



# COMPUTER SYSTEMS FUNDAMENTALS ( 4COSC004W )

Lecture: Week 3. Part 3 of 3

# In this video we will cover:

- Representation of Real values in Binary
  - *Fixed Point representation*
  - *Floating Point representation*

# REAL NUMBERS

Bicimal & IEEE754

# By the end of this unit, you will:

- Understand the representation of Real values in Binary form
  - *Bicimal*
- Be able to represent Decimal real values in Bicimal form
- Be able to represent Bicimal values in Decimal
- Appreciate the limitations of fixed point representations
- Be able to represent Decimal Real values using IEEE754
- Be able to convert from IEEE754 to a real Decimal value

# Real values

- Not all values are Integers
  - 1, 2, 3, 77, ....
- Real (Fractional) values
  - 1.5
  - 1.25
  - 2.75
  - .....

# Bicimal

- Binary format for representing fractional values
  - *Fixed point*

Bicimal					
	$2^{-1}$	$2^{-2}$	$2^{-3}$	$2^{-4}$	
	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	
	0.5	0.25	0.125	0.0625	
•					

# Bicimal 0.5

- Binary format for representing fractional values
  - *Fixed point*

Bicimal 0.5					
	$2^{-1}$	$2^{-2}$	$2^{-3}$	$2^{-4}$	
	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	
	0.5	0.25	0.125	0.0625	

# Bicimal 0.25

- Binary format for representing fractional values
  - *Fixed point*

Bicimal 0.25					
	$2^{-1}$	$2^{-2}$	$2^{-3}$	$2^{-4}$	
	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	
	0.5	0.25	0.125	0.0625	



# Bicimal 0.75

- Binary format for representing fractional values
  - *Fixed point*

Bicimal 0.75					
	$2^{-1}$	$2^{-2}$	$2^{-3}$	$2^{-4}$	
	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	
	0.5	0.25	0.125	0.0625	

# Bicimal 1.625

- Binary format for representing fractional values
  - *Fixed point*

	Bicimal 1.625				
	$2^{-1}$	$2^{-2}$	$2^{-3}$	$2^{-4}$	
	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	
	0.5	0.25	0.125	0.0625	

# Decimal of Bicimal 1.101

	Decimal of 1 . 1 0 1					
1	•	1	0	1		
		$2^{-1}$	$2^{-2}$	$2^{-3}$	$2^{-4}$	
		$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	
1		0.5	0.25	0.125	0.0625	

# Limitations of Bicimal Fixed Point values

- Only positive values
- Not suitable for storing very small or very large real numbers
  - *Avogadro's number*  $6.0221367 \times 10^{+23}$ 
    - Would require about 80 bits for the integer part
  - *Mass of Hydrogen atom*  $1.6733 \times 10^{-24}$ 
    - Would require well over 80 bits for the fractional part
- Hence fixed point format is of limited use for computer representation of different numbers.

# Exact values may require a high resolution:

## ■ With 4 Bcimal Bits:

●	0	0	0	0	= 0
●	0	0	0	1	= 0.0625
●	0	0	1	0	= 0.125
●	0	0	1	1	= 0.1875
●	0	1	0	0	= 0.25
●	0	1	0	1	= 0.3125
●	0	1	1	0	= 0.375
●	0	1	1	1	= 0.4375
●	1	0	0	0	= 0.5
●	1	0	0	1	= 0.5625
●	1	0	1	0	= 0.625
●	1	0	1	1	= 0.6875
●	1	1	0	0	= 0.75
●	1	1	0	1	= 0.8125
●	1	1	1	0	= 0.875
●	1	1	1	1	= 0.9375

# IEEE754

Floating Point representation

# Floating point format

- Very large or very small numbers
- Before IEEE754 standard, different manufacturers used different methods.
- IEEE754 standardised the method of Floating Point representation
- Now adopted by all computer manufacturers
- IEEE754 is simple and efficient method to represent Floating Point format

# A few concepts first:

- Normalized format
  - *Mantissa*
  - *Exponent*



# Normalised Format - Decimal

- 3 parts to a normalised representation:
  - *The integer part (single digit)*
  - *The part beyond the decimal point*
  - *The power part (Exponent)*
- Examples:
  - *10.0 in normalised form is  $1.0 \times 10^1$*
  - *312.0 in normalised form is  $3.12 \times 10^2$*
  - *3.15 in normalised form is  $3.15 \times 10^0$*
  - *0.0004 in normalised form is  $4.0 \times 10^{-4}$*
  - *-400.0 in normalised form is  $-4.0 \times 10^2$*

# Mantissa & Exponent - Decimal

Number	Normalised	Mantissa	Exponent
10	$1.0 \times 10^1$	1.0	1
312	$3.12 \times 10^2$	3.12	2
0.0004	$4.0 \times 10^{-4}$	4.0	-4
3.15	$3.15 \times 10^0$	3.15	0
-400	$-4.0 \times 10^2$	-4.0	2

# Mantissa & Exponent - Decimal

Number
1002
-231
-2
-0.004
-0.12345

# Floating point in Binary

- 0.00001
  - $= 1.0 \times 2^{-5}$ 
    - Mantissa = 1.0
    - Exponent = -5
- -1001.11
  - $= 1.00111 \times 2^{+3}$ 
    - Mantissa = 1.00111
    - Exponent = +3
    - Sign will be dealt with separately

# Converting Decimal 31.75 to Normalised Bicimal Form

Step 1	Convert the integer part (ie. 31) to Binary	11111		
Step 2	Convert the fractional part (ie. 0.75 ) to Bicimal	•	$\frac{1}{2}$	$\frac{1}{4}$
		•	1	1
Step 3	Combine the results from Step 1 and Step 2	11111.11 <sub>2</sub>		
Step 4	Normalise: Move Bicimal point till there is just a single 1 to its left	1.111111 <sub>2</sub> × 2 <sup>+4</sup>		
		Value of Mantissa:	1.111111	
		Value of Exponent:	+4	

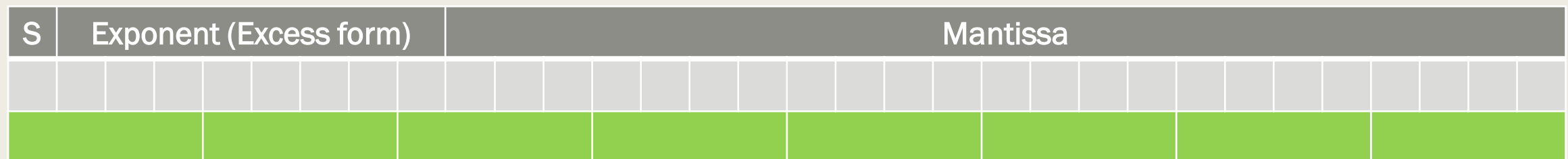
# IEEE754 Format

- Single precision
  - 32 bit in total
  - First bit : Sign Bit
  - Next 8 bits : Exponent (in excess form)
  - Last 23 bits : Mantissa



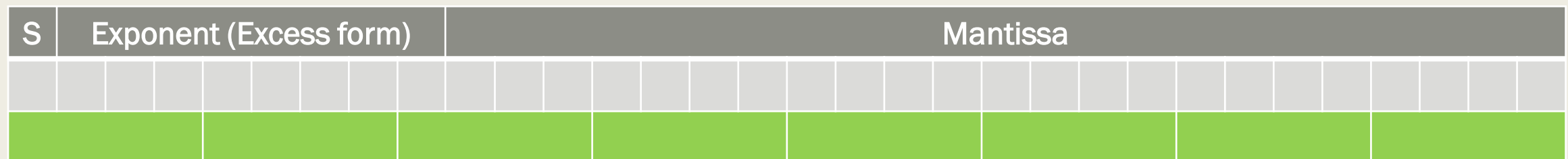
# IEEE754 Format

Step 1:	Original number	$3.25_{10}$
Step 2:	Convert $3_{10}$ to Binary	
Step 3:	Convert $0.25_{10}$ to Binary	
Step 4:	Combine steps 2 & 3	
Step 5:	Normalise the result of step 4	
Step 6:	Mantissa from Step 5	
Step 7:	Exponent from Step 5 in excess form	
	IEEE754 Sign Bit (0 Positive, 1 Negative)	
	IEEE754 Exponent Bits	
	IEEE754 Mantissa Bits	



# IEEE754 Format

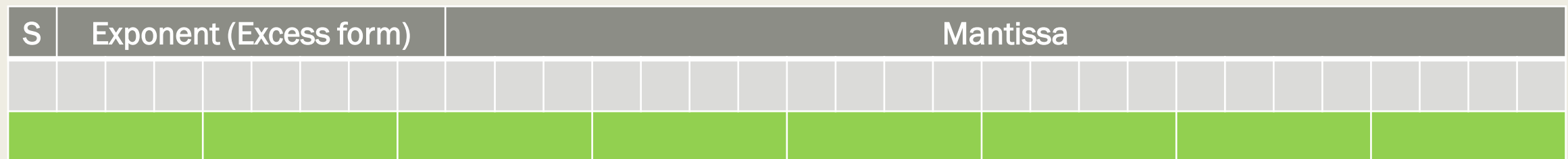
Step 1:	Original number	$3.25_{10}$
Step 2:	Convert $3_{10}$ to Binary	$11_2$
Step 3:	Convert $0.25_{10}$ to Binary	
Step 4:	Combine steps 2 & 3	
Step 5:	Normalise the result of step 4	
Step 6:	Mantissa from Step 5	
Step 7:	Exponent from Step 5 in excess form	
	IEEE754 Sign Bit (0 Positive, 1 Negative)	
	IEEE754 Exponent Bits	
	IEEE754 Mantissa Bits	





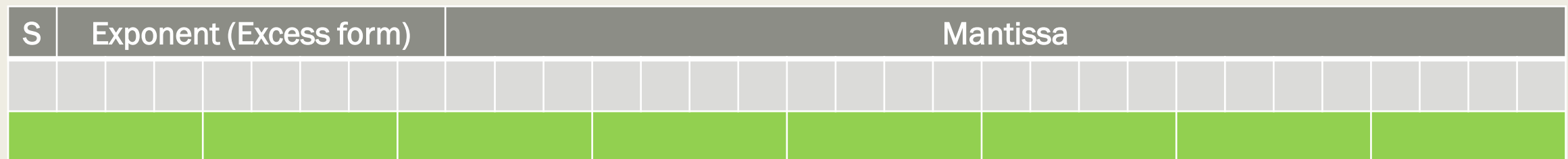
# IEEE754 Format

Step 1:	Original number	$3.25_{10}$
Step 2:	Convert $3_{10}$ to Binary	$11_2$
Step 3:	Convert $0.25_{10}$ to Binary	$0.01_2$
Step 4:	Combine steps 2 & 3	
Step 5:	Normalise the result of step 4	
Step 6:	Mantissa from Step 5	
Step 7:	Exponent from Step 5 in excess form	
	IEEE754 Sign Bit (0 Positive, 1 Negative)	
	IEEE754 Exponent Bits	
	IEEE754 Mantissa Bits	



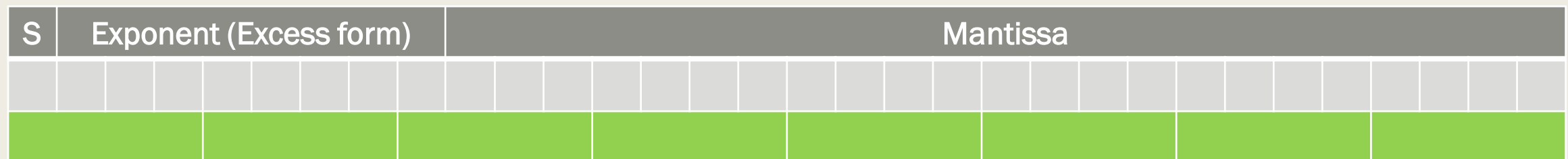
# IEEE754 Format

Step 1:	Original number	$3.25_{10}$
Step 2:	Convert $3_{10}$ to Binary	$11_2$
Step 3:	Convert $0.25_{10}$ to Binary	$0.01_2$
Step 4:	Combine steps 2 & 3	$11.01_2$
Step 5:	Normalise the result of step 4	
Step 6:	Mantissa from Step 5	
Step 7:	Exponent from Step 5 in excess form	
	IEEE754 Sign Bit (0 Positive, 1 Negative)	
	IEEE754 Exponent Bits	
	IEEE754 Mantissa Bits	



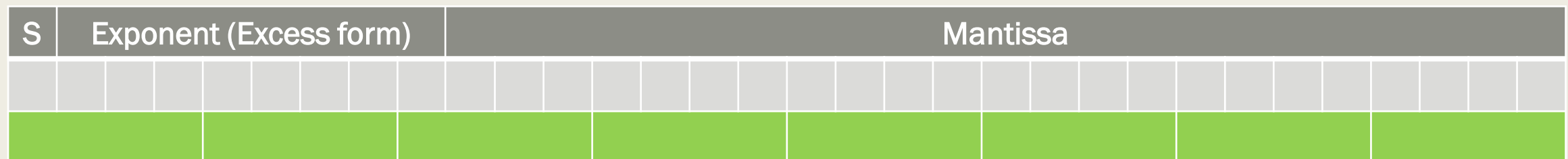
# IEEE754 Format

Step 1:	Original number	$3.25_{10}$
Step 2:	Convert $3_{10}$ to Binary	$11_2$
Step 3:	Convert $0.25_{10}$ to Binary	$0.01_2$
Step 4:	Combine steps 2 & 3	$11.01_2$
Step 5:	Normalise the result of step 4	$1.101 \times 2^1$
Step 6:	Mantissa from Step 5	
Step 7:	Exponent from Step 5 in excess form	
	IEEE754 Sign Bit (0 Positive, 1 Negative)	
	IEEE754 Exponent Bits	
	IEEE754 Mantissa Bits	



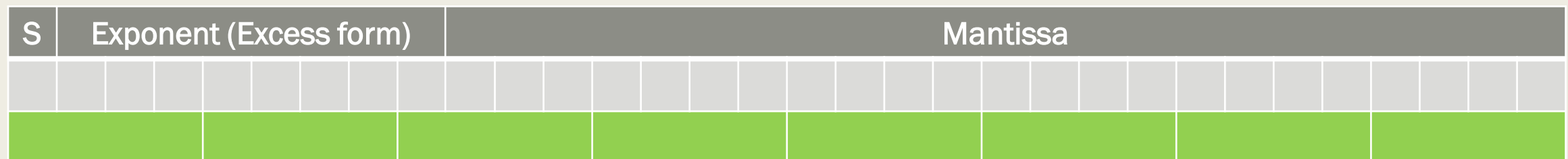
# IEEE754 Format

Step 1:	Original number	$3.25_{10}$
Step 2:	Convert $3_{10}$ to Binary	$11_2$
Step 3:	Convert $0.25_{10}$ to Binary	$0.01_2$
Step 4:	Combine steps 2 & 3	$11.01_2$
Step 5:	Normalise the result of step 4	$1.101 \times 2^1$
Step 6:	Mantissa from Step 5	$1.101$
Step 7:	Exponent from Step 5 in excess form	
	IEEE754 Sign Bit (0 Positive, 1 Negative)	
	IEEE754 Exponent Bits	
	IEEE754 Mantissa Bits	



# IEEE754 Format

Step 1:	Original number	$3.25_{10}$
Step 2:	Convert $3_{10}$ to Binary	$11_2$
Step 3:	Convert $0.25_{10}$ to Binary	$0.01_2$
Step 4:	Combine steps 2 & 3	$11.01_2$
Step 5:	Normalise the result of step 4	$1.101 \times 2^1$
Step 6:	Mantissa from Step 5	$1.101$
Step 7:	Exponent from Step 5 in excess form	$1 + 127 = 128 = 10000000_2$
	IEEE754 Sign Bit (0 Positive, 1 Negative)	
	IEEE754 Exponent Bits	
	IEEE754 Mantissa Bits	



# IEEE754 Format

Step 1:	Original number	$3.25_{10}$
Step 2:	Convert $3_{10}$ to Binary	$11_2$
Step 3:	Convert $0.25_{10}$ to Binary	$0.01_2$
Step 4:	Combine steps 2 & 3	$11.01_2$
Step 5:	Normalise the result of step 4	$1.101 \times 2^1$
Step 6:	Mantissa from Step 5	$1.101$
Step 7:	Exponent from Step 5 in excess form	$1 + 127 = 128 = 10000000_2$
	IEEE754 Sign Bit (0 Positive, 1 Negative)	0
	IEEE754 Exponent Bits	
	IEEE754 Mantissa Bits	

S	Exponent (Excess form)								Mantissa																							
0																																

# IEEE754 Format

Step 1:	Original number	$3.25_{10}$
Step 2:	Convert $3_{10}$ to Binary	$11_2$
Step 3:	Convert $0.25_{10}$ to Binary	$0.01_2$
Step 4:	Combine steps 2 & 3	$11.01_2$
Step 5:	Normalise the result of step 4	$1.101 \times 2^1$
Step 6:	Mantissa from Step 5	$1.101$
Step 7:	Exponent from Step 5 in excess form	$1 + 127 = 128 = 10000000_2$
	IEEE754 Sign Bit (0 Positive, 1 Negative)	0
	IEEE754 Exponent Bits	10000000
	IEEE754 Mantissa Bits	

S	Exponent (Excess form)								Mantissa																			
0	1	0	0	0	0	0	0	0																				

# IEEE754 Format

Step 1:	Original number	$3.25_{10}$
Step 2:	Convert $3_{10}$ to Binary	$11_2$
Step 3:	Convert $0.25_{10}$ to Binary	$0.01_2$
Step 4:	Combine steps 2 & 3	$11.01_2$
Step 5:	Normalise the result of step 4	$1.101 \times 2^1$
Step 6:	Mantissa from Step 5	$1.101$
Step 7:	Exponent from Step 5 in excess form	$1 + 127 = 128 = 10000000_2$
	IEEE754 Sign Bit (0 Positive, 1 Negative)	0
	IEEE754 Exponent Bits	10000000
	IEEE754 Mantissa Bits	1.101

S	Exponent (Excess form)								Mantissa																			
0	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



Denary	Binary				Hexadecimal
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	2
3	0	0	1	1	3
4	0	1	0	0	4
5	0	1	0	1	5
6	0	1	1	0	6
7	0	1	1	1	7
8	1	0	0	0	8
9	1	0	0	1	9
10	1	0	1	0	A
11	1	0	1	1	B
12	1	1	0	0	C
13	1	1	0	1	D
14	1	1	1	0	E
15	1	1	1	1	F

# IEEE754 Format

Step 1:	Original number	$3.25_{10}$
Step 2:	Convert $3_{10}$ to Binary	$11_2$
Step 3:	Convert $0.25_{10}$ to Binary	$0.01_2$
Step 4:	Combine steps 2 & 3	$11.01_2$
Step 5:	Normalise the result of step 4	$1.101 \times 2^1$
Step 6:	Mantissa from Step 5	$1.101$
Step 7:	Exponent from Step 5 in excess form	$1 + 127 = 128 = 10000000_2$
	IEEE754 Sign Bit (0 Positive, 1 Negative)	0
	IEEE754 Exponent Bits	10000000
	IEEE754 Mantissa Bits	1.101

S	Exponent (Excess form)								Mantissa																						
0	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
4				0				5				0				0				0				0				0			

# IEEE754 Format

S	Exponent (Excess form)								Mantissa																						
4				0				5				0				0				0				0				0			
0	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

$$= 128_{10}$$

$$128 - 127 = +1$$

Exponent is: +1

Mantissa: 1.101<sub>2</sub>

$$1.101_2 \times 2^{+1} = 11.01_2$$

+

$$3.25_{10}$$

S	Exponent (Excess form)								Mantissa																						
0	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
4				0				5				0				0				0				0				0			

# IEEE754 Format

Step 1:	Original number	$-0.125_{10}$
Step 2:	Convert $0_{10}$ to Binary	$0_2$
Step 3:	Convert $0.125_{10}$ to Binary	$0.001_2$
Step 4:	Combine steps 2 & 3	$0.001_2$
Step 5:	Normalise the result of step 4	$1.0 \times 2^{-3}$
Step 6:	Mantissa from Step 5	$1.0$
Step 7:	Exponent from Step 5 in excess form	$-3 + 127 = 124 = 01111100_2$
	IEEE754 Sign Bit (0 Positive, 1 Negative)	$1$
	IEEE754 Exponent Bits	$01111100$
	IEEE754 Mantissa Bits	$1.0$

S	Exponent (Excess form)								Mantissa																						
1	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
B				E				0				0				0				0				0				0			

# IEEE754 Format

S	Exponent (Excess form)							Mantissa																											
B				E				0				0				0				0				0				0							
1	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						

$$= 124_{10}$$

$$124 - 127 = -3$$

Exponent is: -3

Mantissa: 1.0<sub>2</sub>

$$1.0_2 \times 2^{-3} = 0.001_2$$

-

$$0.125_{10}$$

S	Exponent (Excess form)								Mantissa																							
1	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
B				E				0				0				0				0				0				0				

# IEEE754 Format

Step 1:	Original number	$-195_{10}$
Step 2:	Convert $195_{10}$ to Binary	$11000011_2$
Step 3:	Convert $0.0_{10}$ to Binary	$0.0_2$
Step 4:	Combine steps 2 & 3	$11000011_2$
Step 5:	Normalise the result of step 4	$1.1000011 \times 2^{+7}$
Step 6:	Mantissa from Step 5	$1.1000011$
Step 7:	Exponent from Step 5 in excess form	$7 + 127 = 134 = 10000110_2$
	IEEE754 Sign Bit (0 Positive, 1 Negative)	1
	IEEE754 Exponent Bits	10000110
	IEEE754 Mantissa Bits	1.1000011

S	Exponent (Excess form)								Mantissa																						
1	1	0	0	0	0	1	1	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
C				3				4				3				0				0				0				0			

# IEEE754 Format

S	Exponent (Excess form)							Mantissa																											
C				3				4				3				0				0				0				0							
1	1	0	0	0	0	1	1	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					

$$= 134_{10}$$

$$134 - 127 = +7$$

Exponent is: +7

Mantissa: 1.1000011<sub>2</sub>

$$1.1000011_2 \times 2^7 = 11000011_2$$

- 195<sub>10</sub>

S	Exponent (Excess form)								Mantissa																							
1	1	0	0	0	0	1	1	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	C				3				4				3				0				0				0				0			

# IEEE754

- Further examples in tutorial
  - *Try out random numbers of your choice*
- This module will not cover:
  - *Double precision (64-bit)*
  - *Zero*



# For this module, we will only consider 4 Bicimal Bits:

●	0	0	0	0	= 0
●	0	0	0	1	= 0.0625
●	0	0	1	0	= 0.125
●	0	0	1	1	= 0.1875
●	0	1	0	0	= 0.25
●	0	1	0	1	= 0.3125
●	0	1	1	0	= 0.375
●	0	1	1	1	= 0.4375
●	1	0	0	0	= 0.5
●	1	0	0	1	= 0.5625
●	1	0	1	0	= 0.625
●	1	0	1	1	= 0.6875
●	1	1	0	0	= 0.75
●	1	1	0	1	= 0.8125
●	1	1	1	0	= 0.875
●	1	1	1	1	= 0.9375

# In this video we looked at:

- Real numbers
  - *Fixed point ( Bcimal )*
  - *Floating point ( IEEE754 )*

# Further reading:

- Computer Systems
  - 3.5

© The University of Westminster (2020)

The right of Noam Weingarten to be identified as author of this work has been asserted by them in accordance with the Copyright, Designs and Patents Act 1988