GIF	Text, executables, documents
Lossy	Some data is lost to achieve much higher compression rates.	Yes	MP3, JPEG, MP4	Images, audio, video

4. Lossless Compression

- **Definition:**
  A method that **reduces file size without losing any data**. After decompression, you get an exact copy of the original.
- **How it works:**
  Lossless algorithms find patterns or repetitions and store them efficiently.

Common Lossless Methods

1. **Run-Length Encoding (RLE)**
    - Works by replacing sequences of repeating data with a shorter representation.
    - Example:
      Original: `AAAAABBBBCCCC`

Compressed: `5A4B4C`

- **Use case:** Simple images, icons, and black-and-white scans.
- **Limitation:** Not effective if data has few repetitions.

2. **Huffman Coding**
   - Assigns **shorter binary codes** to more frequent characters and **longer codes** to less frequent ones.
   - Example:
     If A appears often, it might get code "10";
     If Z appears rarely, it might get "11101".
   - **Real-world use:** ZIP files, PNG images, and text compression.
3. **Dictionary-Based Compression (LZW)**
   - Builds a dictionary of repeating patterns or strings.
   - Example:
     The word "computer" repeated many times is stored once with references to it.
   - **Used in:** GIF and TIFF file formats.

Advantages of Lossless Compression:

- No quality loss.
- Suitable for text, data, and executable files.

Disadvantages:

- Lower compression ratio compared to lossy methods.
- Files may remain relatively large.

5. Lossy Compression

- **Definition:**
  A compression method that **removes some data permanently** to reduce file size drastically.

- **Explanation:**
  The algorithm identifies parts of the data that are less noticeable to humans (e.g., colors we can't distinguish or sounds we can't hear) and removes them.

Common Lossy Methods

1. **JPEG (Joint Photographic Experts Group)**
   - Used for images.
   - Removes tiny color differences that the human eye cannot detect.
   - **Example:**
     A 5 MB photo can be reduced to 1 MB with little visible difference.
   - **Limitation:** Repeated compression can cause visible artifacts (blurring or blockiness).
2. **MP3 (MPEG Audio Layer III)**
   - Used for audio compression.
   - Removes inaudible frequencies (e.g., sounds beyond 20 kHz).
   - **Example:**
     A 50 MB WAV file can be compressed to 5 MB MP3.
   - **Real-world example:** Music streaming on Spotify and Apple Music.
3. **MPEG-4 (H.264 / H.265)**
   - Used for video compression.
   - Stores only the **changes between consecutive frames** rather than each frame completely.

    - **Example:**
      A 1 GB video compressed to 200 MB with almost no visible quality loss.
    - **Used in:** YouTube, Netflix, and Blu-ray video encoding.

Advantages of Lossy Compression:

- Significantly reduces file sizes.
- Ideal for multimedia storage and streaming.

Disadvantages:

- Some data is permanently lost.
- Repeated compression can worsen quality.
- Not suitable for critical files (e.g., text, software, documents).

6. Comparison Between Lossless and Lossy Compression

Feature	Lossless	Lossy
Data recovery	Perfectly reversible	Irreversible
Quality	Identical to original	Slightly reduced
File size	Larger	Smaller
Best used for	Text, documents, code, PNG images	Audio, video, JPEG images
Examples	ZIP, GIF, PNG	MP3, JPEG, MP4

7. Compression Ratio

- **Definition:** The ratio between the original file size and the compressed file size.
  Formula:
  Compression Ratio=Original SizeCompressed Size\text{Compression Ratio} = \frac{\text{Original Size}}{\text{Compressed Size}}Compression Ratio=Compressed SizeOriginal Size

Example:
Original size = 10 MB, Compressed = 2 MB
→ Compression ratio = 10 ÷ 2 = **5:1**

Interpretation:
The file is now 5 times smaller than before.

8. Real-world Applications of Data Compression

- **Websites:** Images and videos compressed to load pages faster.
- **Email attachments:** Files zipped to send quickly.
- **Streaming services:** Reduce buffering using lossy compression.
- **Backups:** Use ZIP or TAR to compress archives efficiently.
- **IoT devices:** Compress sensor data for faster transmission.

Example:
When uploading photos to social media, apps like Instagram compress them to reduce upload time and data usage.

9. Ethical and Practical Considerations

- **Data integrity:** For important information (e.g., legal documents), only lossless compression should be used.
- **Perceived quality:** Some users may notice reduced quality in images or audio.
- **Storage vs quality trade-off:** Choosing between small size or high fidelity depends on purpose (e.g., medical imaging vs entertainment).

10. Summary

Aspect	Description	Example
Purpose	Reduce file size and improve efficiency	Faster downloads
Methods	Lossless and Lossy	ZIP vs JPEG
Key Techniques	RLE, Huffman, LZW, JPEG, MP3	–
Compression Ratio	Original ÷ Compressed	5:1
Key Benefit	Saves space and bandwidth	Cloud storage, media streaming

---

Revision Questions

Define data compression.
Distinguish between lossless and lossy compression.
Explain why data compression is important in communication systems.
Describe the process of Run-Length Encoding (RLE) using an example.
Explain how Huffman coding achieves compression.
Outline one limitation of RLE.
Explain how lossy compression reduces the size of an image.
Define the term “compression ratio” and calculate it for a 6 MB file compressed to 1.5 MB.
Compare the suitability of lossless and lossy compression for text documents.
Discuss why MP3 is preferred for streaming compared to WAV.
Identify one situation where lossless compression is essential.
Explain how MPEG video compression reduces file size.
State one advantage and one disadvantage of JPEG compression.
Describe one ethical concern when using lossy compression in medical imaging.
Explain the trade-off between compression efficiency and data quality.