FLOATING POINT

Ayman Hajja, PhD

REVIEW OF NUMBERS

- Computers are made to deal with numbers
- What can be represented in N bits?
  - 2^N things, and no more! They could be:
    - Unsigned integers
    - Signed integers
    - How about bigger/smaller or numbers with fractional part?

WHAT ABOUT OTHER NUMBERS?

1. Very large numbers? $8.15569 * 10^{10}$
2. Very small numbers? $5.29177 * 10^{-11}$
3. Numbers with both integer and fractional parts? 1.5

- Let's consider the third part first, our solution will help with the first and second

REPRESENTATION OF FRACTIONS WITH FIXED POINT

- "Binary Point" like decimal point signifies boundary between integer and fractional parts:
- Example of a 6-bit representation: xx.yyyy

Bit	x	x	y	y	y	y
Positional Value	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}

- If we assume "fixed binary point", range of 6-bit representation with this format:

0 to 3.9375 (almost 4)

REPRESENTATION OF FRACTIONS WITH FIXED POINT

- "Binary Point" like decimal point signifies boundary between integer and fractional parts:
- Example of a 6-bit representation: xx.yyyy
- $10.1010_2 = ?$
- If we assume "fixed binary point", range of 6-bit representation with this format:

0 to 3.9375 (almost 4)

REPRESENTATION OF FRACTIONS WITH FIXED POINT

- "Binary Point" like decimal point signifies boundary between integer and fractional parts:
- Example of a 6-bit representation: xx.yyyy
- $10.1010_2 = ?$
- $1*2^1 + 1*2^{-1} + 1*2^{-3} = 2.625$
- If we assume "fixed binary point", range of 6-bit representation with this format:

0 to 3.9375 (almost 4)

REPRESENTATION OF FRACTIONS WITH FIXED POINT

- "Binary Point" like decimal point signifies boundary between integer and fractional parts:
- Example of a 6-bit representation: xx.yyyy
- $10.1010_2 = ?$
- $1*2^1 + 1*2^{-1} + 1*2^{-3} = 2.625$
- If we assume "fixed binary point", range of 6-bit representation with this format:

How close is this number to 4?

0 to 3.9375 (almost 4)

REPRESENTATION OF FRACTIONS

- Example: convert the binary to a real decimal:

000000.101100000

1/2 = 0.5
1/4 = 0.25
1/8 = 0.125
1/16 = 0.0625
1/32 = 0.03125
1/64 = 0.015625
1/128 = 0.0078125

REPRESENTATION OF FRACTIONS

- Example: convert the following real decimal to binary:

0.1640625

1/2 = 0.5
1/4 = 0.25
1/8 = 0.125
1/16 = 0.0625
1/32 = 0.03125
1/64 = 0.015625
1/128 = 0.0078125

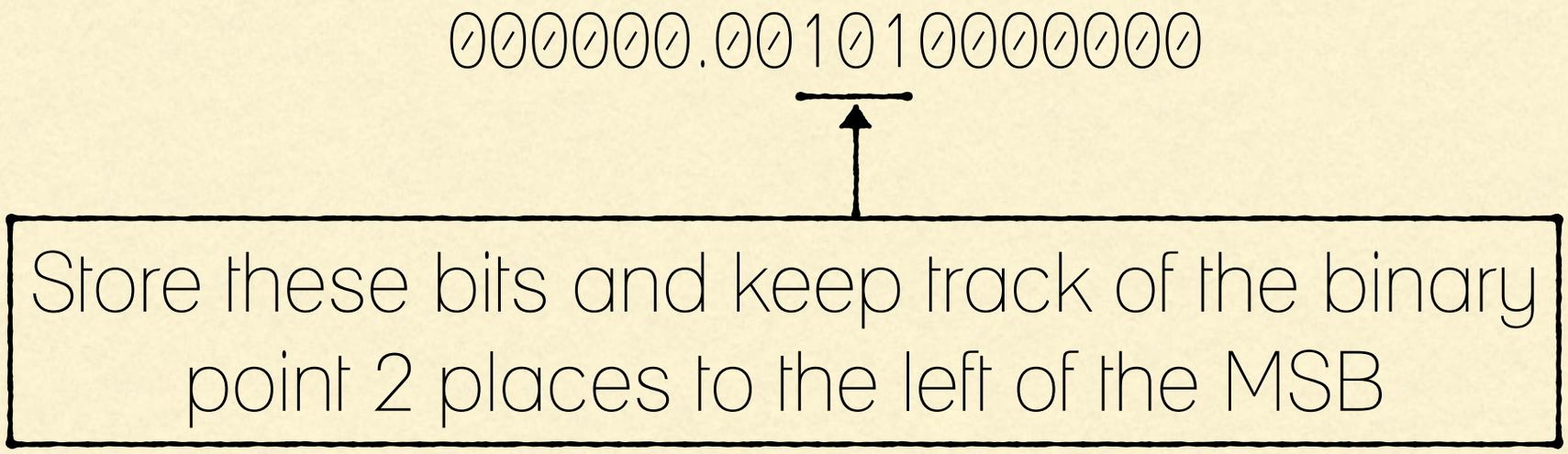
REPRESENTATION OF FRACTIONS

- So far, in our examples we used a “fixed” binary point, what we really want is to “float” the binary point to effectively use our limited bits
- Can we store the following sequence in a xx.yyyy 6-bit representation?

000000.001010000000

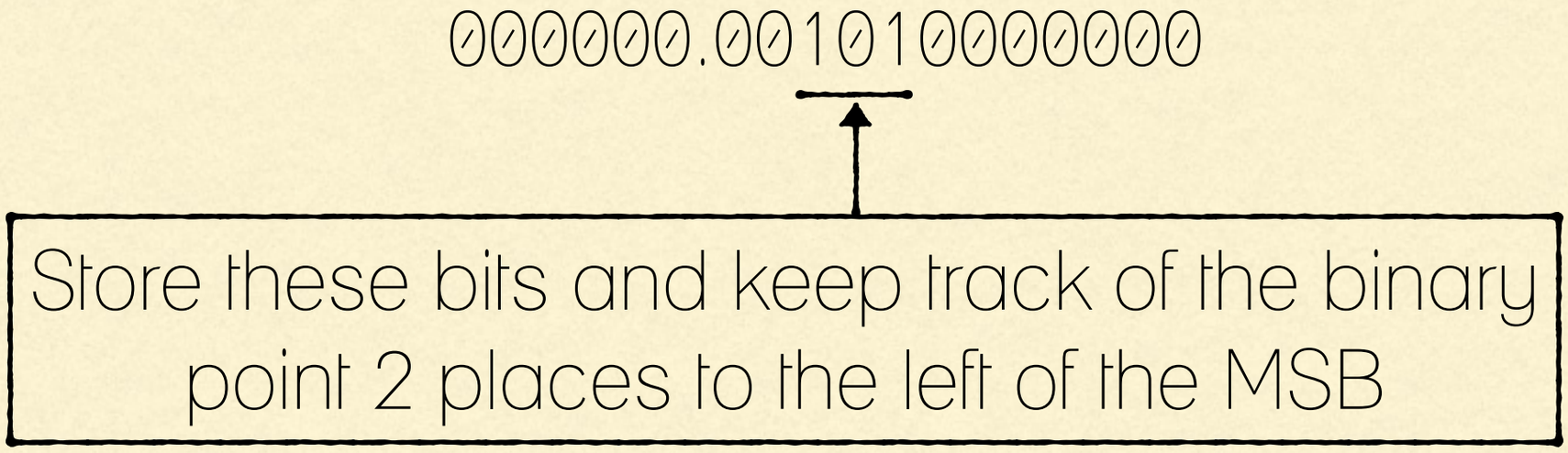
REPRESENTATION OF FRACTIONS

- So far, in our examples we used a “fixed” binary point, what we really want is to “float” the binary point to effectively use our limited bits
- Can we store the following sequence in a xx.yyyy 6-bit representation?

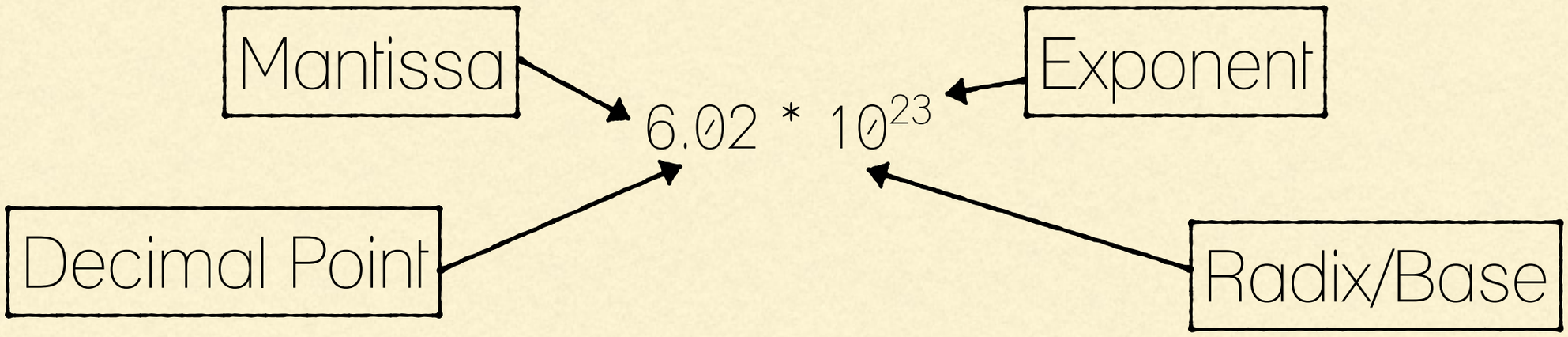


REPRESENTATION OF FRACTIONS

- With floating point representation, each numeral carries an exponent field recording the whereabouts of its binary point

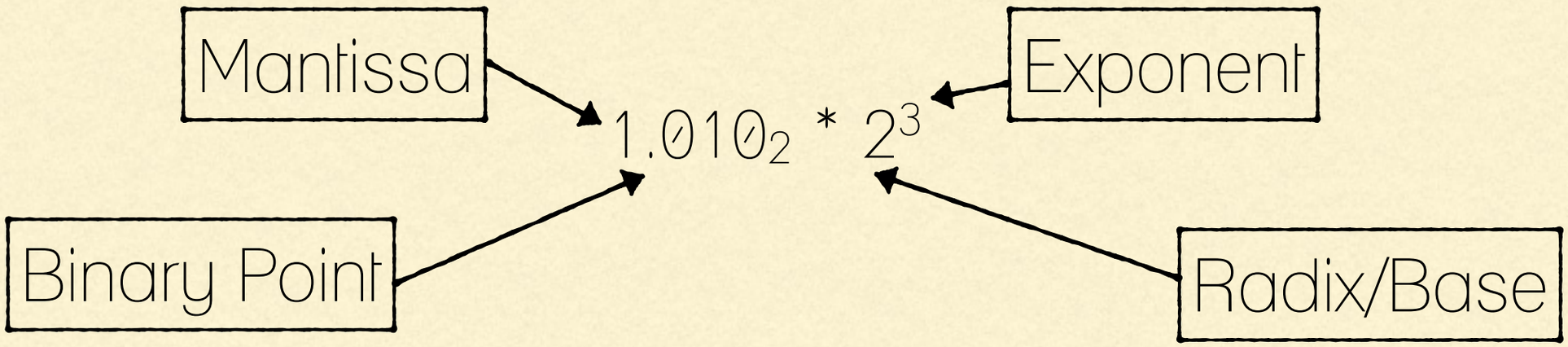


SCIENTIFIC NOTATION (IN DECIMAL)



- Normalized form: no leading 0s (exactly one digit to left of decimal point):
  - Normalized: $1.0 * 10^{-9}$
  - Not normalized: $0.1 * 10^{-8}$ or $10.0 * 10^{-10}$

SCIENTIFIC NOTATION (IN BINARY)



$$1.010_2 * 2^3 = 1.25 * 8 = 10$$

- Computer arithmetic that supports it called floating point, because it represents numbers where the binary point is not fixed

SCIENTIFIC NOTATION (IN BINARY)

- To store the following binary number:

$$000000.001010100000$$

- We first put it in normal scientific form:

$$1.0101 * 2^{-3}$$

SCIENTIFIC NOTATION (IN BINARY)

- To store the following binary number:

000000.001010100000

- We first put it in normal scientific form:

$1.0101 * 2^{-3}$

1) We store '0101' in one field (called the significand)

SCIENTIFIC NOTATION (IN BINARY)

- To store the following binary number:

000000.001010100000

- We first put it in normal scientific form:

$1.0101 * 2^{-3}$

1) We store '0101' in one field (called the significand)

2) We store -3 in another filed (called the exponent)

SCIENTIFIC NOTATION (IN BINARY)

- To store the following binary number:

000000.001010100000

- We first put it in normal scientific form:

$1.0101 * 2^{-3}$

1) We store ‘0101’ in one field (called the significand)

2) We store -3 in another filed (called the exponent)

3) We store the sign of the number (positive) in a third field

FLOATING POINT REPRESENTATION

- Normal format: $+1.xxx\ldots x_2 * 2^{yyy\ldots y}$
- Multiple of Word Size (32 bits)

S	Exponent (8 bits)	Significand (23 bits)

- S represents Sign
  - Exponent represents y's
  - Significand represents x's

FLOATING POINT REPRESENTATION

- Normal format: $+1.xxx\ldots x_2 * 2^{yyy\ldots y}$
- Multiple of Word Size (32 bits)

S	Exponent (8 bits)	Significand (23 bits)

- S represents Sign
  - Exponent represents y's
  - Significand represents x's
- Represents numbers as small as $2.0 * 10^{-38}$ to as large as $2.0 * 10^{38}$

FLOATING POINT REPRESENTATION

- Normal format: $+1.xxx\ldots x_2 * 2^{yyy\ldots y}$
- Multiple of Word Size (32 bits)

S	Exponent (8 bits)	Significand (23 bits)

- S represents Sign
  - Exponent represents y's
  - Significand represents x's

But how is that possible? With only 32 bits we can represent numbers between $-2.0 * 10^{38}$ and $2.0 * 10^{38}$?

- Represents numbers as small as $2.0 * 10^{-38}$ to as large as $2.0 * 10^{38}$

SIMPLIFIED EXAMPLE

- Normal format: $+1.xxx_2 * 2^{yyyy}$
- Simplified example using a smaller sequence (total of 8 bits)

S	Exponent (4 bits)	Significand (3 bits)

Exponent	S (Sign bit)	Significand	Value (Binary)	Value (Decimal)
Value 0	0 Positive	000	1.000	1
Value 0	0 Positive	001	1.001	1.125
Value 0	0 Positive	010	1.010	1.25
Value 0	0 Positive	011	1.011	1.375
Value 0	0 Positive	100	1.100	1.5
Value 0	0 Positive	101	1.101	1.625
Value 0	0 Positive	110	1.110	1.75
Value 0	0 Positive	111	1.111	1.875

SIMPLIFIED EXAMPLE

- Normal format: $+1.xxx_2 * 2^{yyyy}$
- Simplified example using a smaller sequence (total of 8 bits)

S	Exponent (4 bits)	Significand (3 bits)

Exponent	S (Sign bit)	Significand	Value (Binary)	Value (Decimal)
Value 1	0 Positive	000	10.00	2
Value 1	0 Positive	001	10.01	2.25
Value 1	0 Positive	010	10.10	2.5
Value 1	0 Positive	011	10.11	2.75
Value 1	0 Positive	100	11.00	3
Value 1	0 Positive	101	11.01	3.25
Value 1	0 Positive	110	11.10	3.5
Value 1	0 Positive	111	11.11	3.75

SIMPLIFIED EXAMPLE

- Normal format: $+1.xxx_2 * 2^{yyyy}$
- Simplified example using a smaller sequence (total of 8 bits)

S	Exponent (4 bits)	Significand (3 bits)

Exponent	S (Sign bit)	Significand	Value (Binary)	Value (Decimal)
Value 2	0 Positive	000	100.0	4
Value 2	0 Positive	001	100.1	4.5
Value 2	0 Positive	010	101.0	5
Value 2	0 Positive	011	101.1	5.5
Value 2	0 Positive	100	110.0	6
Value 2	0 Positive	101	110.1	6.6
Value 2	0 Positive	110	111.0	7
Value 2	0 Positive	111	111.1	7.5

SIMPLIFIED EXAMPLE

- Normal format: $+1.xxx_2 * 2^{yyyy}$
- Simplified example using a smaller sequence (total of 8 bits)

S	Exponent (4 bits)	Significand (3 bits)

Exponent	S (Sign bit)	Significand	Value (Binary)	Value (Decimal)
Value 3	0 Positive	000	1000.	8
Value 3	0 Positive	001	1001.	9
Value 3	0 Positive	010	1010.	10
Value 3	0 Positive	011	1011.	11
Value 3	0 Positive	100	1100.	12
Value 3	0 Positive	101	1101.	13
Value 3	0 Positive	110	1110.	14
Value 3	0 Positive	111	1111.	15

SIMPLIFIED EXAMPLE

- Normal format: $+1.xxx_2 * 2^{yyyy}$
- Simplified example using a smaller sequence (total of 8 bits)

S	Exponent (4 bits)	Significand (3 bits)

What's the maximum value for our exponent?

Exponent	S			Value (Decimal)
Value 3				8
Value 3				9
Value 3	0 Positive	010	1010.	10
Value 3	0 Positive	011	1011.	11
Value 3	0 Positive	100	1100.	12
Value 3	0 Positive	101	1101.	13
Value 3	0 Positive	110	1110.	14
Value 3	0 Positive	111	1111.	15

SIMPLIFIED EXAMPLE

- Normal format: $+1.xxx_2 * 2^{yyyy}$
- Simplified example using a smaller sequence (total of 8 bits)

S	Exponent (4 bits)	Significand (3 bits)

What's the maximum value for our exponent?

In 2's complement, it's 7, but in this case it's 8. We'll explain more later in this lecture!

Exponent	S			Value (Decimal)
Value 3				8
Value 3				9
Value 3				0
Value 3				1
Value 3				2
Value 3	0 Positive	101	1101.	13
Value 3	0 Positive	110	1110.	14
Value 3	0 Positive	111	1111.	15

SIMPLIFIED EXAMPLE

- Normal format: $+1.xxx_2 * 2^{yyyy}$
- Simplified example using a smaller sequence (total of 8 bits)

S	Exponent (4 bits)	Significand (3 bits)

Exponent	S (Sign bit)	Significand	Value (Binary)	Value (Decimal)
Value 8	0 Positive	000	100000000.	256
Value 8	0 Positive	001	100100000.	288
Value 8	0 Positive	010	101000000.	320
Value 8	0 Positive	011	101100000.	352
Value 8	0 Positive	100	110000000.	384
Value 8	0 Positive	101	110100000.	416
Value 8	0 Positive	110	111000000.	448
Value 8	0 Positive	111	111100000.	480

SIMPLIFIED EXAMPLE

- Normal format: $+1.xxx_2 * 2^{yyyy}$
- Simplified example using a smaller sequence (total of 8 bits)

S	Exponent (4 bits)	Significand (3 bits)

Exponent	S (Sign bit)	Significand	Value (Binary)	Value (Decimal)
Value 0	0 Positive	000	1.000	1
Value 0	0 Positive	001	1.001	1.125
Value 0	0 Positive	010	1.010	1.25
Value 0	0 Positive	011	1.011	1.375
Value 0	0 Positive	100	1.100	1.5
Value 0	0 Positive	101	1.101	1.625
Value 0	0 Positive	110	1.110	1.75
Value 0	0 Positive	111	1.111	1.875

SIMPLIFIED EXAMPLE

- Normal format: $+1.xxx_2 * 2^{yyyy}$
- Simplified example using a smaller sequence (total of 8 bits)

S	Exponent (4 bits)	Significand (3 bits)

Exponent	S (Sign bit)	Significand	Value (Binary)	Value (Decimal)
Value -1	0 Positive	000	.1000	0.5
Value -1	0 Positive	001	.1001	0.5625
Value -1	0 Positive	010	.1010	0.625
Value -1	0 Positive	011	.1011	0.6875
Value -1	0 Positive	100	.1100	0.75
Value -1	0 Positive	101	.1101	0.8125
Value -1	0 Positive	110	.1110	0.875
Value -1	0 Positive	111	.1111	0.9375

SIMPLIFIED EXAMPLE

- Normal format: $+1.xxx_2 * 2^{yyyy}{}_2$
- Simplified example using a smaller sequence (total of 8 bits)

S	Exponent (4 bits)	Significand (3 bits)

Exponent	S (Sign bit)	Significand	Value (Binary)	Value (Decimal)
Value -2	0 Positive	000	.01000	0.25
Value -2	0 Positive	001	.01001	0.28125
Value -2	0 Positive	010	.01010	0.3125
Value -2	0 Positive	011	.01011	0.34375
Value -2	0 Positive	100	.01100	0.375
Value -2	0 Positive	101	.01101	0.40625
Value -2	0 Positive	110	.01110	0.4375
Value -2	0 Positive	111	.01111	0.46875

SIMPLIFIED EXAMPLE

- Normal format: $+1.xxx_2 * 2^{yyyy_2}$
- Simplified example using a smaller sequence (total of 8 bits)

S	Exponent (4 bits)	Significand (3 bits)

Exponent	S (Sign bit)	Significand	Value (Binary)	Value (Decimal)
Value -3	0 Positive	000	.001000	0.125
Value -3	0 Positive	001	.001001	0.140625
Value -3	0 Positive	010	.001010	0.15625
Value -3	0 Positive	011	.001011	0.171875
Value -3	0 Positive	100	.001100	0.1875
Value -3	0 Positive	101	.001101	0.203125
Value -3	0 Positive	110	.001110	0.21875
Value -3	0 Positive	111	.001111	0.234375

SIMPLIFIED EXAMPLE

- Normal format: $+1.xxx_2 * 2^{yyyy}{}_2$
- Simplified example using a smaller sequence (total of 8 bits)

S	Exponent (4 bits)	Significand (3 bits)

What's the minimum value for our exponent?

Exponent	S			Value (Decimal)
Value -3				0.125
Value -3				0.140625
Value -3	0 Positive	010	.001010	0.15625
Value -3	0 Positive	011	.001011	0.171875
Value -3	0 Positive	100	.001100	0.1875
Value -3	0 Positive	101	.001101	0.203125
Value -3	0 Positive	110	.001110	0.21875
Value -3	0 Positive	111	.001111	0.234375

SIMPLIFIED EXAMPLE

- Normal format: $+1.xxx_2 * 2^{yyyy}{}_2$
- Simplified example using a smaller sequence (total of 8 bits)

S	Exponent (4 bits)	Significand (3 bits)

What's the minimum value for our exponent?

In 2's complement, it's -8, but in this case it's -7. We'll explain more later in this lecture!

Exponent	S			Value (Decimal)
Value -3				0.125
Value -3	0 Positive	001	.000101	0.140625
Value -3				5625
Value -3				71875
Value -3				375
Value -3	0 Positive	101	.001101	0.203125
Value -3	0 Positive	110	.001110	0.21875
Value -3	0 Positive	111	.001111	0.234375

SIMPLIFIED EXAMPLE

- Normal format: $+1.xxx_2 * 2^{yyyy_2}$
- Simplified example using a smaller sequence (total of 8 bits)

S	Exponent (4 bits)	Significand (3 bits)

Exponent	S (Sign bit)	Significand	Value (Binary)	Value (Decimal)
Value -7	0 Positive	000	.0000001000	0.0078125
Value -7	0 Positive	001	.0000001001	0.0087890625
Value -7	0 Positive	010	.0000001010	0.009765625
Value -7	0 Positive	011	.0000001011	0.0107421875
Value -7	0 Positive	100	.0000001100	0.01171875
Value -7	0 Positive	101	.0000001101	0.0126953125
Value -7	0 Positive	110	.0000001110	0.013671875
Value -7	0 Positive	111	.0000001111	0.0146484375

SIMPLIFIED EXAMPLE

- Normal format: $+1.xxx_2 * 2^{yyyy_2}$
- Simplified example using a smaller sequence (total of 8 bits)

S	Exponent (4 bits)	Significand (3 bits)

Two problems. What are they?

Exponent	S (Sign bit)	Significand	Value (Binary)	Value (Decimal)
Value -7	0 Positive	000	.0000001000	0.0078125
Value -7	0 Positive	001	.0000001001	0.0087890625
Value -7	0 Positive	010	.0000001010	0.009765625
Value -7	0 Positive	011	.0000001011	0.0107421875
Value -7	0 Positive	100	.0000001100	0.01171875
Value -7	0 Positive	101	.0000001101	0.0126953125
Value -7	0 Positive	110	.0000001110	0.013671875
Value -7	0 Positive	111	.0000001111	0.0146484375

SIMPLIFIED EXAMPLE

- N
- Si

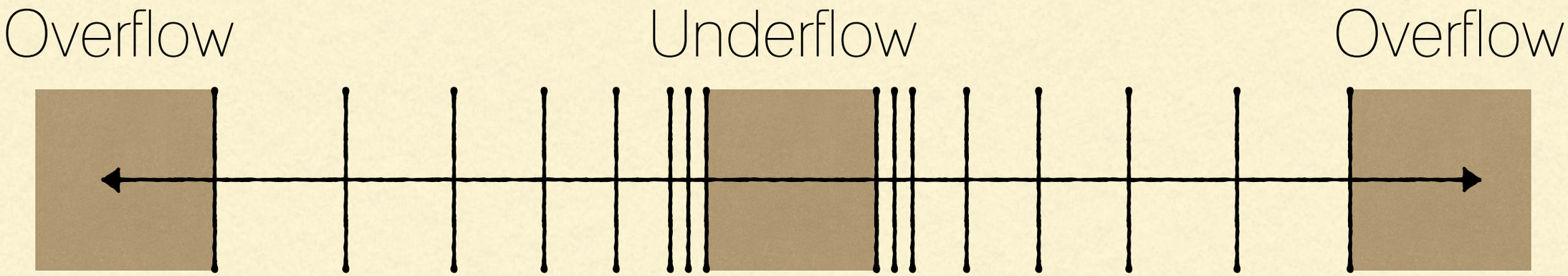
We notice here that 0.0078125 is the smallest number we can represent! But how can we represent 0, and why is the difference between 0.0078125 and 0 is much bigger than the difference between 0.0078125 & 0.0087890625?

Exponent	S (Sign bit)	Significand	Value (Binary)	Value (Decimal)
Value -7	0 Positive	000	.0000001000	0.0078125
Value -7	0 Positive	001	.0000001001	0.0087890625
Value -7	0 Positive	010	.0000001010	0.009765625
Value -7	0 Positive	011	.0000001011	0.0107421875
Value -7	0 Positive	100	.0000001100	0.01171875
Value -7	0 Positive	101	.0000001101	0.0126953125
Value -7	0 Positive	110	.0000001110	0.013671875
Value -7	0 Positive	111	.0000001111	0.0146484375

SIMPLIFIED EXAMPLE

- N
- Si

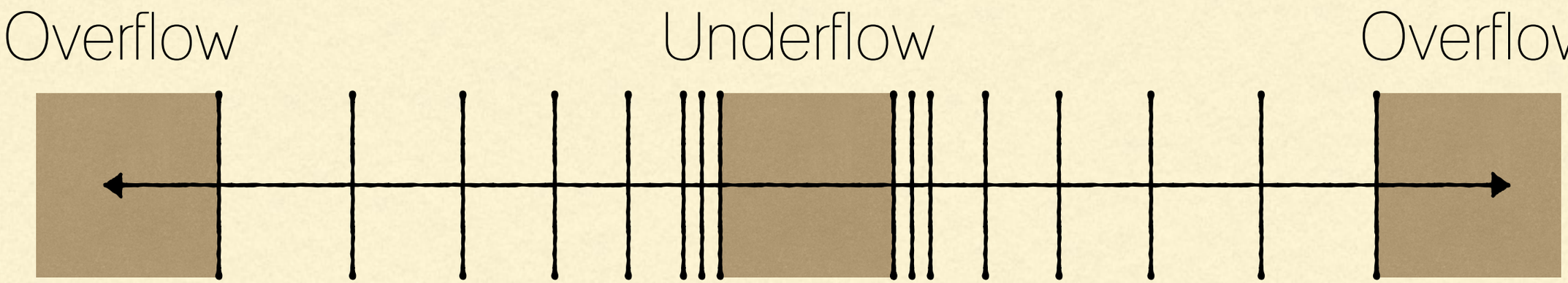
We notice here that 0.0078125 is the smallest number we can represent! But how can we represent 0, and why is the difference between 0.0078125 and 0 is much bigger than the difference between 0.0078125 & 0.0087890625?



SIMPLIFIED EXAMPLE

- N
- Si

We notice here that 0.0078125 is the smallest number we can represent! But how can we represent 0, and why is the difference between 0.0078125 and 0 is much bigger than the difference between 0.0078125 & 0.0087890625?



Solution: When exponent value is -7, we treat it as -6 but denormalized (no implied 1 to the left of the binary point)

SIMPLIFIED EXAMPLE

- Nor
- Sim

Before, we used to take the significand, add the implied 1, then multiply it by 2 (base) raised to the power of the exponent:
$1.000 * 2^{-7} = .0000001000_2 = 0.0078125_{10}$

S	Exponent (4 bits)	Significand (3 bits)

Exponent	S (Sign bit)	Significand	Value (Binary)	Value (Decimal)
Value -7	0 Positive	000	.0000001000	0.0078125
Value -7	0 Positive	001	.0000001001	0.0087890625
Value -7	0 Positive	010	.0000001010	0.009765625
Value -7	0 Positive	011	.0000001011	0.0107421875
Value -7	0 Positive	100	.0000001100	0.01171875
Value -7	0 Positive	101	.0000001101	0.0126953125
Value -7	0 Positive	110	.0000001110	0.013671875
Value -7	0 Positive	111	.0000001111	0.0146484375

SIMPLIFIED EXAMPLE

- Nor
- Sim

Now, we DO NOT add the implied 1, we just simply multiply it by 2 (base) raised to the power of the exponent minus 1 (-6):

$0.000 * 2^{-6} = .000000000_2 = 0.000000_{10}$

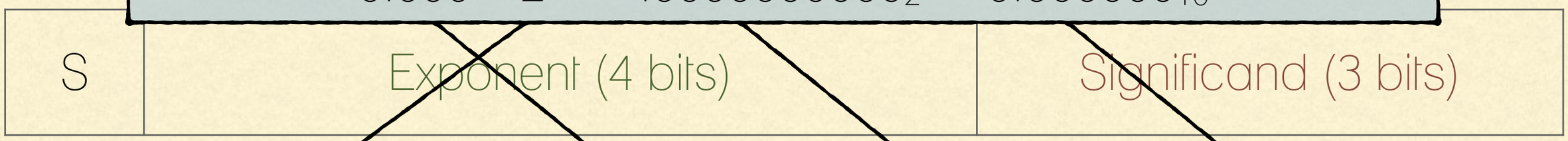
S	Exponent (4 bits)	Significand (3 bits)

Exponent	S (Sign bit)	Significand	Value (Binary)	Value (Decimal)
Value -7	0 Positive	000	.000000000	0.0000000
Value -7	0 Positive	001	.000000001	0.001953125
Value -7	0 Positive	010	.000000010	0.00390625
Value -7	0 Positive	011	.000000011	0.005859375
Value -7	0 Positive	100	.000000100	0.0078125
Value -7	0 Positive	101	.000000101	0.009765625
Value -7	0 Positive	110	.000000110	0.01171875
Value -7	0 Positive	111	.000000111	0.013671875

SIMPLIFIED EXAMPLE

- Nor
- Sim

Now, we DO NOT add the implied 1, we just simply multiply it by 2 (base) raised to the power of the exponent minus 1 (-6):
$0.000 * 2^{-6} = .000000000_2 = 0.000000_{10}$



Exponent	S (Sign bit)	Significand	Value (Binary)	Value (Decimal)
Value -7	0 Positive	000	.000000000	0.0000000
Value -7	0 Positive	001	.000000001	0.001953125
Value -7	0 Positive	010	.000000010	0.00390625
Value -7			11	0.005859375
Value -7			00	0.0078125
Value -7	0 Positive	101	.000000101	0.009765625
Value -7	0 Positive	110	.000000110	0.01171875
Value -7	0 Positive	111	.000000111	0.013671875

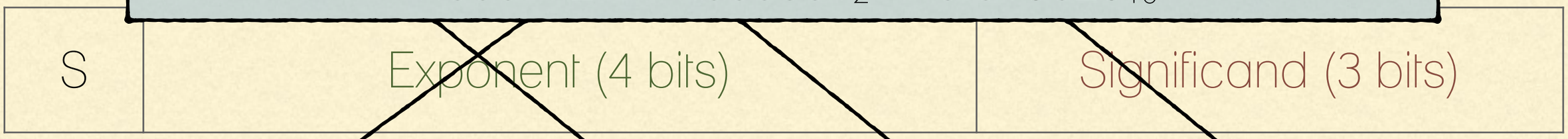
Notice that the difference between consecutive values is roughly .002

SIMPLIFIED EXAMPLE

- Nor
- Sim

Anything other than -7 will be treated as normalized (implied 1) raised to the power of the exponent

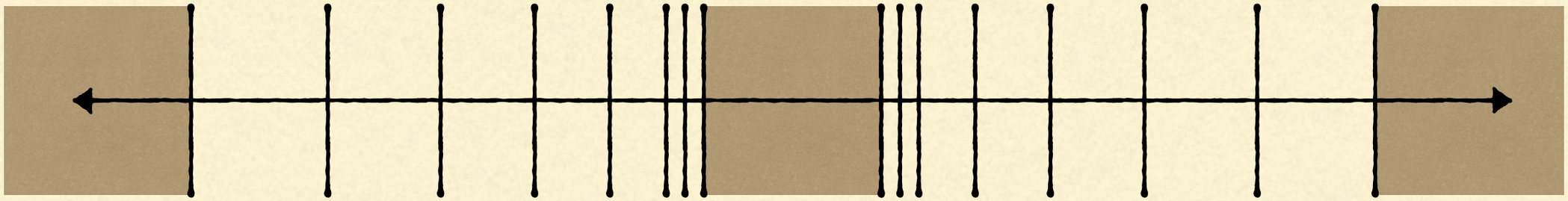
$1.000 * 2^{-6} = .000001_2 = 0.015625_{10}$



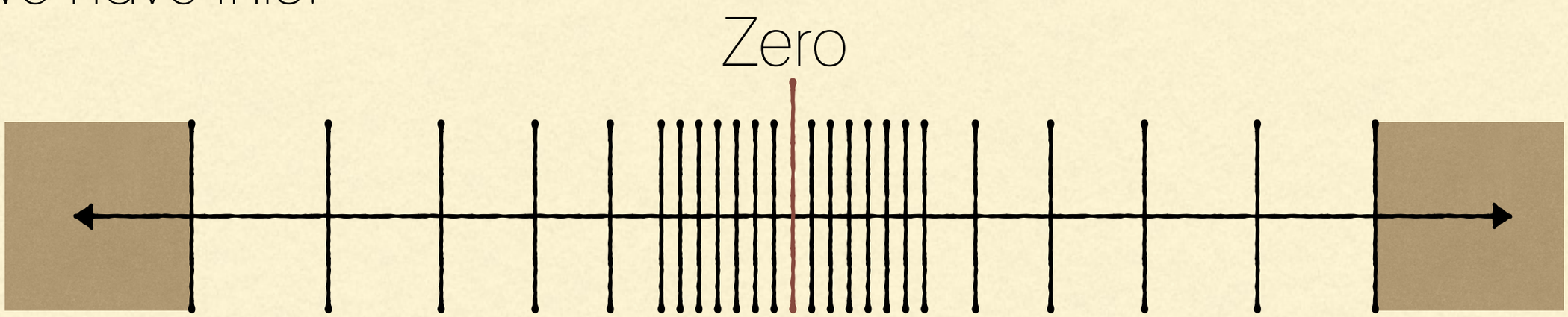
Exponent	S (Sign bit)	Significand	Value (Binary)	Value (Decimal)
Value -6	0 Positive	000	.000001000	0.015625
Value -6	0 Positive	001	.000001001	0.017578125
Value -6	0 Positive	010	.000001010	0.01953125
Value -6	0 Positive	011	.000001011	0.021484375
Value -6	0 Positive	100	.000001100	0.0234375
Value -6	0 Positive	101	.000001101	0.025390625
Value -6	0 Positive	110	.000001110	0.02734375
Value -6	0 Positive	111	.000001111	0.029296875

SIMPLIFIED EXAMPLE

- Now instead of having this chart:



- We have this:



IEEE 754 FLOATING-POINT STANDARD

- Normal format: $+1.xxx\ldots x_2 * 2^{yyy\ldots y}$
- Multiple of Word Size (32 bits)

S	Exponent (8 bits)	Significand (23 bits)

- Significand:
  - To pack more bits, leading 1 implicit for normalized numbers
  - 1 + 23 bits single, 1 + 52 bits double

INTRODUCTION TO BIASED NOTATION

- Here's a chart for 3 bit binary numbers using unsigned representation.

Binary	000	001	010	011	100	101	110	111
Decimal	0	1	2	3	4	5	6	7

- Suppose we wanted to represent negative numbers, but wanted to keep the same ordering, where 000 represents the smallest value and 111 represents the largest value?

INTRODUCTION TO BIASED NOTATION

- This is the idea behind biased representation. The bit-string with N 0's maps to the smallest value, and the bit-string with N 1's maps to the largest value.

Binary	000	001	010	011	100	101	110	111
Decimal	-4	-3	-2	-1	0	1	2	3

- The bias value in this case is 4

INTRODUCTION TO BIASED NOTATION

- How do you convert the following 4-bit biased notation of 3 ‘1110’ to decimal?

INTRODUCTION TO BIASED NOTATION

- How do you convert the following 4-bit biased notation of 3 ‘1110’ to decimal?
  - Convert the binary to unsigned decimal then subtract the bias:
  - 1110 is 14 in decimal, subtract 3 and we have 11

INTRODUCTION TO BIASED NOTATION

- How do you convert the following 4-bit biased notation of 3 ‘1110’ to decimal?
    - Convert the binary to unsigned decimal then subtract the bias:
    - 1110 is 14 in decimal, subtract 3 and we have 11
- How do you convert the decimal 12 to 6-bit biased notation of 5?

INTRODUCTION TO BIASED NOTATION

- How do you convert the following 4-bit biased notation of 3 '1110' to decimal?
    - Convert the binary to unsigned decimal then subtract the bias:
        - 1110 is 14 in decimal, subtract 3 and we have 11
- How do you convert the decimal 12 to 6-bit biased notation of 5?
    - Add the bias to the decimal 12 + 5 = 17, then convert the result to unsigned binary: 10001

FLOATING POINT REPRESENTATION

- Remember that the ‘Exponent’ is a signed number

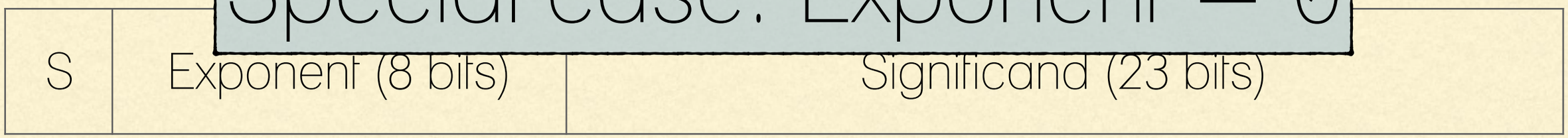
S	Exponent (8 bits)	Significand (23 bits)

- IEEE uses ‘biased’ representation of 127 for single precision. To convert from IEEE floating-point to value:
  - Subtract 127 from exponent field (after converting it to decimal using unsigned binary) to get actual value of exponent
  - Value is $(-1)^{S} * (1 + .\text{Significand}) * 2^{(\text{Exponent} - 127)}$

FLOATING POINT REPRESENTATION

- Remember that the 'Exponent' is a signed number

Special case: Exponent = 0



- IEEE uses 'biased' representation of 127 for single precision. To convert from IEEE floating-point to value:
  - Subtract 127 from exponent field (after converting it to decimal using unsigned binary) to get actual value of exponent
  - Value is $(-1)^{S} * (1 + .\text{Significand}) * 2^{(\text{Exponent} - 127)}$

REPRESENTATION FOR 0 AND INFINITY

Exponent Value	Significand	Result
-127 (all zeros)	0	0
-127 (all zeros)	anything != 0	+/- floating point (denormalized)
between -126 & 127	anything	+/- floating point (normalized)
128 (all ones)	0	+/- Infinity

CONVERTING NORMALIZED FROM BASE 10 TO IEEE 754

- Let's convert 10.25 from base 10 to IEEE 754 single precision. Here's the steps:
    - Convert the number to base 2
        - This results in 1010 + 0.01, which is 1010.01
    - Write this in binary normalized scientific notation
        - $1.01001 * 2^3$
    - Convert 3 to the correct bias: x - 127 must be equal to 3. What's x?

CONVERTING NORMALIZED FROM BASE 10 TO IEEE 754

- Let's convert 10.25 from base 10 to IEEE 754 single precision. Here's the steps:
    - Convert the number to base 2
        - This results in 1010 + 0.01, which is 1010.01
    - Write this in binary normalized scientific notation
        - $1.01001 * 2^3$
    - Convert 3 to the correct bias: x - 127 must be equal to 3. What's x? 130 (goes in exponent field)

CONVERTING NORMALIZED FROM BASE 10 TO IEEE 754

- Let's convert 10.25 from base 10 to IEEE 754 single precision. Here's the steps:
    - Convert the number to base 2
        - This results in 1010 + 0.01, which is 1010.01
    - Write this in binary normalized scientific notation
        - $1.01001 * 2^3$
    - Convert 3 to the correct bias: x - 127 must be equal to 3. What's x? 130 (goes in exponent field)
    - What goes in significand field?

CONVERTING NORMALIZED FROM BASE 10 TO IEEE 754

- Let's convert 10.25 from base 10 to IEEE 754 single precision. Here's the steps:
    - Convert the number to base 2
        - This results in 1010 + 0.01, which is 1010.01
    - Write this in binary normalized scientific notation
        - $1.01001 * 2^3$
    - Convert 3 to the correct bias: x - 127 must be equal to 3. What's x? 130 (goes in exponent field)
    - What goes in significand field? 01001

QUESTION

- After converting 160_{10} to the IEEE 754, what values will you end up with?

A. Sign bit: 0, Exponent: 10100000, Significand: 1000000...
B. Sign bit: 1, Exponent: 10100000, Significand: 1000000...
C. Sign bit: 0, Exponent: 10000110, Significand: 0100000...
D. Sign bit: 0, Exponent: 10000110, Significand: 0010000...

QUESTION

- After converting 160_{10} to the IEEE 754, what values will you end up with?

A. Sign bit: 0, Exponent: 10100000, Significand: 1000000...

B. Sign bit: 1, Exponent: 10100000, Significand: 1000000...

C. Sign bit: 0, Exponent: 10000110, Significand: 0100000...

D. Sign bit: 0, Exponent: 10000110, Significand: 0010000...