

Floating Point I

CSE 351 Autumn 2020

Instructor:

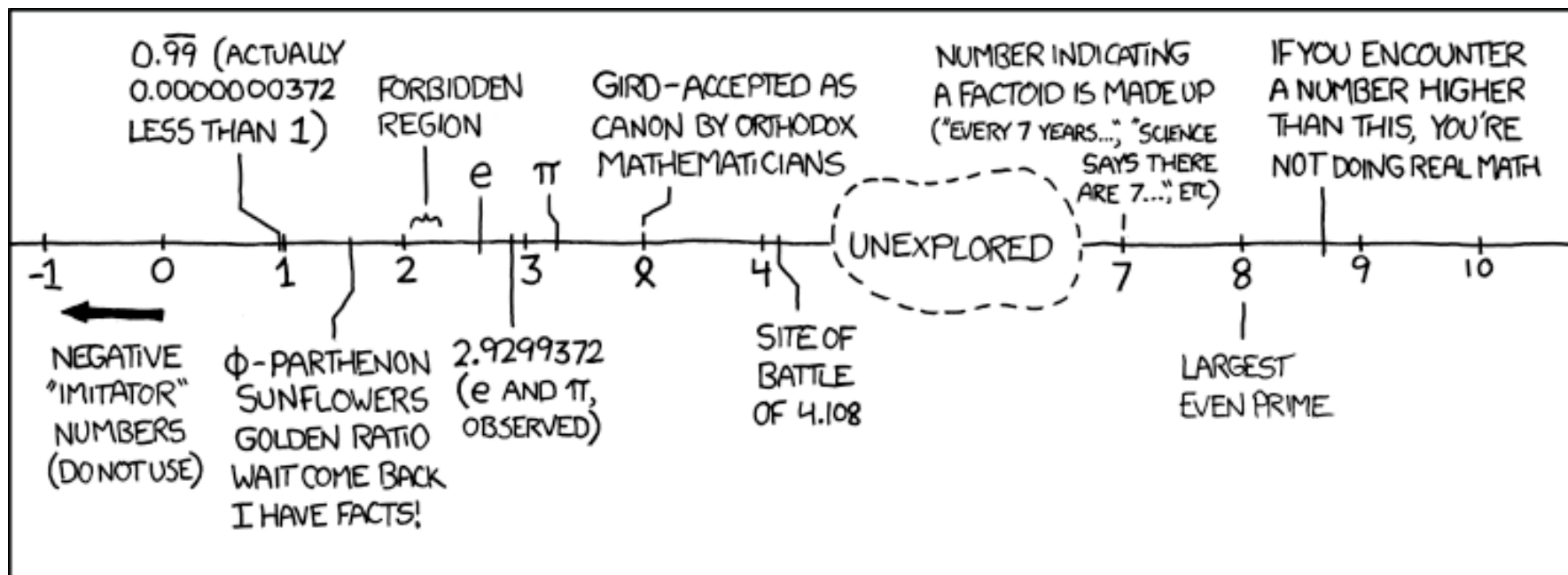
Justin Hsia

Teaching Assistants:

Aman Mohammed
 Cosmo Wang
 Joy Dang
 Kyrie Dowling
 Yan Zhe Ong

Ami Oka
 Hang Do
 Julia Wang
 Mariam Mayanja

Callum Walker
 Jim Limprasert
 Kaelin Laundry
 Shawn Stanley



Administrivia

- ❖ hw5 due Wednesday, hw6 due Friday

- ❖ Lab 1a due tonight at 11:59 pm
 - Submit `pointer.c` and `lab1Areflect.txt`
 - Make sure you submit *something* to Gradescope before the deadline and that the file names are correct
 - Can use late day tokens to submit up until Wed 11:59 pm

- ❖ Lab 1b due next Monday (10/19)
 - Submit `aisle_manager.c`, `store_client.c`, and `lab1Breflect.txt`

Reading Review

- ❖ Terminology:
 - normalized scientific binary notation
 - trailing zeros
 - sign, mantissa, exponent \leftrightarrow bit fields S, M, and E
 - float, double
 - biased notation (exponent), implicit leading one (mantissa)
 - rounding errors

- ❖ Questions from the Reading?

$$2^{-1} = 0.5$$

$$2^{-2} = 0.25$$

$$2^{-3} = 0.125$$

$$2^{-4} = 0.0625$$

Review Questions

- ❖ Convert 11.375_{10} to normalized binary scientific notation

$$8 + 2 + 1 + 0.25 + 0.125$$

$$2^3 + 2^1 + 2^0 + 2^{-2} + 2^{-3} = \underline{1011} \cdot 011_2 = \boxed{1.011011 \times 2^3}$$

- ❖ What is the correct value encoded by the following floating point number?

S E M
0b 0 | 1000 0000 | 110 0000 0000 0000 0000 0000

- bias = $2^{w-1} - 1 = 2^7 - 1 = 127$
- exponent = $E - \text{bias} = 2^7 - 127 = 128 - 127 = 1$
- mantissa = $1.M = 1.110...0_2$

$$(-1)^0 \times 1.11_2 \times 2^1 = 11.1_2 = \boxed{+3.5}$$

Number Representation Revisited

❖ What can we represent in one word?

- Signed and Unsigned Integers
- Characters (ASCII)
- Addresses

❖ How do we encode the following:

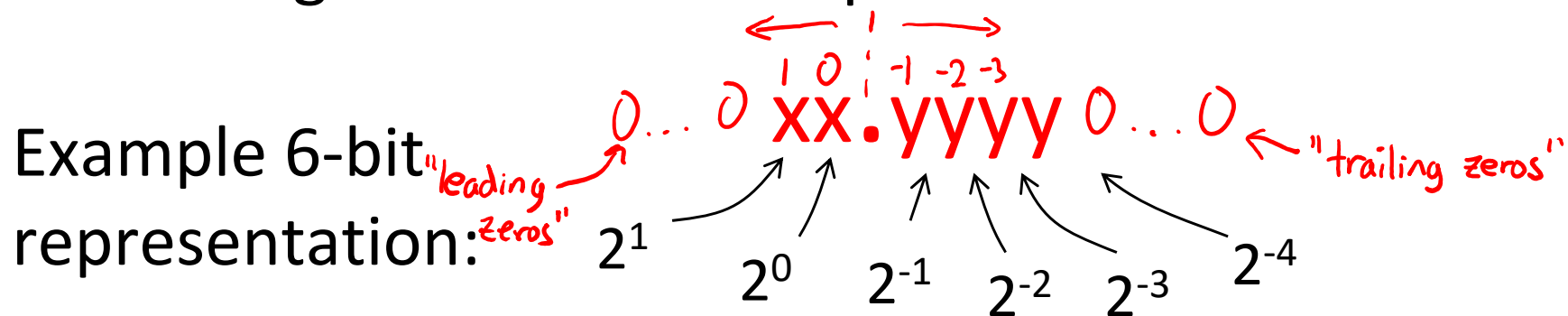
- Real numbers (*e.g.*, 3.14159)
- Very large numbers (*e.g.*, 6.02×10^{23})
- Very small numbers (*e.g.*, 6.626×10^{-34})
- Special numbers (*e.g.*, ∞ , NaN)



**Floating
Point**

Representation of Fractions

- ❖ “Binary Point,” like decimal point, signifies boundary between integer and fractional parts:

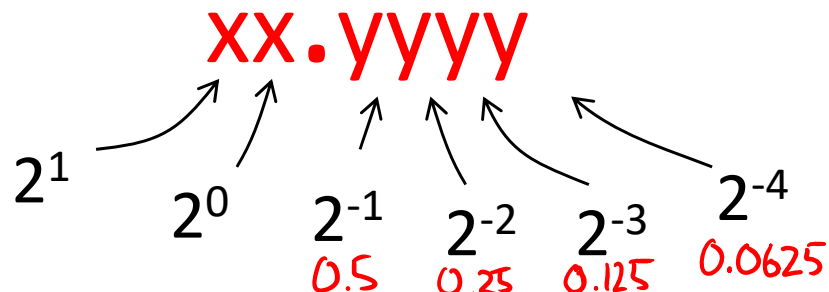


- ❖ Example: $10.1010_2 = 1 \times 2^1 + 1 \times 2^{-1} + 1 \times 2^{-3} = 2.625_{10}$

Representation of Fractions

- ❖ “Binary Point,” like decimal point, signifies boundary between integer and fractional parts:

Example 6-bit representation:



- ❖ In this 6-bit representation:
 - What is the encoding and value of the smallest (most negative) number?
 - What is the encoding and value of the largest (most positive) number?
 - What is the smallest number greater than 2 that we can represent?

$00.0000_2 = 0$

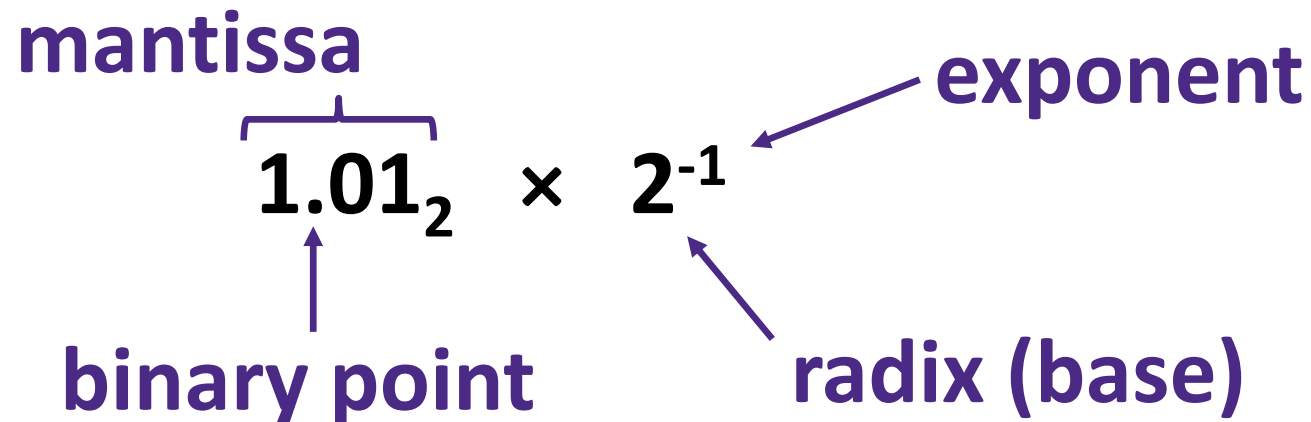
$11.111\underbrace{1}_{2^{-4}} = 4 - 2^{-4}$

$2^L = 10.0000_2$

$10.0001 = 2 + 2^{-4}$

can't represent anything in-between!



Scientific Notation (Binary)



The diagram illustrates the components of binary scientific notation. It shows the expression $1.01_2 \times 2^{-1}$. A bracket above the digits '1.01' is labeled 'mantissa'. An arrow points to the dot between '1' and '01' labeled 'binary point'. An arrow points to the '-1' in the exponent labeled 'exponent'. An arrow points to the '2' in the base labeled 'radix (base)'.

- ❖ *Normalized form*: exactly one digit (non-zero) to left of binary point
- ❖ Computer arithmetic that supports this called **floating point** due to the “floating” of the binary point
 - Declare such variable in C as `float` (or `double`)

IEEE Floating Point

- ❖ IEEE 754 (established in 1985)
 - Standard to make numerically-sensitive programs portable
 - Specifies two things: *representation scheme* and result of *floating point operations*
 - Supported by all major CPUs
- ❖ Driven by numerical concerns
 - **Scientists**/numerical analysts want them to be as **real** as possible
 - **Engineers** want them to be **easy to implement** and **fast** 
 - Scientists mostly won out:
 - Nice standards for rounding, overflow, underflow, but...
 - Hard to make fast in hardware
 - Float operations can be an order of magnitude slower than integer ops
FLOPs  *used in computer benchmarks*

Floating Point Encoding

❖ Use normalized, base 2 scientific notation:

- Value: $\pm 1 \times \text{Mantissa} \times 2^{\text{Exponent}}$

- Bit Fields: $(-1)^S \times 1.M \times 2^{(E-\text{bias})}$

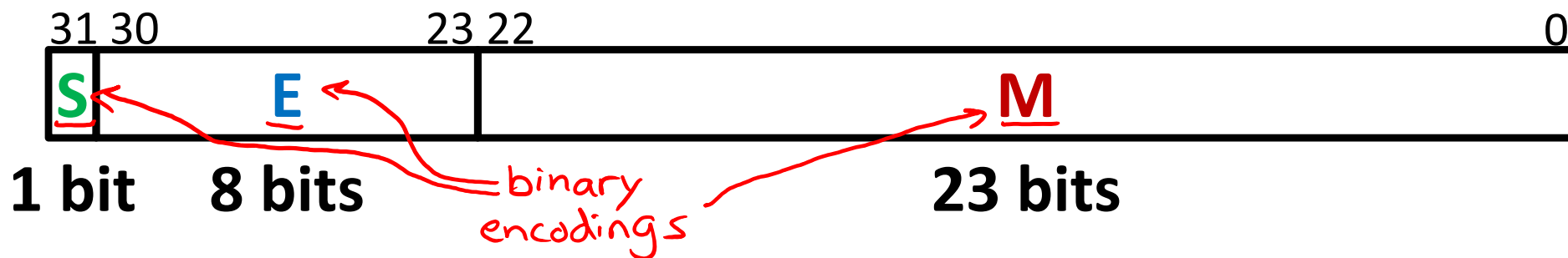
❖ Representation Scheme: *(3 separate fields within 32 bits)*

- Sign bit (0 is positive, 1 is negative)

- Mantissa (a.k.a. significand) is the fractional part of the number in normalized form and encoded in bit vector **M**

- Exponent weights the value by a (possibly negative) power of 2 and encoded in the bit vector **E**

values

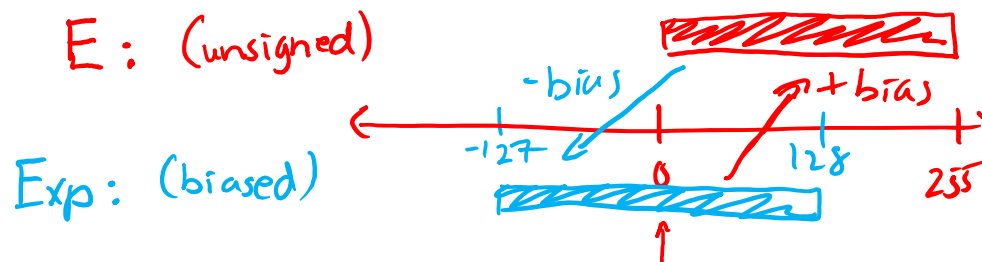


The Exponent Field

❖ Use **biased notation**

$w=8$, can encode $2^8=256$ exponents

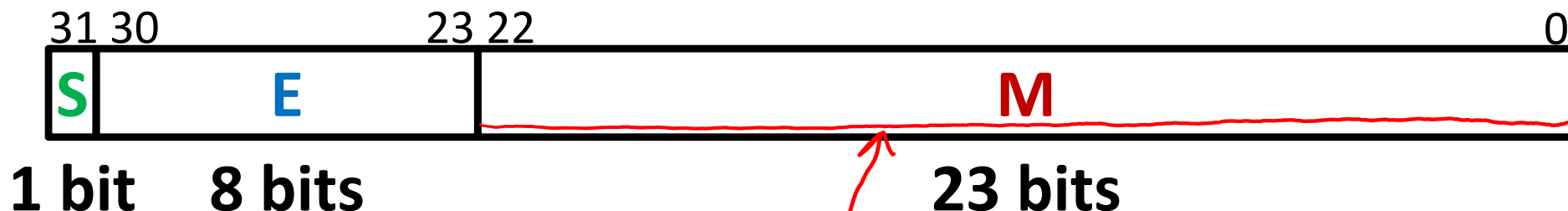
- Read exponent as unsigned, but with **bias of $2^{w-1}-1 = 127$**
- Representable exponents roughly $\frac{1}{2}$ positive and $\frac{1}{2}$ negative
- $\text{Exp} = E - \text{bias} \leftrightarrow E = \text{Exp} + \text{bias}$
 - Exponent 0 ($\text{Exp} = 0$) is represented as $E = 0b\ 0111\ 1111 = 2^7 - 1$



❖ Why biased?

- Makes floating point arithmetic easier
- Makes somewhat compatible with two's complement hardware

The Mantissa (Fraction) Field



$$(-1)^S \times (1 . M) \times 2^{(E - \text{bias})}$$

❖ Note the implicit leading 1 in front of the M bit vector

- Example: 0b ^{⊕, Exp = 0} 0011 ^{Man = 1.10...0} 1111 1100 0000 0000 0000 0000 0000 0000
 is read as $1.1_2 = 1.5_{10}$, not $0.1_2 = 0.5_{10}$

- Gives us an extra bit of *precision*

❖ Mantissa “limits”

- Low values near $M = 0b0\dots0$ are close to 2^{Exp}

$$\hookrightarrow 2^{\text{Exp}} \times 1.0\dots0 = 2^{\text{Exp}}$$

- High values near $M = 0b1\dots1$ are close to $2^{\text{Exp}+1}$

$$\hookrightarrow 2^{\text{Exp}} \times 1.1\dots1 = 2^{\text{Exp}} (2 - 2^{-23}) = 2^{\text{Exp}+1} - 2^{\text{Exp}-23}$$

Normalized Floating Point Conversions

❖ FP → Decimal

1. Append the bits of M to implicit leading 1 to form the mantissa.
2. Multiply the mantissa by $2^{E - \text{bias}}$.
3. Multiply the sign $(-1)^S$.
4. Multiply out the exponent by shifting the binary point.
5. Convert from binary to decimal.

❖ Decimal → FP

1. Convert decimal to binary.
2. Convert binary to normalized scientific notation.
3. Encode sign as S (0/1).
4. Add the bias to exponent and encode E as unsigned.
5. The first bits after the leading 1 that fit are encoded into M.

Practice Question

- ❖ Convert the decimal number **-7.375** into floating point representation

$$-7.375 = -(4+2+1 + 0.25 + 0.125) = -(2^2+2^1+2^0+2^{-2}+2^{-3}) = -111.011_2 = -1.11011_2 \times 2^2$$

$$S = \underline{1}, E = 2+127 = 129 = 0b\underline{1000\ 0001}, M = 0b\underline{11011\ 0\dots 0}$$

$$0b\underline{1100\ 0001\ 110\ 110}\ 0\dots 0 = \boxed{0x\ C0EC\ 0000}$$

Challenge Question

- ❖ Find the sum of the following binary numbers in normalized scientific binary notation:

$$\begin{array}{r}
 \begin{array}{r}
 \overset{1}{\sim} \overset{1}{\sim} \\
 0.0101 \times 2^2 \\
 + 1.11 \times 2^2 \\
 \hline
 10.0001 \times 2^2
 \end{array} \\
 \end{array}
 = \boxed{1.00001 \times 2^3}$$

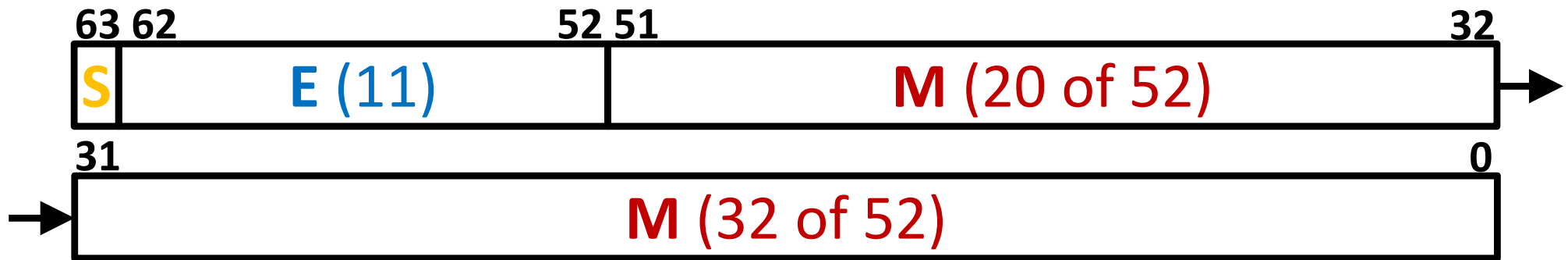
① match exponents
 ② sum mantissas
 ③ normalize

Precision and Accuracy

- ❖ **Precision** is a count of the number of bits in a computer word used to represent a value
 - Capacity for accuracy
- ❖ **Accuracy** is a measure of the difference between the *actual value of a number* and its computer representation
 - *High precision permits high accuracy but doesn't guarantee it. It is possible to have high precision but low accuracy.*
 - **Example:** `float pi = 3.14;`
 - `pi` will be represented using all 24 bits of the mantissa (highly precise), but is only an approximation (not accurate)

Need Greater Precision?

❖ **Double Precision** (vs. Single Precision) in 64 bits



- C variable declared as double
- Exponent bias is now $2^{10}-1 = 1023$, *bias = $2^{w-1}-1$*
- **Advantages:** greater precision (larger mantissa), greater range (larger exponent)
- **Disadvantages:** more bits used, slower to manipulate

Current Limitations

❖ Largest magnitude we can represent?

$\curvearrowright \text{Exp} = 128$
 $E = 0b1111\ 1111, M = 0b1\dots1$

❖ Smallest magnitude we can represent?

$E = 0b0000\ 0000, M = 0b0\dots0$
 $\curvearrowright \text{Exp} = -127$

■ Limited *range* due to width of **E** field

❖ What happens if we try to represent $2^0 + 2^{-30}$?

$1.\overbrace{0\dots0}^{29 \text{ zeros}}$
 \uparrow
 M stores first 23 zeros

■ Rounding due to limited *precision*: stores 2^0

❖ There is a need for *special cases*

■ How do we represent the value zero?

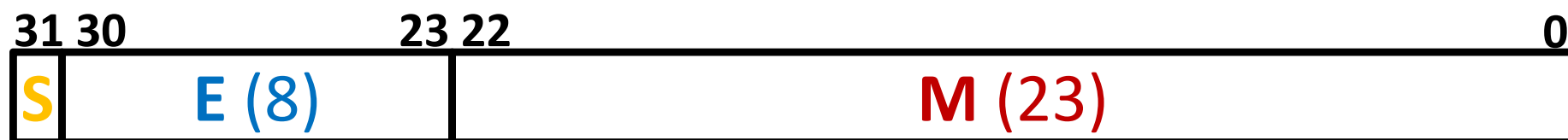
$0 \neq \pm 1.M \times 2^{E-\text{bias}}$

■ What about ∞ and NaN?

???

Summary

- ❖ Floating point approximates real numbers:



- Handles large numbers, small numbers, special numbers
- Exponent in biased notation (bias = $2^{w-1} - 1$)
 - Size of exponent field determines our representable *range*
 - Outside of representable exponents is *overflow* and *underflow*
- Mantissa approximates fractional portion of binary point
 - Size of mantissa field determines our representable *precision*
 - Implicit leading 1 (normalized) except in special cases
 - Exceeding length causes *rounding*