FLOATING POINT NUMBERS

Englander Ch. 5

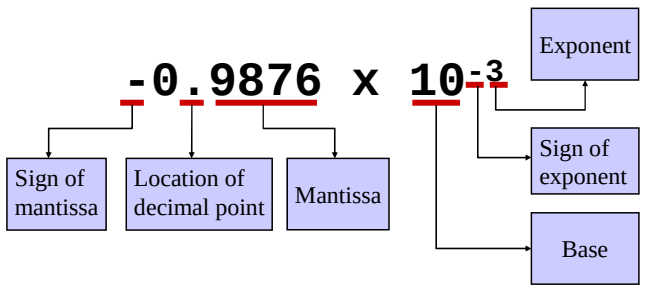
Exponential Notation

- The following are equivalent representations of 1,234

```
123,400.0    x 10^-2
 12,340.0    x 10^-1
  1,234.0    x 10^0
    123.4    x 10^1
     12.34   x 10^2
      1.234  x 10^3
      0.1234 x 10^4
```

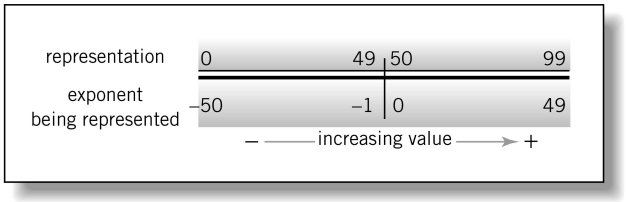
The representations differ in that the decimal place – the "point" -- "floats" to the left or right (with the appropriate adjustment in the exponent).

Parts of a Floating Point Number



Exponent Excess 50 Representation

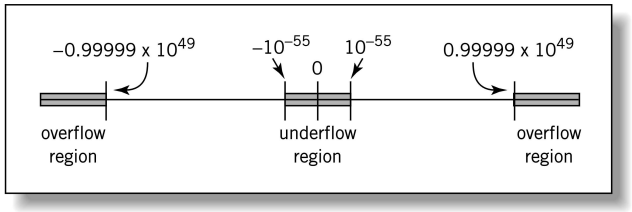
- With 2 digits for the exponent and 5 for the mantissa: from .00001 x 10^{-50} to .99999 x 10^{49}



Englander: The Architecture of Computer Hardware and Systems Software, 2nd edition
Chapter 5, Figure 05-01

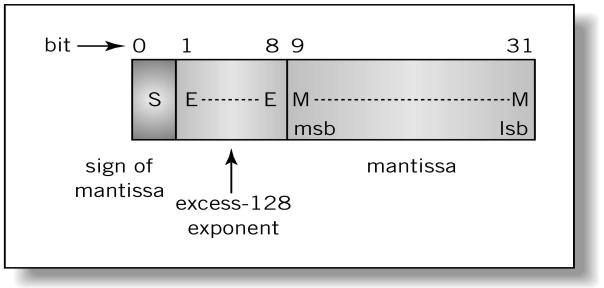
Overflows / Underflows

- From .00001 x 10^{-50} to .99999 x 10^{49}
  1 x 10^{-55} to .99999 x 10^{49}



Englander: The Architecture of Computer
Hardware and Systems Software, 2nd edition
Chapter 5, Figure 05-02

Typical Floating Point Format

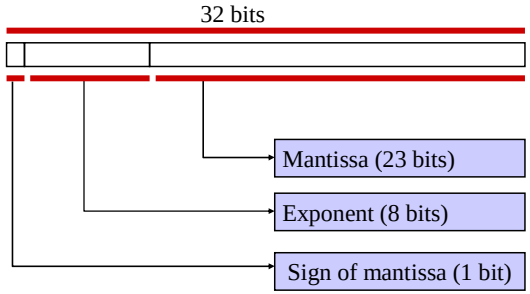


Englander: The Architecture of Computer
Hardware and Systems Software, 2nd edition
Chapter 5, Figure 05-04

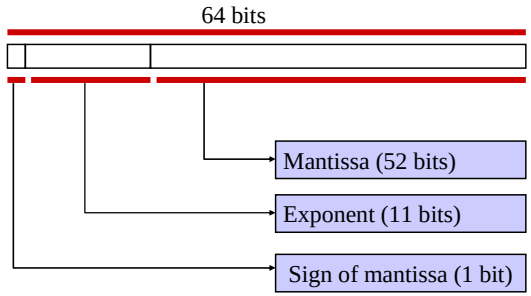
IEEE 754 Standard

- Most common standard for representing floating point numbers
- **Single precision**: 32 bits, consisting of...
  - Sign bit (1 bit)
  - Exponent (8 bits)
  - Mantissa (23 bits)
- **Double precision**: 64 bits, consisting of…
  - Sign bit (1 bit)
  - Exponent (11 bits)
  - Mantissa (52 bits)

Single Precision Format



Double Precision Format



Normalization

- The mantissa is ***normalized***
- Has an implied decimal place on left
- Has an implied "1" on left of the decimal place
- E.g.,
  - Mantissa: `10100000000000000000000`
  - Representation: $1.101_2 = 1.625_{10}$

Excess Notation

- To include both positive and negative exponents, "excess-*n*" notation is used
- Single precision: excess 127
- Double precision: excess 1023
- The value of the exponent stored is *n* larger than the actual exponent
- E.g., – excess 127, **`10000111`**
  - Exponent: **`135 – 127 = 8 (value)`**
  - Representation:

Excess Notation
- Sample -

Represent exponent of 14_{10} in excess 127 form:

$\mathbf{127_{10}}$	=	+ $\mathbf{01111111_2}$
$\mathbf{14_{10}}$	=	+ $\mathbf{\underline{00001110}_2}$
Representation	=	$\mathbf{10001101_2}$

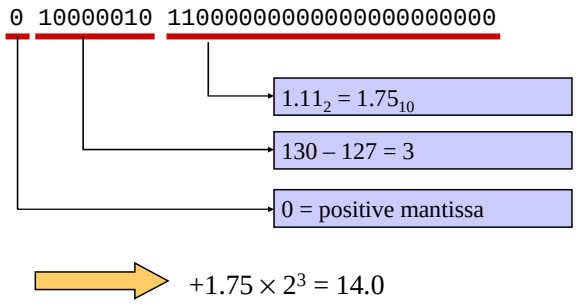
Excess Notation
- Sample -

Represent exponent of -8_{10} in excess 127 form:

$\mathbf{127_{10}}$	=	+ $\mathbf{01111111_2}$
- $\mathbf{8_{10}}$	=	- $\mathbf{00001000_2}$
Representation	=	$\mathbf{01110111_2}$

Example

- Single precision



Exercise – Floating Point Conversion (1)

- What decimal value is represented by the following 32-bit floating point number?

```
1 10000010 11110110000000000000000
```

- Answer:_____

Skip answer | Answer

Exercise – Floating Point Conversion (1)

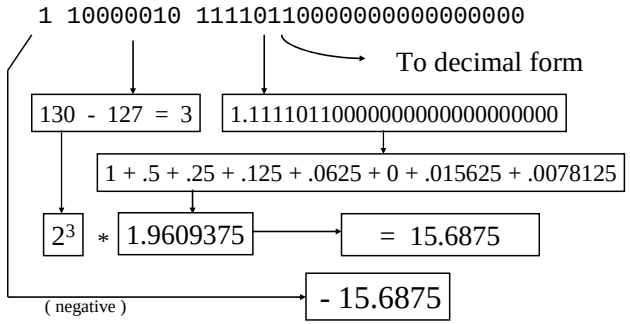
Answer

- What decimal value is represented by the following 32-bit floating point number?

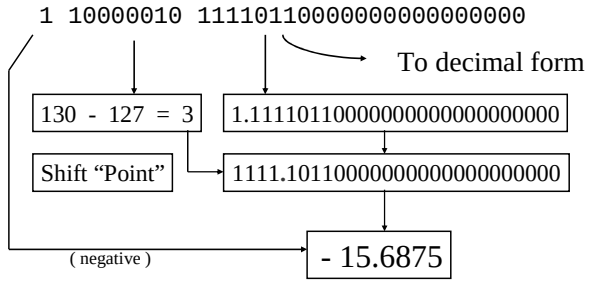
```
1 10000010 11110110000000000000000
```

- Answer: -15.6875

Step by Step Solution



Step by Step Solution : Alternative Method



Exercise – Floating Point Conversion (2)

- Express 3.14 as a 32-bit floating point number
  - Answer:_____
  - (Note: only use 10 significant bits for the mantissa)

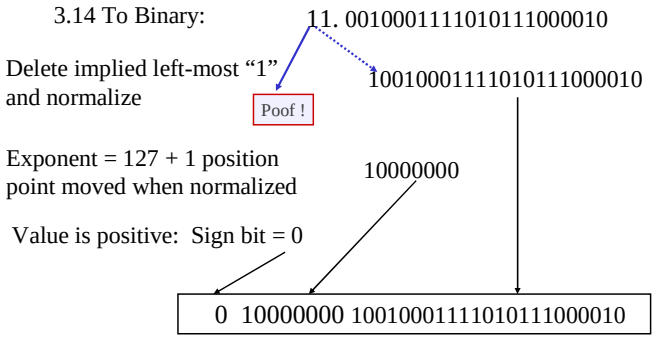
Skip answer | Answer

Exercise – Floating Point Conversion (2)

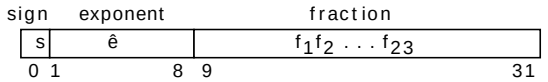
Answer

- Express 3.14 as a 32-bit floating point number
  - Answer: 0 10000000 10010001111010111000010

Detail Solution : 3.14 to IEEE Simple Precision



IEEE Single-Precision Floating Point Format



ê	e	Value	Type
255	none	none	Infinity or NaN
254	127	$(-1)^s \times (1.f_1f_2...) \times 2^{127}$	Normalized
...	...	...	...
2	-125	$(-1)^s \times (1.f_1f_2...) \times 2^{-125}$	Normalized
1	-126	$(-1)^s \times (1.f_1f_2...) \times 2^{-126}$	Normalized
0	-126	$(-1)^s \times (0.f_1f_2...) \times 2^{-126}$	Denormalized

- Exponent bias is 127 for normalized #s

Decimal Floating-Point Add and Subtract Examples

Operands	Alignment	Normalize & round
6.144×10^2	0.06144×10^4	1.003644×10^5
$+9.975 \times 10^4$	$+9.975 \times 10^4$	$+ .0005 \times 10^5$
	10.03644×10^4	1.004×10^5

Operands	Alignment	Normalize & round
1.076×10^{-7}	1.076×10^{-7}	7.7300×10^{-9}
-9.987×10^{-8}	-0.9987×10^{-7}	$+ .0005 \times 10^{-9}$
	0.0773×10^{-7}	7.730×10^{-9}

Floating Point Calculations: Addition

- Numbers must be aligned: have the same exponent (the larger one, to protect precision)
- Add mantissas. If overflow, adjust the exponent
- Ex. 0 51 99718 (e = 1) and 0 49 67000 (e = -1)
  - Align numbers: 0 51 99718
    0 51 00670
  - Add them:
    99718
    + 00670
    1 00388 ← Overflow
  - Round the number and adjust exponent: 0 52 10039

Floating Point Calculations: Multiplication

- (a * 10^e) * (b * 10^f) = a * b * 10^(e+f)
- Rule: multiply mantissas; add exponents
  But: (n + e) + (n + f) = 2 * (n + e + f)
  ➔ Must subtract excess n from result

- Ex. 0 51 99718 (e = 1) and 0 49 67000 (e = -1)
  Mantissas: .99718 * .67000 = 0.6681106
  Exponents: 51 + 49 = 100 and 100 – 50 = 50
  Normalize: .6681106 → .66811

  Final result: .66811 * 10 (since 50 means e = 0)