UNIT – I

Boolean Algebra and Logic Gates: Binary codes, Binary Storage and Registers, Binary logic. Digital logic gates. Data Representation: Data types, Complements, Fixed Point Representation, Floating Point Representation Digital Computers: Introduction, Block diagram of Digital Computer, Definition of Computer Organization, Computer Design and Computer Architecture.

Introduction to Digital Systems

Digital systems process information using discrete values, primarily binary 0 and 1.

Analog systems represent information as continuously varying physical quantities.

Digital systems offer high reliability, easy storage, programmable processing, and noise tolerance.

Computers are complex digital systems built from logic gates, storage elements, registers, memory, and processing units.

Binary representation forms the foundation for arithmetic, logic, control, and data storage.

Binary Codes

Representing characters and numerical information in digital systems

BCD

Binary Coded Decimal
Each decimal digit is represented by 4 bits.
Example: 59 → 0101 1001

ASCII

Character coding scheme.
Uses binary codes to represent letters, digits, and symbols.

Gray Code

Adjacent values differ by only one bit.
Useful in position encoders and error reduction.

Unicode

Represents characters from many writing systems.
Widely used in modern computing.

Binary codes provide a standard way to represent information inside digital computers.

Binary Storage and Registers

Bit

Smallest unit of digital information.
Value is 0 or 1.

Byte

Group of 8 bits.
Common unit for character and memory storage.

Word

Fixed-size group of bits processed by a computer at a time.

Register

Fast storage inside CPU.

Registers are high-speed storage elements used to hold operands, addresses, instructions, and intermediate results.

Examples: Program Counter (PC), Instruction Register (IR), Memory Address Register (MAR), and Accumulator.

Binary Logic

Binary logic works with two logical states: 0 (FALSE/LOW) and 1 (TRUE/HIGH).

Boolean variables can represent the state of a switch, sensor, control signal, or digital circuit.

Three basic operations are AND, OR, and NOT.

Truth tables list the output for every possible combination of input values.

Boolean expressions provide a mathematical description of digital circuits.

AND

Output = 1 only when ALL inputs are 1.

OR

Output = 1 when ANY input is 1.

NOT

Output is the complement of the input.

Digital Logic Gates

Basic gates and their logical functions

AND

$Y = A \cdot B$
All inputs must be 1

OR

$Y = A + B$
Any input can be 1

NOT

$Y = \overline{A}$
Inverts input

AND Truth Table

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

OR Truth Table

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

NOT Truth Table

A	Y
0	1
1	0

Universal and Exclusive Logic Gates

NAND

$Y = \overline{(A \cdot B)}$
Universal gate.
Can implement other gates.

NOR

$Y = \overline{(A + B)}$
Universal gate.
Useful for logic realization.

XOR

$Y = A \oplus B$
Output is 1 when inputs are different.

XNOR

$Y = \overline{(A \oplus B)}$
Output is 1 when inputs are equal.

NAND and NOR are called universal gates because any Boolean function can be constructed using only NAND gates or only NOR gates.

XOR and XNOR are widely used in adders, parity circuits, comparators, and error-detection logic.

Boolean Algebra

Boolean algebra is a mathematical system used to analyze and simplify digital logic circuits.

Boolean variables take only two values: 0 and 1.

Basic operations: AND (·), OR (+), and NOT (¯ or ').

Boolean expressions can be simplified to reduce the number of gates and circuit complexity.

Example: A + A·B = A (Absorption Law).

AND

$A \cdot 0 = 0$
$A \cdot 1 = A$

OR

$A + 0 = A$
$A + 1 = 1$

Complement

$A \cdot \overline{A} = 0$
$A + \overline{A} = 1$

Boolean Algebra Laws and Theorems

Identity

$A + 0 = A$
$A \cdot 1 = A$

Null

$A + 1 = 1$
$A \cdot 0 = 0$

Idempotent

$A + A = A$
$A \cdot A = A$

Involution

$\overline{(\overline{A})} = A$

De Morgan

$\overline{(A+B)} = \overline{A} \cdot \overline{B}$
$\overline{(A \cdot B)} = \overline{A} + \overline{B}$

Absorption

$A + AB = A$
$A(A+B) = A$

Data Representation

Data representation defines how information is encoded and s

Common categories include numeric data, character data, lo ata.

Numeric data can be represented as integers or real numbers

Signed representations are required to represent both positive and negative numbers.

The chosen representation affects storage size, range, precisi

Numeric

Integers
Real / fractional values

Character

ASCII
Unicode

Logical

TRUE / FALSE
1 / 0

Complements

1's Complement

Invert every bit.
0 → 1 and 1 → 0.
Example: 101001 → 010110

2's Complement

Take 1's complement and add 1.
Example: 101001 → 010110 + 1 = 010111

9's Complement

For decimal digits, subtract each digit from 9.

10's Complement

9's complement + 1.

2's complement is the most common method for signed integer representation.

It simplifies subtraction because subtraction can be performed as addition of a number's 2's complement.

It provides a single representation for zero.

Fixed-Point Representation

Fixed-point representation stores the binary point at a fixed position.

It is suitable for integers and fractional values when a known precision is sufficient.

Signed integers may use sign-magnitude, 1's complement, or 2's complement representation.

2's complement is widely used because arithmetic operations are efficient.

Example: In an 8-bit unsigned representation, values range from 0 to 255.

Sign-Magnitude

MSB indicates sign.
Remaining bits represent magnitude.

1's Complement

Negative value is obtained by complementing every bit.

2's Complement

1's complement + 1.
Most common signed format.

Floating-Point Representation

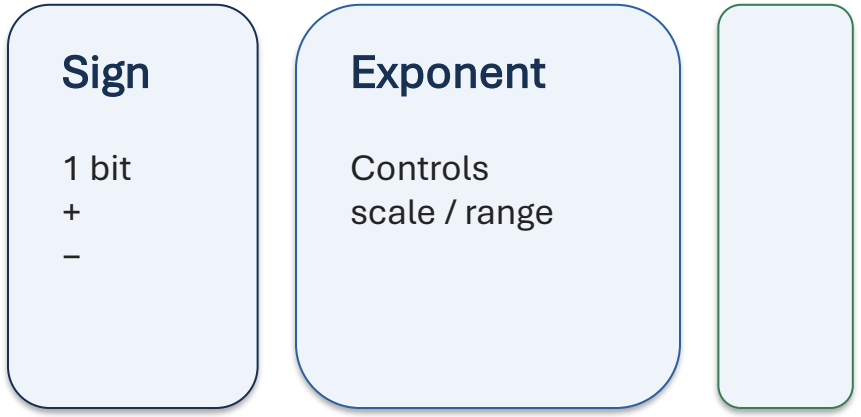
Floating-point representation is used for very large, very small, or fractional numbers.

A floating-point number is generally represented using sign, exponent, and significand (mantissa/fraction).

Normalization places the significand in a standard form.

Floating-point representation provides a large dynamic range but has finite precision.

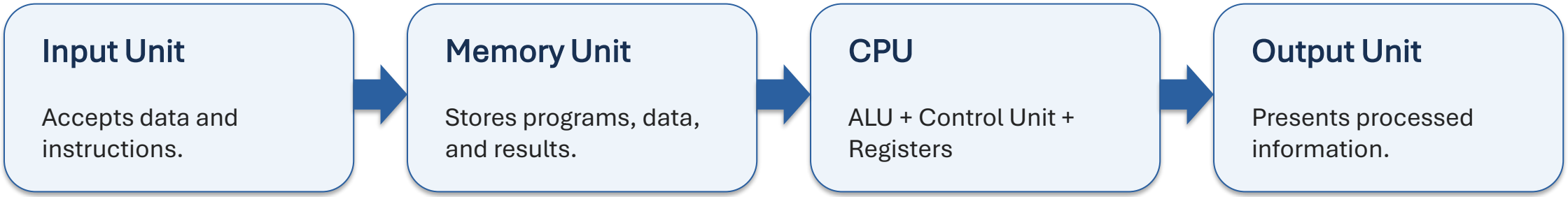
IEEE 754 is a widely used standard for floating-point representation.



General Form

Value ≈ (−1)^sign × significand × base^exponent

Digital Computer – Introduction & Block Diagram



Computer Organization, Design and Architecture

Computer Architecture

Programmer-visible attributes.

- Instruction set
- Data types
- Addressing modes
- Memory addressing
- I/O mechanisms

Computer Organization

How components are interconnected and operate.

- Control signals
- Memory technology
- Buses
- CPU implementation
- I/O interfaces

Computer Design

Detailed hardware implementation.

- Circuit design
- Processor datapath
- Control logic
- Memory circuits
- Physical realization

Key idea: Architecture describes what the computer does; organization describes how it is implemented; design turns the organization into actual hardware.

Unit summary: Binary codes → Logic gates → Boolean algebra → Data representation → Digital computer fundamentals.