# ΗΡΥ 201– Ψηφιακοί Υπολογιστές

## Γ. Παπαευσταθίου

# Floating Point Numbers
# Examples

# 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
```

$123{,}400.0 \times 10^{-2}$

$12{,}340.0 \times 10^{-1}$

$\boxed{1{,}234.0 \times 10^{0}}$

$123.4 \times 10^{1}$

$12.34 \times 10^{2}$

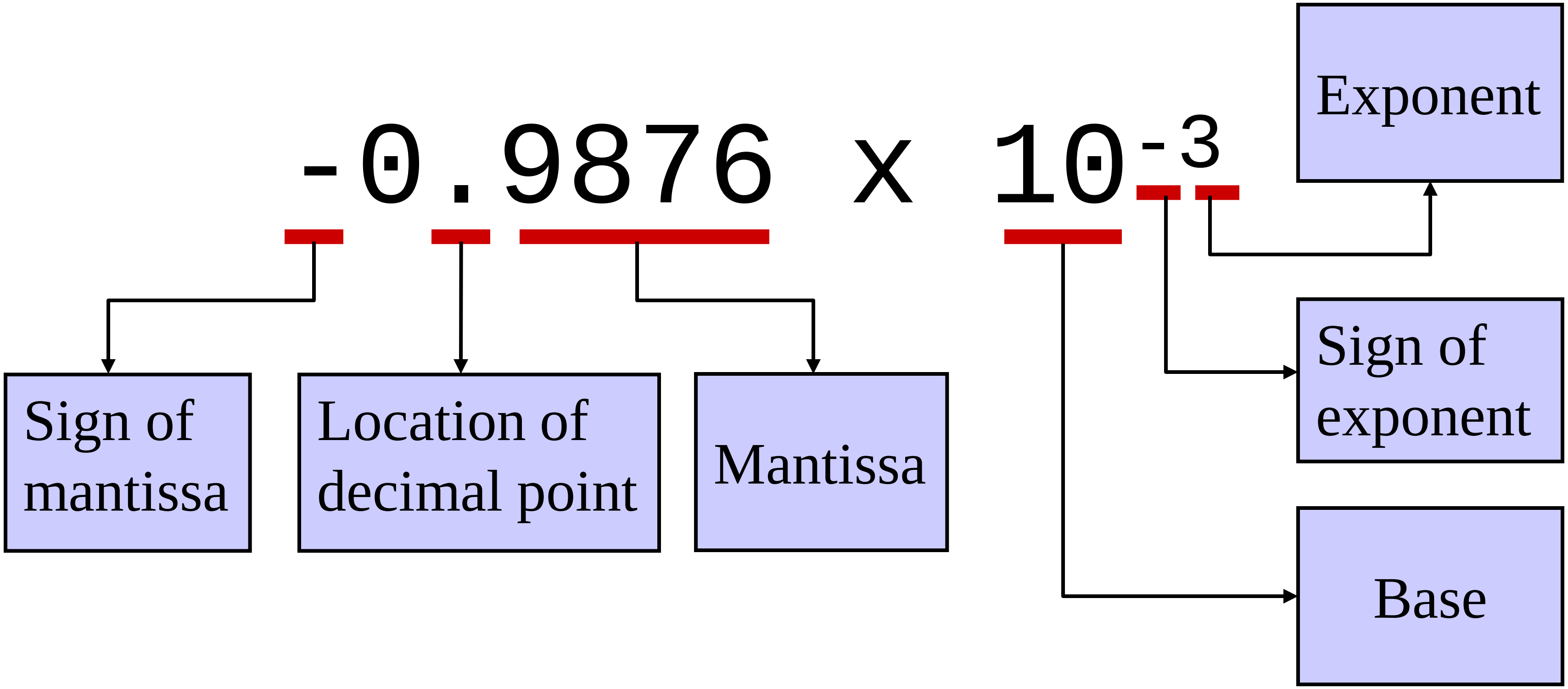
$1.234 \times 10^{3}$

$0.1234 \times 10^{4}$

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

p. 122

# Parts of a Floating Point Number




p. 123

# 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)

# Representation

- Sign + exponent + coefficient

| | | |
|---|---|---|
| S | Exp | Coefficient |

  - 1 + 8 + 23 = 32 bits
  - 1+ 11 + 52 = 64 bits (double precision)

# The sign bit

- 0 indicates a positive number
- 1 a negative number

# The exponent (I)

  - 8 bits for single precision
  - 11 bits for double precision
- With 8 bits, we can represent exponents between -126 and + 127
  - All-zeroes value is reserved for the zeroes and denormalized numbers
  - All-ones value are reserved for the infinities and NaNs (Not a Number)

# The exponent (II)

- Exponents are represented using a biased notation
  - **Stored value = actual exponent + bias**
- For 8 bit exponents, bias is 127
  - Stored value of 1 corresponds to –126
  - Stored value of 254 corresponds to +127

0 and 255 are reserved for special values

# The exponent (III)

- Biased notation simplifies comparisons:
  - If two normalized floating point numbers have different exponents, the one with the  bigger exponent is the bigger of the two

# Special values (I)

- ***<u>Signed zeroes:</u>***
  - IEEE 754 distinguishes between +0 and –0
  - Represented by
    - Sign bit: 0 or 1
    - Biased exponent: all zeroes
    - Coefficient: all zeroes

# Special values (II)

- ***<u>Denormalized numbers:</u>***
  - Numbers whose coefficient cannot be normalized
    - *Smaller than* $\mathbf{2^{-126}}$
  - Will have a coefficient with leading zeroes and exponent field equal to zero
    - Reduces the number of significant digits
    - Lowers accuracy

# Special values (III)

- ***<u>Infinities:</u>***
  - $+\infty$ and $-\infty$
  - Represented by
    - Sign bit: 0 or 1
    - Biased exponent: all ones
    - Coefficient: all zeroes

# Special values (IV)

- ***<u>NaN:</u>***
  - For ***Not a Number***
  - Often result from illegal divisions:
    **0/0, ∞/∞, ∞/–∞, –∞/∞, and –∞/–∞**
  - Represented by
    - Sign bit: 0 or 1
    - Biased exponent: all ones
    - Coefficient: non zero

# The coefficient

- Also known as ***fraction*** or ***significand***
- ***Most significant bit is always one***
  - Implicit and not ***represented***

    - Biased exponent is $\mathbf{127}_{\mathbf{\textit{ten}}}$
    - True coefficient is ***implicit one*** followed by all zeroes

| | | |
|---|---|---|
| 0 | 01…1 | 00000000000000000000000 |

# Decoding a floating point number

- Sign indicated by first bit
- Subtract 127 from biased exponent to obtain power of two:

  **<be> – 127**
- Use coefficient to construct a normalized binary value with a binary point:

  **1.<coefficient>**
- Number being represented is

  $\mathbf{1.\langle coefficient\rangle \times 2^{\langle be\rangle - 127}}$

# First example

| | | |
|---|---|---|
| 0 | 01…1 | 00000000000000000000000000000 |

- Sign bit is zero:

  **Number is positive**
- Biased exponent is 127

  **Power of two is zero**
- Normalized binary value is

  **1.0000000**
- Number is $\mathbf{1 \times 2^0 = 1}$

# Second example

| | | |
|---|---|---|
| 0 | 10…0 | 10000000000000000000000000000 |

- Sign bit is zero:

  **Number is positive**

- Biased exponent is 128

  **Power of two is 1**

- Normalized binary value is

  **1.1000000**

- Number is $\mathbf{1.1 \times 2^1 = 11 = 3_{ten}}$

# Third example

| | | |
|---|---|---|
| 1 | 01…10 | 11000000000000000000000000000 |

- Sign bit is 1:

  **Number is negative**
- Biased exponent is 126

  **Power of two is** –1
- Normalized binary value is

  **1.1100000**
- Number is –$\mathbf{1.11 \times 2}^{-1}$ **= –0.111 = –7/8**$_{\mathbf{ten}}$

# Can we do it now?

| | | |
|---|---|---|
| 0 | $129_{ten}$ | 10100000000000000000000000000000 |

- Sign bit is 0:

  **Number is ____________**
- Biased exponent is 129

  **Power of two is _______**
- Normalized binary value is

  **1.___________**
- Number is ___________________________

# Encoding a floating point number

- Use sign to pick sign bit
- Normalize the number:
  Convert it to form **1.<*more bits*>** $\times$ **2**$^{<exp>}$
- Add **127** to exponent **<*exp*>** to obtain
  **biased exponent <*be*>**
- Coefficient **<coeff>** is equal to fractional part **<*more bits*>** of number

# First example

- Represent **7**:
  - **Convert to binary: 111**
  - **Normalize: $1.11 \times 2^2$**
  - **Sign bit is 0**
  - **Biased exponent is 127 + 2 = $10000001_{two}$**
  - **Coefficient is 1100…0**

| | | |
|---|---|---|
| 0 | 10…01 | 11000000000000000000000000000 |

# Second example

- Represent **1/2**
  - **Convert to binary: 0.1**
  - **Normalize: $1.0 \times 2^{-1}$**
  - **Sign bit is 0**
  - **Biased exponent is 127 – 1 = $01111110_{two}$**
  - **Coefficient is 00…0**

| | | |
|---|---|---|
| 0 | 01…10 | 00000000000000000000000000000 |

# Third example

- Represent **–2**
  - **Convert to binary: 10**
  - **Normalize: $1.0 \times 2^1$**
  - **Sign bit is 1**
  - **Biased exponent is 127 + 1 = $10000000_{two}$**
  - **Coefficient is 00…0**

| | | |
|---|---|---|
| 1 | 10…00 | 00000000000000000000000000000 |

# Fourth example

- Represent **9/4**
  - **Convert to binary:** $\mathbf{1001 \times 2^{-2}}$
  - **Normalize:** $\mathbf{1.001 \times 2^{1}}$
  - **Sign bit is 0**
  - **Biased exponent is 127 + 1 = $\mathbf{10000000_{two}}$**
  - **Coefficient is 0010…0**

| | | |
|---|---|---|
| 1 | 10…00 | 00100000000000000000000000000 |

# Can we do it now?

- Represent **6.25**:
  - **Convert to binary:** ________
  - **Normalize:** **1.______ × 2______**
  - **Sign bit is ______**
  - **Biased exponent is 127 + ___ = ______ten**
  - **Coefficient is__________**

# Range

- Can represent numbers between
  **$1.00\ldots0 \times 2^{-126}$** and **$1.11\ldots1 \times 2^{127}$**
  - Say between **$2^{-126}$** and **$2^{128}$**
- **Observing that $2^{10} \approx 10^3$**
  we divide the exponents by 10 and multiply them by 3 to obtain the interval expressed in powers of 10
  - Approximate range is **$10^{-38}$** to **$10^{38}$**

# More Examples

- Single precision

```
0 10000010 11000000000000000000000
```

$1.11_2$

$130 - 127 = 3$

0 = positive mantissa

$$\Rightarrow \quad +1.11_2 \times 2^3 = 1110.0_2 = 14.0_{10}$$

# Hexadecimal

- It is convenient and common to represent the original floating point number in hexadecimal
- The preceding example…

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 100 | 0001 | 0 110 | 0000 | 0000 | 0000 | 0000 | 0000 |
| 4 | 1 | 6 | 0 | 0 | 0 | 0 | 0 |

# Go all the way again

- E.g., What decimal value is represented by the following 32-bit floating point number?

$$\mathtt{C17B0000}_{16}$$

- Step 1
  - Express in binary and find S, E, and M

$\texttt{C17B0000}_{16} =$

$\underbrace{\texttt{1}}_{S}\ \underbrace{\texttt{10000010}}_{E}\ \underbrace{\texttt{11110110000000000000000}}_{M}{}_{2}$

1 = negative

0 = positive

- Step 2
  - Find "real" exponent, *n*
  - $n = E - 127$

    $= 10000010_2 - 127$

    $= 130 - 127$

    $= 3$

- Step 3
  - Put S, M, and *n* together to form binary result
  - (Don’t forget the implied “1.” on the left of the mantissa.)

$$-1.1111011_2 \text{ x } 2^{n} =$$

$$-1.1111011_2 \text{ x } 2^{3} =$$

$$-1111.1011_2$$

- Step 4
  – Express result in decimal

$$-1111.1011_2$$

-15

$2^{-1} = 0.5$

$2^{-3} = 0.125$

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

0.6875

Answer: -15.6875

# Converting <u>to</u> Floating Point

- E.g., Express $36.5625_{10}$ as a 32-bit floating point number (in hexadecimal)

- Step 1
  - Express original value in binary

$$36.5625_{10} =$$

$$100100.1001_2$$

- Step 2
  - Normalize

$$100100.1001_2 =$$

$$1.001001001_2 \times 2^5$$

- Step 3
  - Determine S, E, and M

$$\underbrace{+}_{S}1.\underbrace{001001001}_{M}{}_2 \times 2^{\underbrace{5}_{n}}$$

E $= n + 127$

$= 5 + 127$

$= 132$

$= 10000100_2$

S = 0 (because the value is positive)

- Step 4
  - Put S, E, and M together to form 32-bit binary result

$$\underbrace{0}_{S}\ \underbrace{\texttt{10000100}}_{E}\ \underbrace{\texttt{00100100100000000000000}}_{M}{}_2$$

- Step 5
  - Express in hexadecimal

$$0\ 10000100\ 00100100100000000000000_2 =$$

$$0100\ 0010\ 0001\ 0010\ 0100\ 0000\ 0000\ 0000_2 =$$

$$4\quad 2\quad 1\quad 2\quad 4\quad 0\quad 0\quad 0_{16}$$

Answer: $42124000_{16}$

# Double precision arithmetic (I)

- Use 64-bit double words
- Allows us to have
  - One bit for sign
  - Eleven bits for exponent
    - 2,048 possible values
  - Fifty-two bits for coefficient
    - Plus the implicit leading bit

# Double precision arithmetic (II)

- Exponents are still represented using a biased notation
  - **Stored value = actual exponent + bias**
- For 11-bit exponents, bias is **1023**
  - Stored value of 1 corresponds to **–1,022**
  - Stored value of 2,046 corresponds to **+1,023**
  - Stored values of 0 and 2,047 are reserved for special cases

# Double precision arithmetic (III)

- Can now represent numbers between **$1.00\ldots0 \times 2^{-1,022}$** and **$1.11\ldots1 \times 2^{1,203}$**
  - Say between **$2^{-1,022}$** and **$2^{1,204}$**
  - Approximate range is **$10^{-307}$** to **$10^{307}$**
    - In reality, more like **$10^{-308}$** to **$10^{308}$**

# Double precision arithmetic (IV)

- We  now have 53 significant bits
  - Theoretical precision of $1/2^{53}$. that is, roughly $1/10^{16}$
- Can now add correctly billions or trillions

# If that is now enough, …

- Can use 128-bit quad words
- Allows us to have
  - One bit for sign
  - Fifteen bits for exponent
    - From **–16382** to **+16383**
  - One hundred twelve bits for coefficient
    - Plus the implicit leading bit

# Decimal floating point addition (I)

- $\mathbf{5.25 \times 10^3 + 1.22 \times 10^2 = ?}$
- Denormalize number with smaller exponent:
  $\mathbf{5.25 \times 10^3 + 0.122 \times 10^3}$
- Add the numbers:
  $\mathbf{5.25 \times 10^3 + 0.122 \times 10^3 = 5.372 \times 10^3}$
- Result is normalized

# Decimal floating point addition (II)

- $\mathbf{9.25 \times 10^3 + 8.22 \times 10^2 = ?}$
- Denormalize number with smaller exponent:
  $\mathbf{9.25 \times 10^3 + 0.822 \times 10^3}$
- Add the numbers:
  $\mathbf{9.25 \times 10^3 + 0.822 \times 10^3 = 10.072 \times 10^3}$
- Normalize the result:
  $\mathbf{10.072 \times 10^3 = 1.0072 \times 10^4}$

# Binary floating point addition (I)

- Say **1001 + 10** or $\mathbf{1.001 \times 2^3 + 1.0 \times 2^1}$
- Denormalize number with smaller exponent:
  $\mathbf{1.001 \times 2^3 + 0.01 \times 2^3}$
- Add the numbers:
  $\mathbf{1.001 \times 2^3 + 0.01 \times 2^3 = 1.011 \times 2^3}$
- Result is normalized

# Binary floating point addition (II)

- Say **101 + 11** or $\mathbf{1.01 \times 2^2 + 1.1 \times 2^1}$
- Denormalize number with smaller exponent:
  $\mathbf{1.01 \times 2^2 + 0.11 \times 2^2}$
- Add the numbers:
  $\mathbf{1.01 \times 2^2 + 0.11 \times 2^2 = 10.00 \times 2^2}$
- Normalize the results
  $\mathbf{10.00 \times 2^2 = 1.000 \times 2^3}$

# Binary floating point subtraction

- Say **101 – 11** or $\mathbf{1.01 \times 2^2 - 1.1 \times 2^1}$
- Denormalize number with smaller exponent:
  $\mathbf{1.01 \times 2^2 - 0.11 \times 2^2}$
- Perform the subtraction:
  $\mathbf{1.01 \times 2^2 - 0.11 \times 2^2 = 0.10 \times 2^2}$
- Normalize the results
  $\mathbf{0.10 \times 2^2 = 1.0 \times 2^1}$

# Decimal floating point multiplication

- Exponent of product is the ***sum*** of the exponents of multiplicand and multiplier
- Coefficient of product is the product of the coefficients of multiplicand and multiplier
- Compute sign using usual rules of arithmetic
- May have to renormalize the product

# Decimal floating point multiplication

- $\mathbf{6 \times 10^3 \times 2.5 \times 10^2 = ?}$
- Exponent of product is:

  $$\mathbf{3 + 2 = 5}$$

- Multiply the coefficients:

  $$\mathbf{6 \times 2.5 = 15}$$

- **Result will be positive**
- Normalize the result:

  $$\mathbf{15 \times 10^5 = 1.5 \times 10^6}$$

# Binary floating point multiplication

- Exponent of product is the ***sum*** of the exponents of multiplicand and multiplier
- Coefficient of product is the product of the coefficients of multiplicand and multiplier
- Compute sign using usual rules of arithmetic
- May have to renormalize the product

# Binary floating point multiplication

- Say **110** $\times$ **11** or $\mathbf{1.1 \times 2^2 \times 1.1 \times 2^1}$
- Exponent of product is:

  $$\mathbf{2 + 1 = 3}$$

- Multiply the coefficients:

  $$\mathbf{1.1 \times 1.1 = 10.01}$$

- Result will be positive
- Normalize the result:

  $$\mathbf{10.01 \times 2^3 = 1.001 \times 2^4}$$

# FP division

- Very tricky
- One good solution is to multiply the dividend by the inverse of the divisor

# A trap

- Addition does not necessarily commute:
  - $-9 \times 10^{37} + 9 \times 10^{37} + 4 \times 10^{-37}$
- Observe that
  - $(-9 \times 10^{37} + 9 \times 10^{37}) + 4 \times 10^{-37} = 4 \times 10^{-37}$

while

  - $-9 \times 10^{37} + (9 \times 10^{37} + 4 \times 10^{-37}) = 0$

due to the limited accuracy of FP numbers