

System Programming

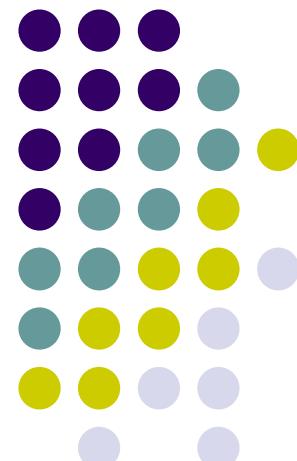
05. IEEE floating point examples

2019. Fall

Instructor: Joonho Kwon

jhwon@pusan.ac.kr

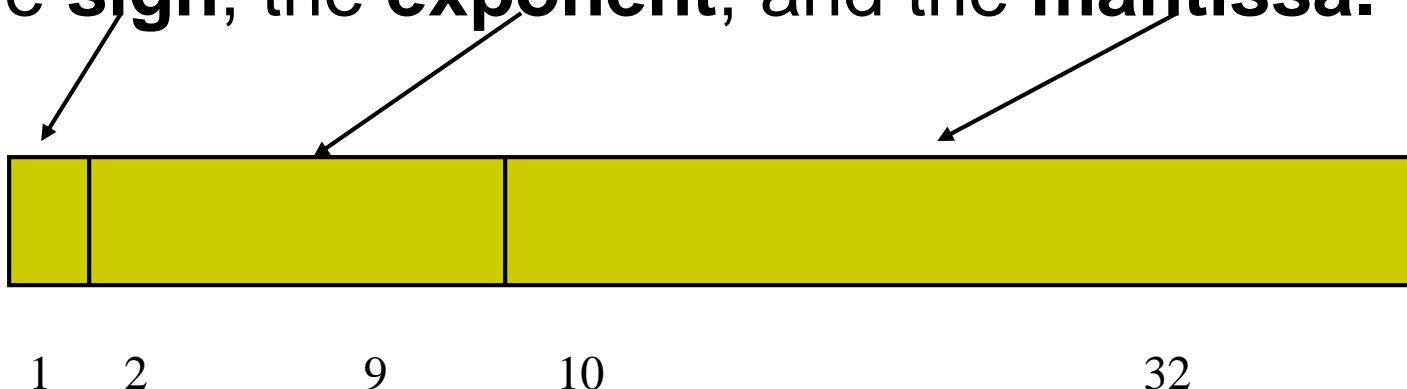
Data Science Lab @ PNU



IEEE Floating Point Representation



- Floating point numbers can be stored into 32-bits, by dividing the bits into three parts:
the sign, the exponent, and the mantissa.



IEEE Floating Point Representation



- The first (leftmost) field of our floating point representation will **STILL** be the sign bit:
 - 0 for a positive number,
 - 1 for a negative number.

Storing the Binary Form



How do we store a radix point?

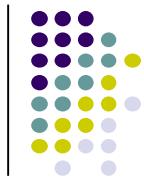
- All we have are zeros and ones...

Make sure that the radix point is **ALWAYS** in the same position within the number.

Use the IEEE 32-bit standard

→ the **leftmost** digit must be a 1

Solution is Normalization



Every binary number, **except the one corresponding to the number zero**, can be normalized by choosing the exponent so that the radix point falls to the right of the leftmost 1 bit.

$$37.25_{10} = \underbrace{100101.01}_2 = 1.0010101 \times 2^5$$

$$7.625_{10} = \underbrace{111.101}_2 = 1.11101 \times 2^2$$

$$0.3125_{10} = \underbrace{0.0101}_2 = 1.01 \times 2^{-2}$$

IEEE Floating Point Representation



- The second field of the floating point number will be the **exponent**.
- The exponent is stored as an unsigned 8-bit number, RELATIVE to a **bias of 127**.
 - Exponent 5 is stored as $(127 + 5)$ or 132
 - $132 = 10000100$
 - Exponent -5 is stored as $(127 + (-5))$ or 122
 - $122 = 01111010$

Try It Yourself



How would the following exponents be stored (8-bits, 127-biased):

$$2^{-10}$$

$$2^8$$

(Answers on next slide)

Answers



$$2^{-10}$$

exponent

-10

8-bit

bias

+127

value

117 → 01110101

$$\overline{2^8}$$

exponent

8

8-bit

bias

+127

value

135 → 10000111

IEEE Floating Point Representation



- The **mantissa** is the set of 0's and 1's to the right of the radix point of the **normalized** (when the digit to the left of the radix point is 1) binary number.

Ex: **1.00101** $\times 2^3$

(The mantissa is 00101)

- The mantissa is stored in a 23 bit field, so we add zeros to the right side and store:

001010000000000000000000

Decimal Floating Point to IEEE standard Conversion



Ex 1: Find the IEEE FP representation of
40.15625

Step 1.

Compute the binary equivalent of the whole part and the fractional part. (i.e. convert **40** and **.15625** to their binary equivalents)

Decimal Floating Point to IEEE standard Conversion



$\begin{array}{r} 40 \\ - 32 \\ \hline 8 \\ - 8 \\ \hline 0 \end{array}$	Result: 101000	$\begin{array}{r} .15625 \\ - .12500 \\ \hline .03125 \\ - .03125 \\ \hline .0 \end{array}$	Result: $.00101$
--	----------------------------	---	----------------------------

So: $40.15625_{10} = 101000.00101_2$

Decimal Floating Point to IEEE standard Conversion



Step 2. Normalize the number by moving the decimal point to the right of the leftmost one.

$$101000.00101 = 1.010000101 \times 2^5$$
A red curved arrow originates from the decimal point in the original binary floating-point number "101000.00101" and points to the decimal point in the normalized form "1.010000101".

Decimal Floating Point to IEEE standard Conversion



Step 3. Convert the exponent to a biased exponent

$$127 + 5 = 132$$

And convert biased exponent to 8-bit unsigned binary:

$$132_{10} = 10000100_2$$

Decimal Floating Point to IEEE standard Conversion



Step 4. Store the results from steps 1-3:

Sign	Exponent	Mantissa
(from step 3)	(from step 2)	

0	10000100	01000001010000000000000
---	----------	-------------------------

Decimal Floating Point to IEEE standard Conversion



Ex 2: Find the IEEE FP representation of **-24.75**

Step 1. Compute the binary equivalent of the whole part and the fractional part.

24	.75
<u>- 16</u>	Result:
8	11000
<u>- 8</u>	.25
0	.11
	<u>- .25</u>
	.0

$$\text{So: } -24.75_{10} = -11000.11_2$$

Decimal Floating Point to IEEE standard Conversion



Step 2.

Normalize the number by moving the decimal point to the right of the leftmost one.

$$-11000.11 = -1.100011 \times 2^4$$

Decimal Floating Point to IEEE standard Conversion.



Step 3. Convert the exponent to a biased exponent

$$127 + 4 = 131$$
$$\Rightarrow 131_{10} = 10000011_2$$

Step 4. Store the results from steps 1-3

Sign	Exponent	mantissa
1	10000011	1000110..0

IEEE standard to Decimal Floating Point Conversion.



- Do the steps in reverse order
- In reversing the normalization step move the radix point the number of digits equal to the exponent:
 - If exponent is **positive**, move to the **right**
 - If exponent is **negative**, move to the **left**

IEEE standard to Decimal Floating Point Conversion.



Ex 1: Convert the following 32-bit binary number to its decimal floating point equivalent:

Sign

Exponent

Mantissa

1

01111101

010..0

IEEE standard to Decimal Floating Point Conversion..



Step 1: Extract the biased exponent and unbiased it

$$\text{Biased exponent} = 01111101_2 = 125_{10}$$

$$\text{Unbiased Exponent: } 125 - 127 = -2$$

IEEE standard to Decimal Floating Point Conversion..



Step 2: Write Normalized number in the form:

1 . Mantissa x 2 ^{Exponent} -----

For our number:

$$-1.01 \times 2^{-2}$$

IEEE standard to Decimal Floating Point Conversion.



Step 3: Denormalize the binary number from step 2
(i.e. move the decimal and get rid of ($\times 2^n$) part):

-0.0101_2 (negative exponent – move left)

Step 4: Convert binary number to the FP equivalent
(i.e. Add all column values with 1s in them)

$$-0.0101_2 = - (0.25 + 0.0625)$$

$$= -0.3125_{10}$$

IEEE standard to Decimal Floating Point Conversion.



Ex 2: Convert the following 32 bit binary number to its decimal floating point equivalent:

Sign

0

Exponent

10000011

Mantissa

10011000..0

IEEE standard to Decimal Floating Point Conversion..



Step 1: Extract the biased exponent and unbiased it

$$\text{Biased exponent} = 1000011_2 = 131_{10}$$

$$\text{Unbiased Exponent: } 131 - 127 = 4$$

IEEE standard to Decimal Floating Point Conversion..



Step 2: Write Normalized number in the form:

1 . Mantissa $\times 2^{\text{Exponent}}$

For our number:

$$1.10011 \times 2^4$$

IEEE standard to Decimal Floating Point Conversion.



Step 3: Denormalize the binary number from step 2
(i.e. move the decimal and get rid of ($\times 2^n$) part:

11001.1_2 (positive exponent – move right)

Step 4: Convert binary number to the FP equivalent
(i.e. Add all column values with 1s in them)

$$11001.1 = 16 + 8 + 1 + .5$$

$$= 25.5_{10}$$