
Tikrit University
Electrical Engineering Department

# EE-317
# Computer Engineering
# 2024-2025

# Arithmetic: Floating Point Representation

**Jalal Nazar Abdulbaqi, Ph.D.**

*jalal.abdulbaqi@tu.edu.iq*

# Outline

- Arithmetic on Floating Point
  - Floating Point Representation
  - IEEE Floating-Point Format Standard

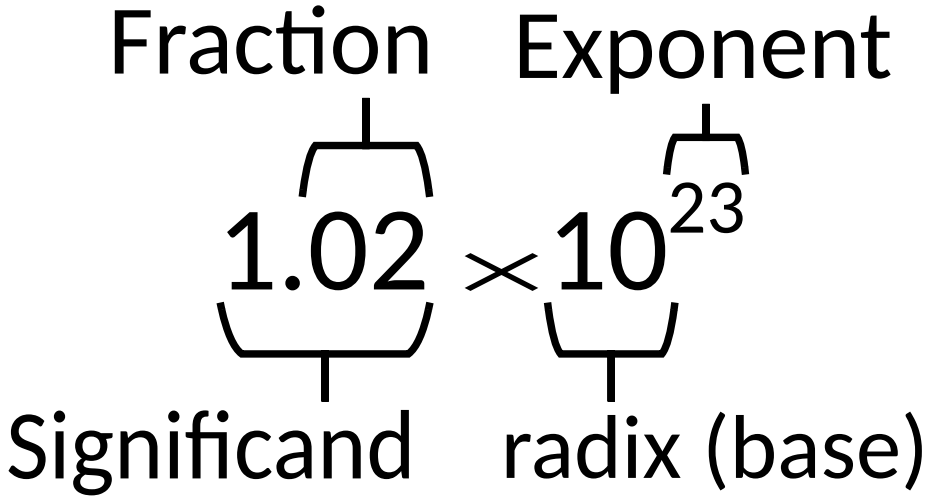
# Floating Point

- Representation for non-integral numbers
  - Including very small and very large numbers
- What can be represented in N bits?
  - Unsigned **0** to **$2^N$**
  - $2^s$ Complement **$-2^{N-1}$** to **$2^{N-1} - 1$**
- With RISC-V singed integers the 32-bit architecture allows us to represent numbers from (**$-2^{31}$**) to (**$+2^{31} -1$**)
  (**-2,147,483,648** to **+2,147,483,647**)

# Floating Point

- But, what about **very large** numbers?
  9,349,398,989,787,762,244,859,087,678
  $9.34939899 \times 10^{27}$ (Scientific Notation)
- What about **very small** numbers?
  0.000000000000000000045691
  $4.5691 \times 10^{-24}$
- Floating point representation allows much larger range at the expense of accuracy

# Scientific Notation




$-2.34 \times 10^{56}$ ← Normalized

$+0.002 \times 10^{-4}$ ← Not normalized

$+987.02 \times 10^{9}$ ← Not normalized

# Scientific Notation

- Decimal notation

$$-1.673 \times 10^{24}$$ Sign, magnitude

  - Exponent may be negative

- Binary notation (binary *floating* point)

$$\pm 1.\text{xxxxxxx}_{\text{two}} \times 2^{\text{yyyy}}$$

yyyy ⟵ Exponent (E)

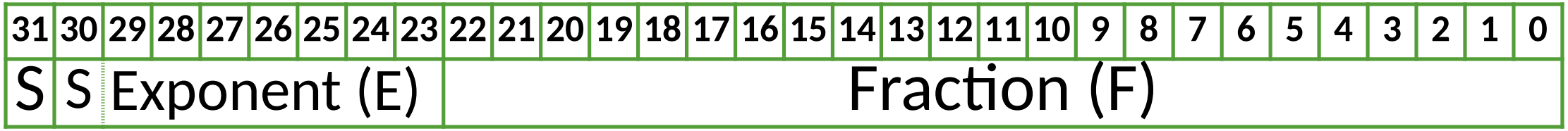
Number of **y**s determines range

Fraction (F)

Number of **x**s determines accuracy

# Floating Point Representation

- Single Precision



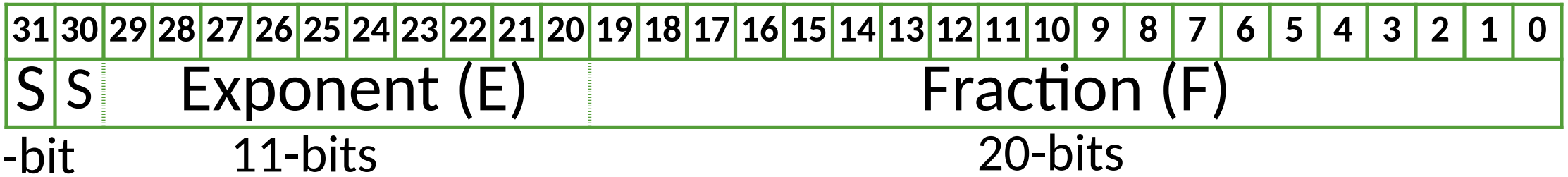

1-bit 8-bits 23-bits

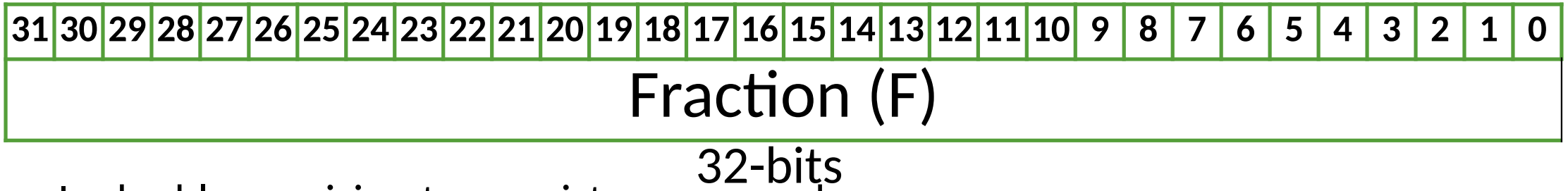
- Smallest fraction $\mathbf{2.0_{ten} x 10^{-38}}$ and Largest number $\mathbf{2.0_{ten} x 10^{38}}$

- When exponent is **too large** – or **too small** – an exception **Overflow**, or **underflow**. It is written into the floating point control and status register **fcsr** (no interrupt).

# Floating Point Representation

- Double Precision




- In double-precision two registers are used.
- The range is from smallest fraction $\mathbf{2.0_{ten}\ x10^{-308}}$ to largest number $\mathbf{2.0_{ten}x10^{308}}$
- Fraction now has 52 bits.

# IEEE Floating-Point Format Standard

- Defined by IEEE Std 754 Since 1985
- Developed in response to divergence of representations
- Now almost universally adopted
- Two representations
  - Single precision (32-bit)
  - Double precision (64-bit)

# IEEE Floating-Point Format Standard

$$x = (-1)^{S} \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} - \text{Bias})}$$

- Bias
  - Single precision (32-bit) = **127**
  - Double precision (64-bit) = **1023**
- $127_{ten} = 111\ 1111_{two}$ (7-bit)
- $1023_{ten} = 11\ 1111\ 1111_{two}$ (10-bit)

# Single-Precision Range

- Exponents 00000000 and 11111111 reserved
- Smallest value
  - **Exponent**: 00000001 ➡ actual exponent = 1 – 127 = –126
  - **Fraction**: 000…00 ➡ significand = $1.0_{ten}$

$$\pm 1.0_{ten} \times 2^{-126} \approx \pm 1.2_{ten} \times 10^{-38}$$

- Largest value
  - **Exponent**: 11111110 ➡ actual exponent = 254 – 127 = +127
  - **Fraction**: 111…11 ➡ significand ≈ $2.0_{ten}$

$$\pm 2.0_{ten} \times 2^{+127} \approx \pm 3.4_{ten} \times 10^{+38}$$

# Double-Precision Range

- Exponents 0000…00 and 1111…11 reserved
- Smallest value
  - Exponent: 00000000001 ➡ actual exponent = 1 – 1023 = –1022
  - Fraction: 000…00 ➡ significand = $1.0_{ten}$

$$\pm 1.0_{ten} \times 2^{-1022} \approx \pm 2.2_{ten} \times 10^{-308}$$

- Largest value
  - Exponent: 11111111110 ➡ actual exponent = 2046 – 1023 = +1023
  - Fraction: 111…11 ➡ significand ≈ $2.0_{ten}$

$$\pm 2.0_{ten} \times 2^{+1023} \approx \pm 1.8_{ten} \times 10^{+308}$$

# Floating-Point Example (Decimal → Binary)

- Represent −0.75

1. $-0.75_{ten} = -3_{ten} / 4_{ten} = -3_{ten} / 2^2_{ten} = -11_{two} / 2^2_{ten}$

$$= -0.11_{two} = \mathbf{(-1)^1 \times 1.1_{two} \times 2^{-1}}$$

2. $0.75 \times 2 = \mathbf{1} + 0.5$ (MSB)

$0.5 \times 2 = \mathbf{1} + 0$ (LSB)

$-0.75_{ten} = -0.11_{two} \rightarrow -0.11_{two} = \mathbf{(-1)^1 \times 1.1_{two} \times 2^{-1}}$

# Floating-Point Example (Decimal → Binary)

- $-0.75 = (-1)^1 \times 1.1_{two} \times 2^{-1}$ : $(-1)^S \times 1.F_{two} \times 2^{E-Bias}$
  - Sign (S) = 1
  - Fraction (F) = $1000…00_{two}$
  - Exponent (E) = –1 + Bias
    - Single: –1 + 127 = 126 = $01111110_{two}$
    - Double: –1 + 1023 = 1022 = $01111111110_{two}$
- Single: 10111111001000…00
- Double: 10111111111101000…00

# Floating-Point Example (Binary → Decimal)

- What number is represented by the single-precision float

  1100000010100000000000000

  - Sign (S) = 1
  - Fraction (F) = $01000\ldots00_{two} = 0.01 = 1 \times 2^{-2} = 1 / 4 = 0.25_{ten}$
  - Exponent (E) = $10000001_{two} = 2^7 + 2^0 = 128+1 = 129_{ten}$

$$X = (-1)^1 \times (1 + 0.01_2) \times 2^{(129 - 127)} = -1 \times 1.25 \times 2^2 = -5.0_{ten}$$

# Convert Real Numbers (Binary → Decimal)

$101.1011_{two}$ → integer . fraction

- Integer: $101_{two} = 2^2 \times 1 + 2^1 \times 0 + 2^0 \times 1 = 4+0+1 = 5$
- Fraction: $0.1011_{two} = 2^{-1} \times 1 + 2^{-2} \times 0 + 2^{-3} \times 1 + 2^{-4} \times 1$

$= 1/2 + 0 + 1/8 + 1/16$

$= 0.5 + 0 + 0.125 + 0.0625 = 0.6875$

$101.1011_{two} = 5.6875_{ten}$

# Convert Real Numbers (Decimal → Binary)

$5.6875_{ten}$ → integer . fraction

- Integer: 5/2 = 1 reminder **1** (LSB)
  2/2 = 1 reminder **0**
  1/2 = 0 reminder **1** (MSB)

$5 = 101_{two}$

- Fraction: 0.6875×2 = **1** + 0.375 (MSB)
  0.375×2 = **0** + 0.75
  0.75×2 = **1** + 0.5
  0.5×2 = **1** +0 (LSB)

$0.6875_{ten} = 0.1011_{two}$

# Summary

- Floating point representation is used for **non-integral**, **large** and **small numbers**.
- There are two types of Floating point representation: **Single Precision** (32-bit) and **double Precision** (64-bit)
- **IEEE 754** is a standard that used by all modern architectures to represent the floating point numbers