# COMPUTER ORGANIZATION & DESIGN

SE – CIS  Batch 2018
Spring 2020

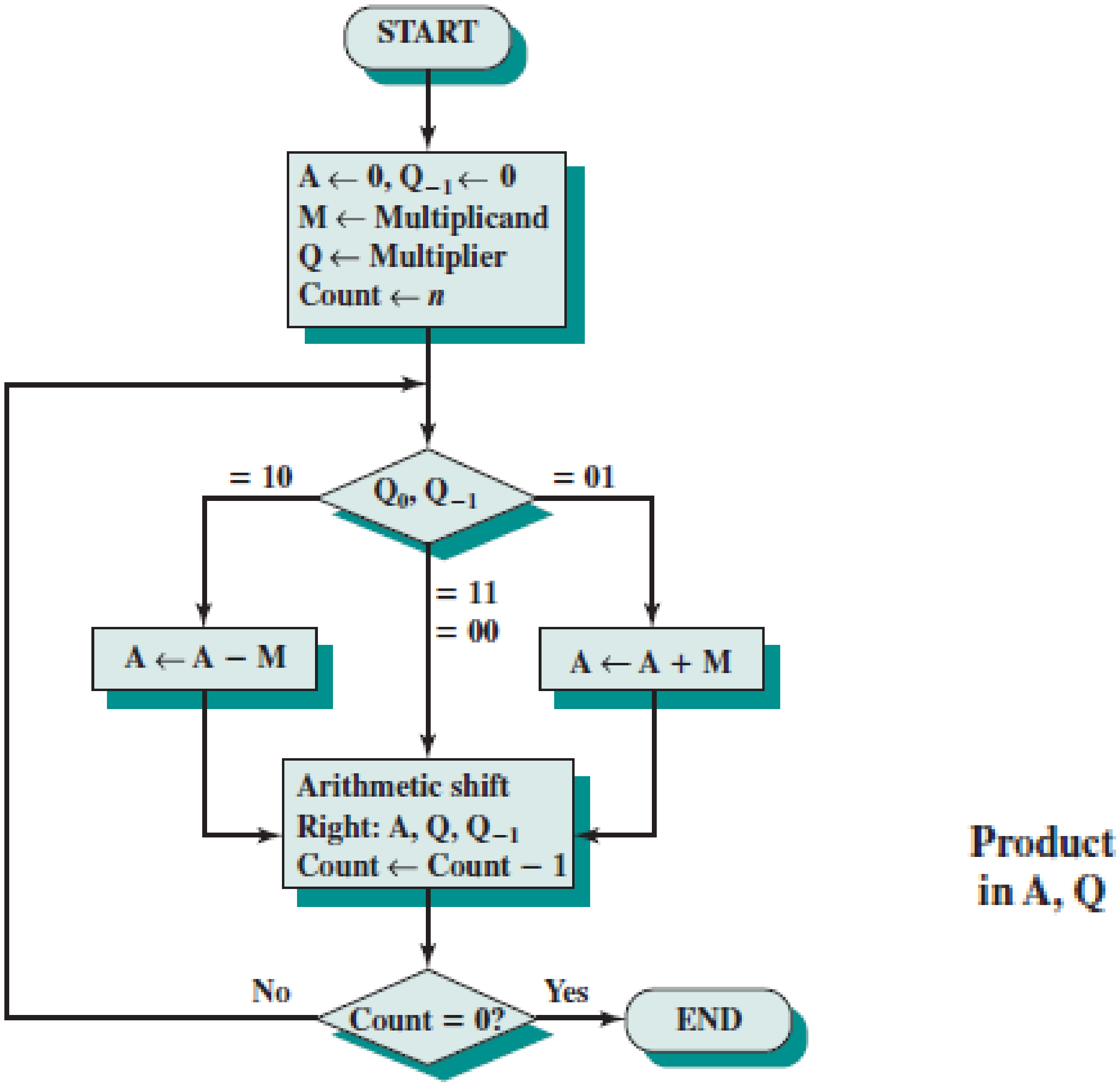
Instructor : **Anita Ali**

# Lecture Plan

- Signed Integer Multiplication
- Real Number Representation
- IEEE 754 Binary 32 Format

    (Single Precision Format)

# Signed Multiplication – Booth's Algorithm



# Signed Multiplication – Booth's Algorithm

## Notion Used

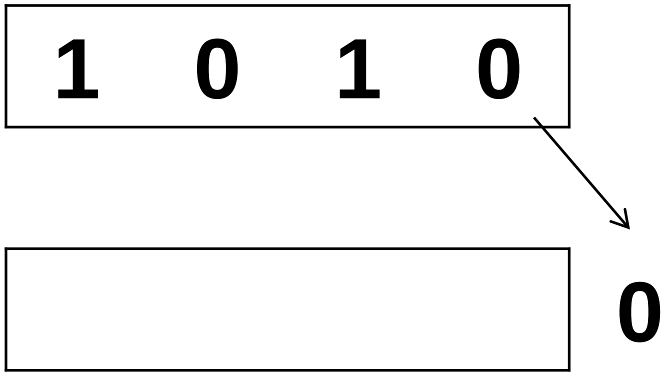
| | | |
|---|---|---|
| A, M & Q | : | n-bit registers |
| $Q_{-1}$ | : | an extra bit at right hand side of bit 0 of register Q |

# Signed Multiplication – Booth’s Algorithm

**Arithmetic Shift Right**

Shift all bits *one* position towards right and retain sign bit (MSB)

**Example**

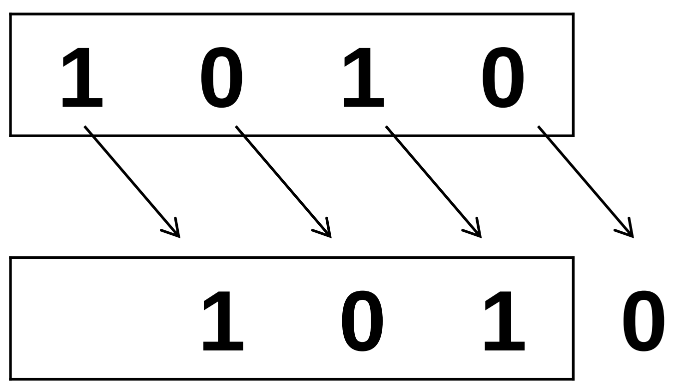


# Signed Multiplication – Booth's Algorithm

## Arithmetic Shift Right

Shift all bits *one* position towards right and retain sign bit (MSB)

## Example

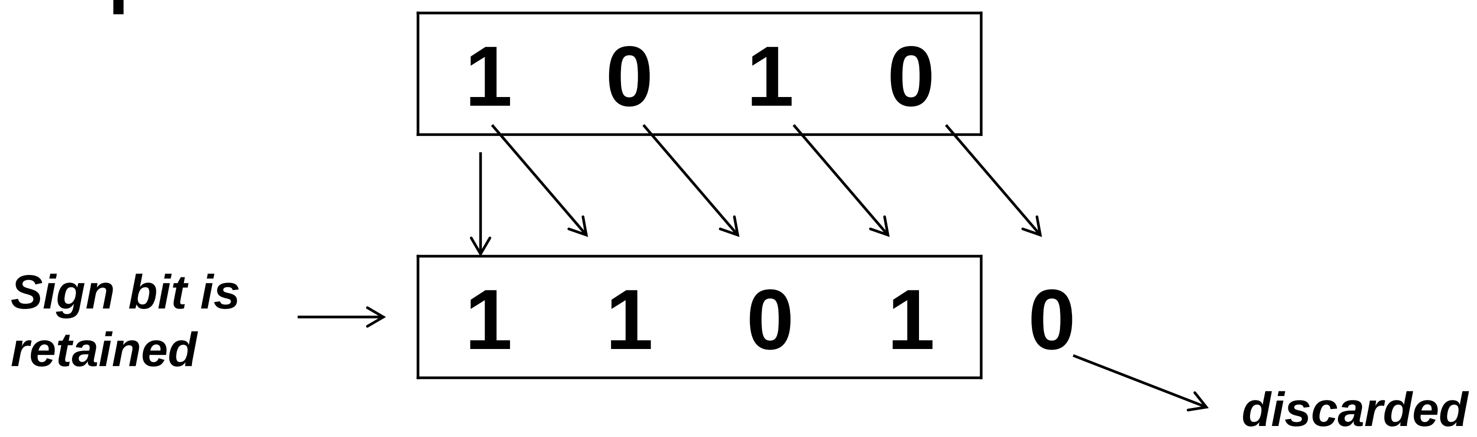


# Signed Multiplication – Booth's Algorithm

**Arithmetic Shift Right (ASR)**

Shift all bits *one* position towards right and retain sign bit (MSB)

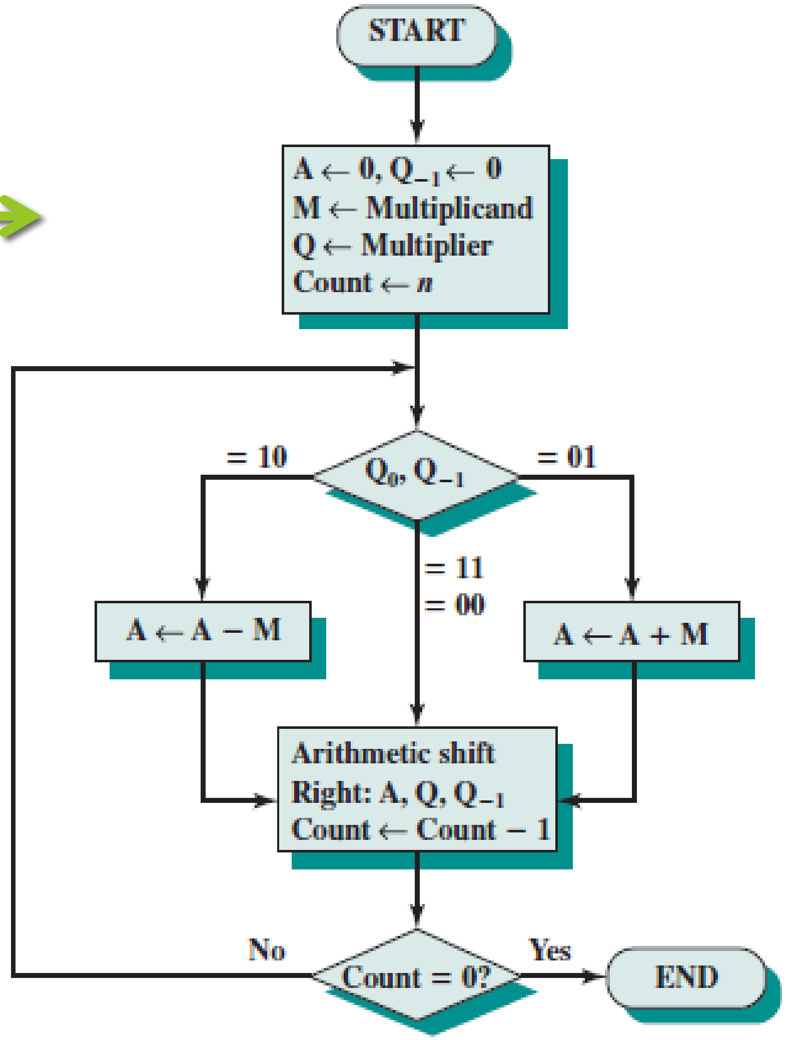
**Example**



# Example : - 7 x - 8 (4-bits)

| A | Q | Q-1 | Count | Remarks |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

# Signed Multiplication – Booth's Algorithm




Product in A, Q

# Example : - 7 x - 8 (4-bits)

**M = 1001**

| A | Q | Q-1 | Count | Remarks |
|---|---|---|---|---|
| 0000 | 1000 | 0 | 4 | Initialization |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

# Example : - 7 x - 8 (4-bits)

**M = 1001**

| A | Q | Q-1 | Count | Remarks |
|---|---|---|---|---|
| 0000 | 1000 | 0 | 4 | Initialization |
| 0000 | 0100 | 0 | 3 | $Q_0$ $Q_{-1}$ = 00 ASR A, Q, Q-1 , Count -1 |
| | | | 3 | Count ≠ 0 ***so*** next cycle |
| 0000 | 0010 | 0 | 3 | $Q_0$ $Q_{-1}$ = 00 ASR A, Q, Q-1 , Count -1 |
| | | | 2 | Count ≠ 0 ***so*** next cycle |
| 0000 | 0001 | 0 | 1 | $Q_0$ $Q_{-1}$ = 00 ASR A, Q, Q-1 , Count -1 |
| | | | | Count ≠ 0 ***so*** next cycle |
| 0111 | 0001 | 0 | 1 | $Q_0$ $Q_{-1}$ = 10 A ← A – M ( or A + M' ) |
| 0011 | 1000 | 1 | 0 | ASR A, Q, Q-1 , Count -1 |
| | | | | Count = 0 ***so*** end |

# Example : - 7 x - 8 (4-bits)

**M = 1001**

| A | Q | Q-1 | Count | Remarks |
|---|---|---|---|---|
| 0000 | 1000 | 0 | 4 | Initialization |
| 0000 | 0100 | 0 | 3 | $Q_0$ $Q_{-1}$ = 00 ASR A, Q, Q-1 , Count -1 |
| | | | 3 | Count ≠ 0 ***so*** next cycle |
| 0000 | 0010 | 0 | 3 | $Q_0$ $Q_{-1}$ = 00 ASR A, Q, Q-1 , Count -1 |
| | | | 2 | Count ≠ 0 ***so*** next cycle |
| 0000 | 0001 | 0 | 1 | $Q_0$ $Q_{-1}$ = 00 ASR A, Q, Q-1 , Count -1 |
| | | | | Count ≠ 0 ***so*** next cycle |
| 0111 | 0001 | 0 | 1 | $Q_0$ $Q_{-1}$ = 10 A ← A – M ( or A + M' ) |
| 0011 | 1000 | 1 | 0 | ASR A, Q, Q-1 , Count -1 |
| **0011** | **1000** | | | Count = 0 ***so*** end |

# Example : - 7 x - 8 (4-bits)

**M = 1001**

- Product is

$$\mathbf{0011\ 1000 = 56)_{10}}$$

# Homework 3.1

1. Use Booth's Algorithm to multiply following decimal numbers: (Use only as many bits as required for 2's complement representation of operands)

a) (+4) x (-5)

b) (-9) x (+3)

# Real Number Representation

Digital representation of Real numbers can be of two forms

1. Fixed Point Representation

2. Floating Point Representation

# 1. Fixed Point Representation

- Position of radix point is fixed
- e.g. **(6, 2) fixed point format**
    - It uses total 6 bits
    - 2 bits reserved for fraction part
    - Scheme is simple & fast

# (6, 2) Fixed Point Format

- Frequently used in DSP & Image processing applications
- Which require stringent performance

**Downside**

- Suffers from low precision and range

# 2. Floating Point (FP) Representation

- Also known as scientific notation
- Numbers are represented in the form

$$\text{mantissa} \times \text{base}^{\text{exponent}}$$

- ***Large integers*** can be represented as real numbers
- Too ***small numbers*** can be expressed
- *Superior precision* than fixed point representation

# Pre- Floating Point Era

- Early processors did not directly support floating point in hardware
  - e.g. Intel processors before 80486 microprocessor

# Pre- Floating Point Era

- Each floating point operation was compiled into a ***sequence of integer instructions***
  - Time consuming
- Computer used for graphics / engineering applications often had ***math co-processor***
- It supported floating point operation in ***hardware***

# Floating P Compatibility Issue

- Till 1985, different machines had different floating point representation schemes
- Data stored in one machine was difficult to be interpreted by other machine, rather impossible

# Problems with FP Representation

Same floating point number can be represented in a number of ways

$$111.101 \times 2^{5}$$

$$= 111101 \times 2^{2}$$

$$= 1.11101 \times 2^{7}$$

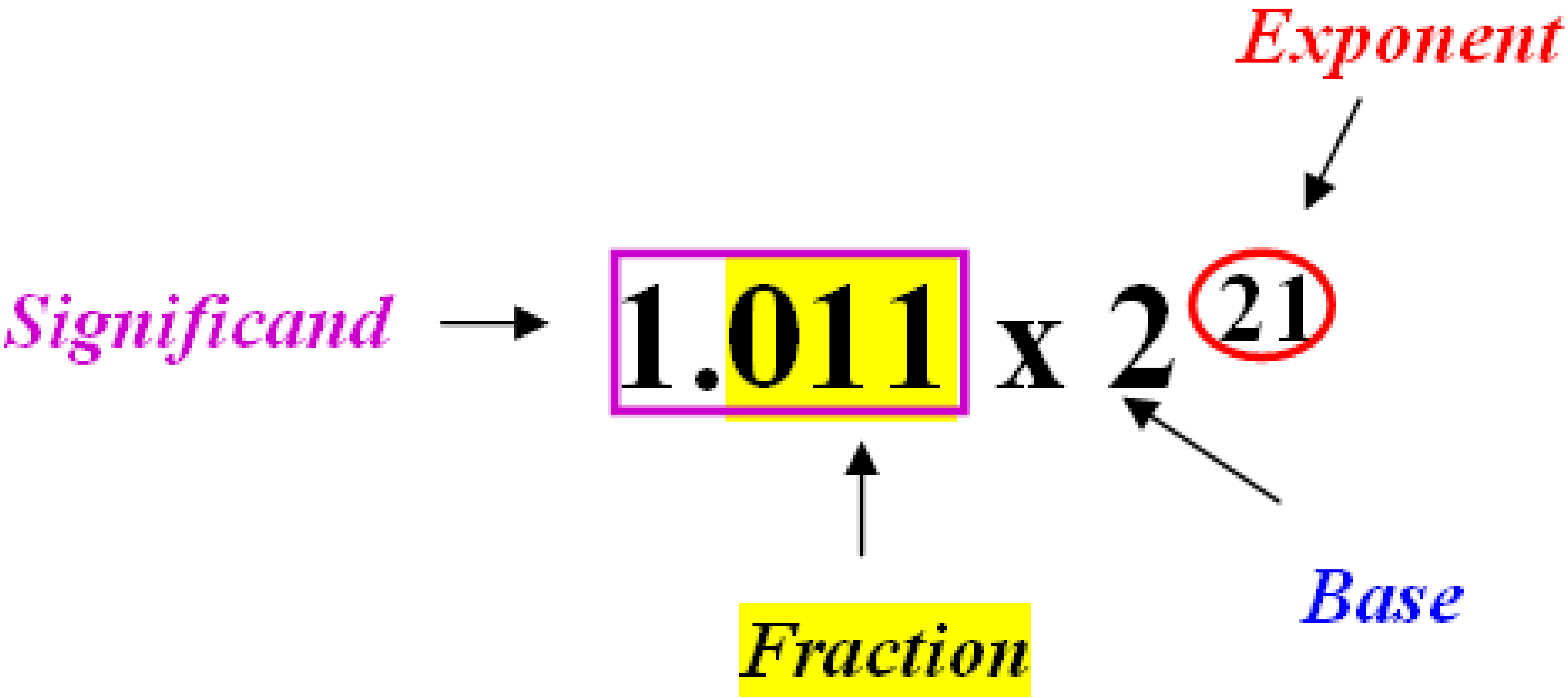
# IEEE 754 Standard

- IEEE developed a standard for floating point representation in computers
- First released in 1985
- Last revised in 2019

# IEEE 754 Standard – Binary Representation

- Binary 32 Format (Single Precision Format)
- Binary 64 Format (Double Precision Format)
- Binary 128 Format

# Binary Representation

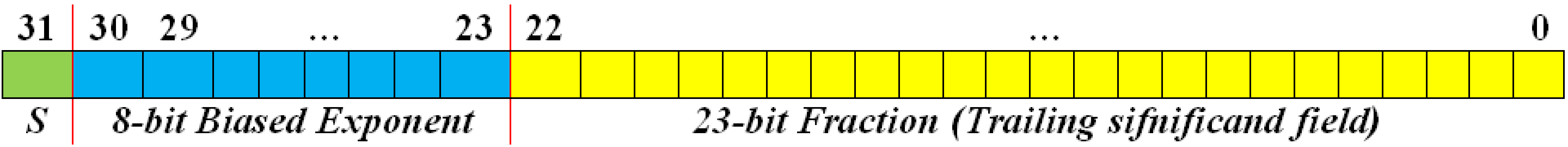


# Binary 32 Format (Single Precision Format)

- Numbers in normalized, scientific binary notation
- i.e. number of the form **1.bbb**
- Where b→ any binary digit (1 / 0)
- e.g. **1.001010** → Normalized

  **101.1001** → Not normalized

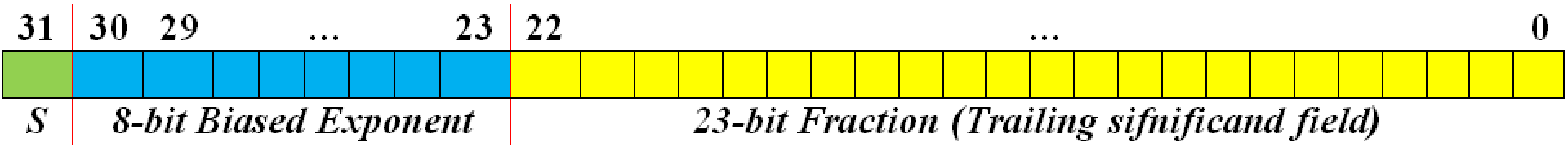
# Binary 32 Format (Single Precision Format)




## S (Sign bit)

- Bit **31** is used to represent sign of the number
- **0** for positive numbers
- **1** for negative numbers

# Binary 32 Format (Single Precision Format)

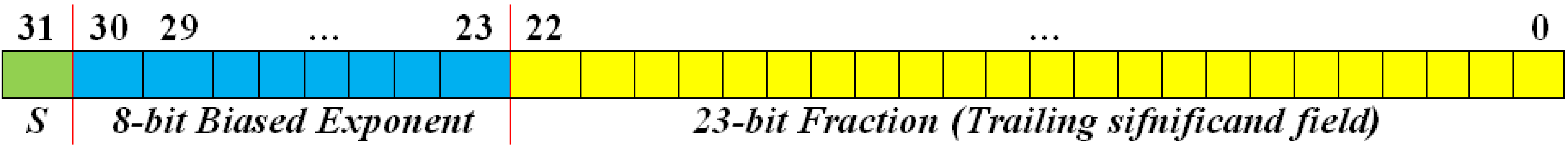



## Biased Exponent Field

- Bit **23** to bit **30** (***8 bits***)
- Bias is **127** → exponent is ***Excess-127*** encoded

**Biased exponent = True exponent + 127**

# Binary 32 Format (Single Precision Format)




## Fraction Field

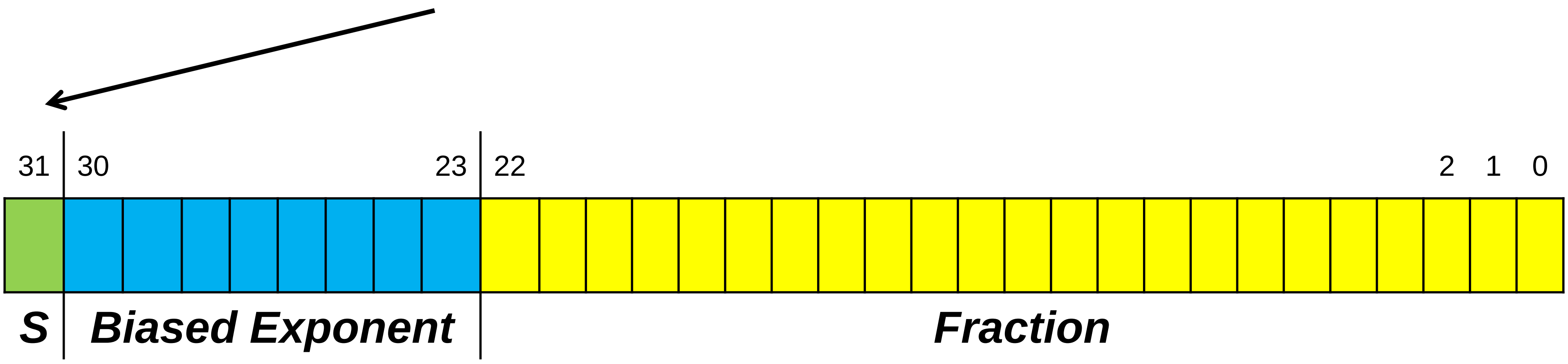
- Bit **0** to bit **22** (***23 bits***)
- Holds only fraction part of the significand

# Example

Represent **101.1 x 2 $^{19}$** in IEEE 754 Binary 32 Format

**Sign Bit**

- Sign of number is positive
- so sign bit (bit 31) is **0**

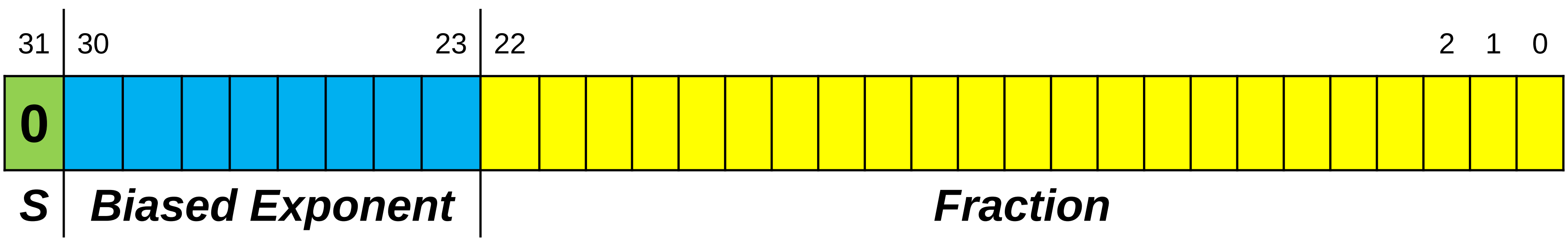


# Example

Represent **101.1 x 2 $^{19}$** in IEEE 754 Binary 32 Format

**<u>Sign Bit</u>**

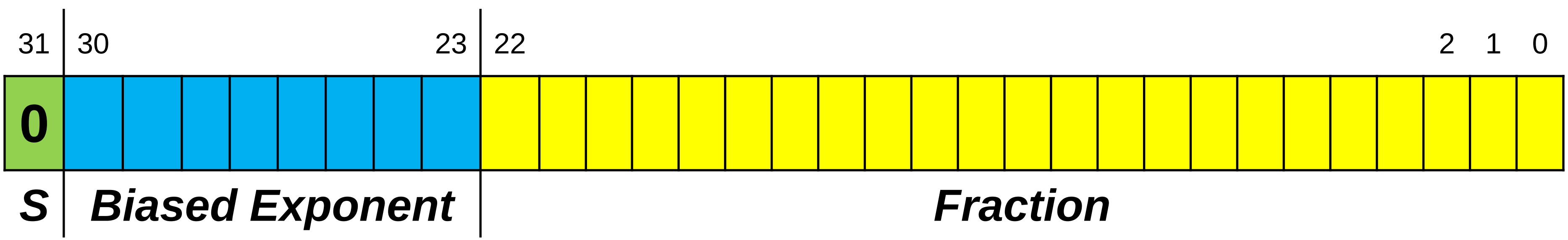
- Sign of number is positive
- so sign bit (bit 31) is **0**



# Example – 101.1 x 2 $^{19}$

## Normalizing Significand

$$1\ 0\ 1\ .\ 1 \times 2^{19}$$

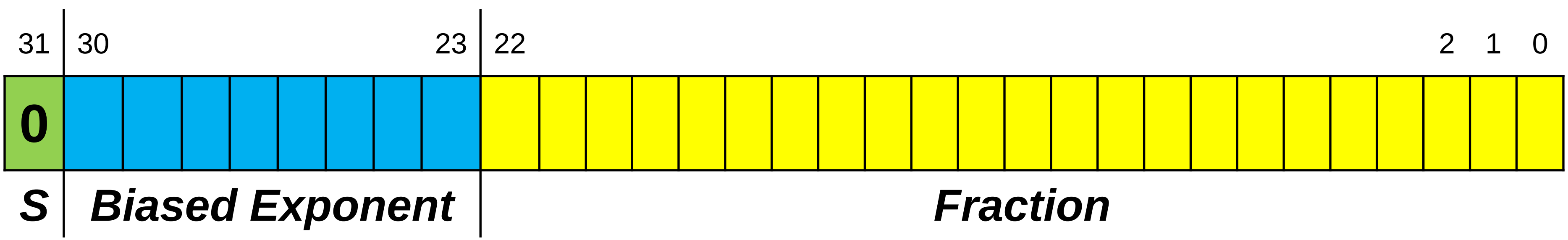


# Example – 101.1 x 2 $^{19}$

## Normalizing Significand

$$1\,0\,1\,.\,1 \times 2^{19}$$

$$= 1\,.\,0\,1\,1 \times 2^{19+2}$$



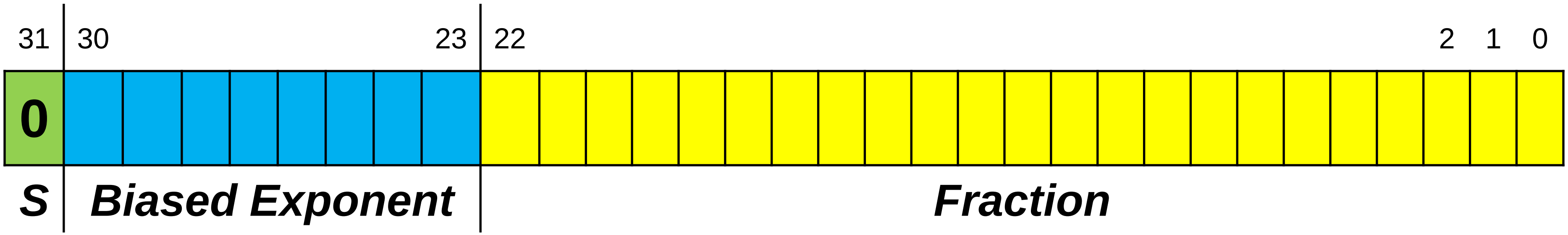
# Example – 101.1 x 2 $^{19}$

**Normalizing Significand**

$$1\ 0\ 1\ .\ 1 \times 2^{19}$$

$$= 1\ .\ 0\ 1\ 1 \times 2^{19+2} = 1\ .\ 0\ 1\ 1 \times 2^{21}$$

(Significand is normalized & exponent is adjusted accordingly)



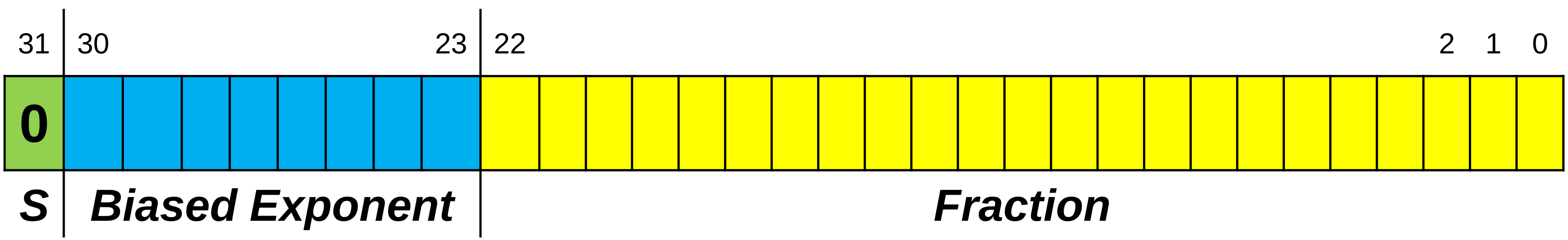
# Example – $101.1 \times 2^{19}$

## Normalizing Significand

$$1\,0\,1\,.\,1 \times 2^{19}$$

$$= 1\,.\,0\,1\,1 \times 2^{19+2} = 1\,.\,0\,1\,1 \times 2^{21}$$

Fraction : **0 1 1**



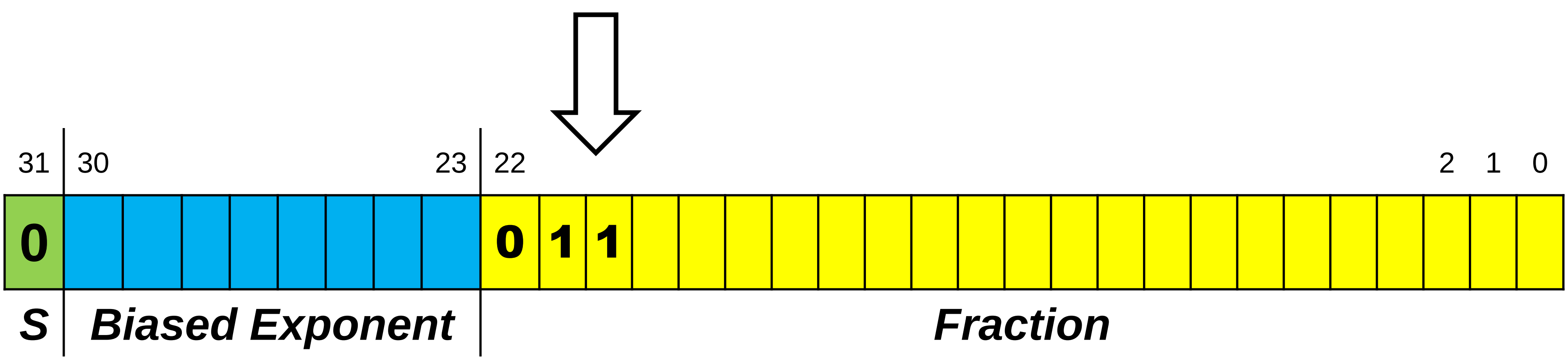
# Example – $101.1 \times 2^{19}$

## Normalizing Significand

$$101.1 \times 2^{19}$$

$$= 1.011 \times 2^{19+2} = 1.011 \times 2^{21}$$

Fraction : **0 1 1**



# Example – $101.1 \times 2^{19}$

## Normalizing Significand

$$1\,0\,1\,.\,1 \times 2^{19}$$

$$= 1\,.\,0\,1\,1 \times 2^{19+2} = 1\,.\,0\,1\,1 \times 2^{21}$$

Fraction : **0 1 1**

*remaining bits are filled up with 0*

| 31 | 30 … 23 | 22 … 2 1 0 |
|---|---|---|
| 0 | | 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| ***S*** | ***Biased Exponent*** | ***Fraction*** |

# Example – 101.1 x 2 $^{19}$

**Biased Exponent**

No : **1 . 0 1 1 x 2 $^{21}$**

Biased exponent = True exponent + **127**

| 31 | 30 … 23 | 22 … 2 1 0 |
| --- | --- | --- |
| 0 | | 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| *S* | *Biased Exponent* | *Fraction* |

# Example – 101.1 x $2^{19}$

## Biased Exponent

No : **1 . 0 1 1 x $2^{21}$**

Biased exponent = True exponent + **127**

= **21 + 127** = **148**

| 31 | 30 … 23 | 22 … 2 1 0 |
|---|---|---|
| 0 | | 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| *S* | ***Biased Exponent*** | ***Fraction*** |

# Example – $101.1 \times 2^{19}$

## Biased Exponent

No : $1.011 \times 2^{21}$

Biased exponent = True exponent + **127**

= **21 + 127** = **148**

= **10010100 )2**

| 31 | 30 … 23 | 22 … 2 1 0 |
|---|---|---|
| 0 | | 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| ***S*** | ***Biased Exponent*** | ***Fraction*** |

# Example – 101.1 x $2^{19}$

## Biased Exponent

No : **1 . 0 1 1 x $2^{21}$**

Biased exponent = True exponent + **127**

= **21 + 127** = **148**

= **10010100 )2**

| 31 | 30 … 23 | 22 … 2 1 0 |
|---|---|---|
| 0 | 1 0 0 1 0 1 0 0 | 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| *S* | *Biased Exponent* | *Fraction* |

# Example – 101.1 x 2 $^{19}$

**101.1 x 2 $^{19}$**

↓ *IEEE 754 Binary 32 Format*

| 31 | 30 23 | 22 2 1 0 |
|---|---|---|
| 0 | 1 0 0 1 0 1 0 0 | 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| *S* | *Biased Exponent* | *Fraction* |

# Example – $101.1 \times 2^{19}$

$$101.1 \times 2^{19}$$

*IEEE 754 Binary 32 Format*

| 31 | 30 … 23 | 22 … 2 1 0 |
|---|---|---|
| 0 | 1 0 0 1 0 1 0 0 | 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| *S* | *Biased Exponent* | *Fraction* |

*In hexadecimal form*

**0x 4A300000**

# IEEE 754 - Binary 32 Format

A number in IEEE 754- Binary 32 Format has the following decimal value

$$(-1)^{S}\,(1+f) \times 2^{(Exponent\,-\,127)}$$

# Class Activity

Represent given number using IEEE 754 Binary 32 Format (Single Precision Format)

$$111001.1001 \times 2^{-60}$$

# Class Activity - Answer

Represent given number using IEEE 754 Binary 32 Format (Single Precision Format)

$$-111001.1001 \times 2^{-60}$$

| 31 | 30 | | | | | | | 23 | 22 | | | | | | | | | | | | | | | | | | | | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| *S* | ***Biased Exponent*** | | | | | | | | ***Fraction*** | | | | | | | | | | | | | | | | | | | | | | |

# Advantages of Normalization

1. Prevents multiple representation of same floating point number

$$111.101 \quad \times \quad 2^5$$

$$= \quad 111101 \quad \times \quad 2^2$$

$$= \quad 1.11101 \quad \times \quad 2^7$$

*Normalized Form*

# Advantages of Normalization

1. Prevents multiple representation of same floating point number

$$111.101 \times 2^5$$

$$= 111101 \times 2^2$$

$$= 1.11101 \times 2^7$$

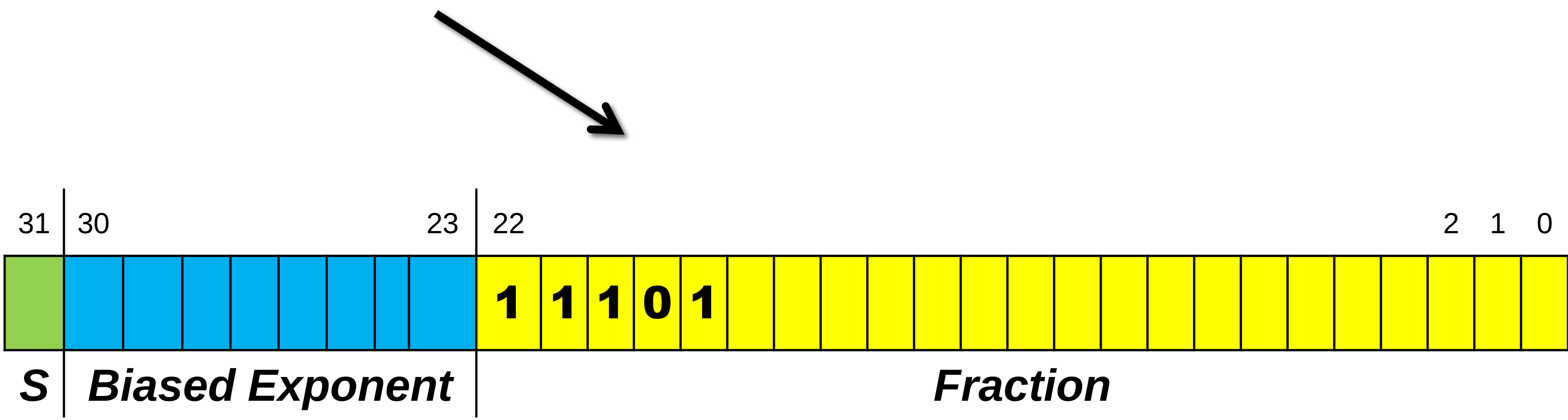
***Normalized Form***

# Advantages of Normalization

2. Avoids storing 1 to the left of binary point

(also known as ***hidden*** 1 / ***implicit*** 1)

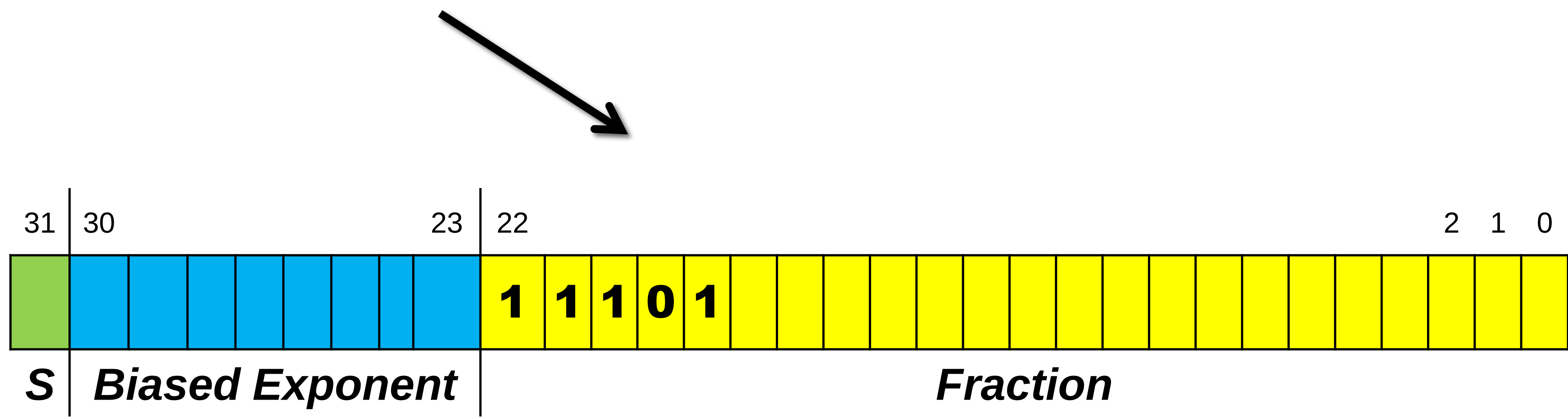
$$1.11101 \times 2^7$$



# Advantages of Normalization

3. Avoids representing the position of the radix(binary) point

$$1.11101 \times 2^7$$



# Why Biased Exponent?

- If instead of biasing 2'c complement method is used to store exponent then sorting of the numbers becomes difficult
- Because  number having negative exponent ***looks*** larger than a number with positive exponent

# 2's Complement Representation

| Number | Representation | Number | Representation |
|---|---|---|---|
| 0 | 0000 | -1 | 1111 |
| 1 | 0001 | -2 | 1110 |
| 2 | 0010 | -3 | 1101 |
| 3 | 0011 | -4 | 1100 |
| 4 | 0100 | -5 | 1011 |
| 5 | 0101 | -6 | 1010 |
| 6 | 0110 | -7 | 1001 |
| 7 | 0111 | -8 | 1000 |

# Why Biased Exponent?

- Biasing resolves this issue
- It is an unsigned notation
- After biasing, data appears like an ordinary integer number
- Where negative numbers appear as a smaller number than a positive number
- Hence comparison /sorting becomes easier

# Comparison/Sorting of FP Numbers

- After biasing comparison/sorting can proceed in 3 steps

1. Compare sign bits → **S**= 0 is greater

2. Compare **E**, if sign are same → big **E** wins

3. Compare **f**, if exponents are same → big **f** wins

# Homework 3.3

Homework will be shared via **Goggle Classroom**

# Recap

- Signed Integer Multiplication
- Real Number Representation
- IEEE 754 Binary 32 Format

  (Single Precision Format) (to be contd....)

# Reading Assignment

- Go through all relevant sections from
- Chapter 3 of text book

  **Computer Organization & Design** (5th edition)

- Chapter 10 of reference book

  **Computer Organization & Architecture** (10th ed)

**Note:** *Be prepared for* ***quiz*** *in upcoming live session*

# Stay Home
# Stay Safe