COMPUTER

✓ 2. Number Systems and Data Representation

2.1 Numbering Systems

2.1.1 Decimal System

- **Definition**: A number system that uses 10 digits: 0 to 9.
- **Key Point**: It is also called **Base-10** system.
- **Place Value**: Each digit has a place value that is a power of 10.
- **Example**:
  The number **345** = $(3 \times 10^2) + (4 \times 10^1) + (5 \times 10^0)$
  = 300 + 40 + 5 = **345**

2.1.2 Binary System

- **Definition**: A number system that uses only 2 digits: 0 and 1.
- **Key Point**: It is also called **Base-2** system. Used by computers.
- **Each binary digit = 1 bit**
- **Place Value**: Each digit's place value is a power of 2.
- **Example**:
  Binary **1011** = $(1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0)$
  = 8 + 0 + 2 + 1 = **11 (in decimal)**
- **Conversion Rule (Decimal to Binary)**:
  - Divide the number by 2 repeatedly
  - Write the remainders from **bottom to top**
  - **Example**:
    Decimal 13 → Binary 1101

2.1.3 Octal System

- **Definition**: A number system that uses 8 digits: 0 to 7.
- **Base**: **Base-8**
- **Relation with Binary**: 1 octal digit = 3 binary digits
- **Example**:
  Octal **725** = $(7 \times 8^2) + (2 \times 8^1) + (5 \times 8^0)$ = 448 + 16 + 5 = **469 (decimal)**
- **Conversion Rule (Decimal to Octal)**:
  - Divide by 8 repeatedly and write remainders bottom to top

- **Example**: Decimal 100 → Octal 144

2.1.4 Hexadecimal System

- **Definition**: A number system that uses 16 symbols: 0–9 and A–F (A=10, B=11, … F=15)
- **Base**: **Base-16**
- **Relation with Binary**: 1 hex digit = 4 binary digits
- **Example**:
  Hex **3A** = $(3 \times 16^1) + (10 \times 16^0) = 48 + 10 =$ **58 (decimal)**
- **Conversion Rule (Decimal to Hex)**:
  - Divide the number by 16 repeatedly
  - Write remainders bottom to top
  - **Example**: Decimal 255 → Hex = FF

2.2 Data Representation in Computers

2.2.1 Binary Encoding

- **Definition**: The process of representing data (numbers, text, etc.) using 0s and 1s.

2.2.2 Whole Numbers and Integers

2.2.2.1 Whole Numbers (W)

- **Definition**: Numbers starting from 0 and going up (0, 1, 2, 3...)
- **Key Point**: These are non-negative integers.

2.2.2.2 Integers (Z)

- **Definition**: Numbers that include negative, zero, and positive values
  Example: -3, -2, -1, 0, 1, 2, 3...

2.2.2.3 Two's Complement

- **Definition**: A method to represent negative integers in binary.
- **Steps**:
  1. Invert all bits (change 0 to 1, and 1 to 0)
  2. Add 1 to the result

- **Example**:
  Binary of +5 = 00000101
  Step 1: Invert = 11111010
  Step 2: Add 1 = 11111011 → This is -5 in two's complement

2.3 Storing Real Values (Floating Point Numbers)

2.3.1 Floating-Point Representation

- **Definition**: A way to store decimal (real) numbers in computers using binary.
- **Formula**:
  Sign × Mantissa × 2^Exponent

2.3.1.1 Single Precision (32-bit)

- **Definition**: Uses 32 bits:
  - 1 bit for sign
  - 8 bits for exponent
  - 23 bits for mantissa
- **Range**: Approx. 1.4×10^{-45} to 3.4×10^{38}
- **Used in**: Normal precision calculations

2.3.1.2 Double Precision (64-bit)

- **Definition**: Uses 64 bits:
  - 1 sign bit
  - 11 exponent bits
  - 52 mantissa bits
- **Used in**: Scientific and engineering calculations needing high precision

---

2.4 Binary Arithmetic Operations

2.4.1 Binary Addition

- **Rules**:
  - 0 + 0 = 0
  - 0 + 1 = 1
  - 1 + 1 = 0 (carry 1)
  - 1 + 1 + 1 = 1 (carry 1)

- **Example**:
  1101 + 1011 = 11000 (in binary)

2.4.2 Binary Subtraction

- **Method**: Use **two's complement** to subtract.
- **Example**:
  9 - 5 = Add binary of 9 and two's complement of 5

2.4.3 Binary Multiplication

- **Rule**: Multiply like in decimal. Shift left for each new row.
- **Example**:
  101 × 11 =
  → 101
  → 1010
  → Result = 1111

2.4.4 Binary Division

- **Rule**: Similar to decimal long division
- **Steps**:
  - Subtract
  - Shift right
  - Repeat
- **Example**:
  1010 ÷ 10 = 101 (binary 10 ÷ 2 = 5 decimal)

2.5 Text Encoding Schemes

2.5.1 ASCII

- **Definition**: American Standard Code for Information Interchange
- **Uses 7 bits** to represent characters
- **Range**: 0 to 127
- **Example**:
  - 'A' = 65
  - 'a' = 97
  - '0' = 48

2.5.1.1 Extended ASCII

- **Definition**: Uses 8 bits (256 characters)
- **Supports**: More symbols and foreign characters

2.5.2 Unicode

- **Definition**: A universal character set for almost all languages
- **Supports**: 1 million+ characters
- **Encoding Types**:
  - **UTF-8**: 1 to 4 bytes, compatible with ASCII
  - **UTF-16**: 2 or 4 bytes, not ASCII-compatible
  - **UTF-32**: Fixed 4 bytes per character

2.6 Storing Images, Audio, and Video

2.6.1 Storing Images

- **Definition**: Images are stored as pixels. Each pixel holds color information.
- **Color Encoding**: RGB (Red, Green, Blue)
- **Example**: A 1024×768 image = 786,432 pixels
- **Formats**:
  - JPEG (compressed)
  - PNG (lossless)
  - GIF (animated, limited colors)

2.6.2 Storing Audio

- **Definition**: Sound is converted into digital format using sampling
- **Process**:
  - Sampling: Taking sound wave values at regular intervals
  - Quantizing: Converting sample values to binary
- **Formats**: MP3, WAV, AAC

2.6.3 Storing Video

- **Definition**: Video = Sequence of images (frames) with audio
- **Frame Rate**: 24, 30, or 60 frames per second

- **Formats**: MP4, AVI, MKV

2.6.4 File Storage

- **Definition**: All digital data is stored in binary form using files
- **Storage Devices**:
  - HDD (Hard Disk Drive)
  - SSD (Solid State Drive)
  - Cloud Storage
