

Computer Systems and Architecture

Data Representation

Stephen Pauwels

Academic Year 2019-2020

Base 10 to Base 2

- Convert $(23.375)_{10}$ to base 2.



Base 10 to Base 2

- Convert $(23.375)_{10}$ to base 2.
- Step 1. Remainder method: convert $(23)_{10}$ to base 2

		Remainder	
$23 / 2 = 11$		1	LSB
$11 / 2 = 5$		1	
$5 / 2 = 2$		1	
$2 / 2 = 1$		0	
$1 / 2 = 0$		1	MSB

- $(23)_{10} = (10111)_2$

Base 10 to Base 2

- Convert $(23.375)_{10}$ to base 2.
 - Step 1. Remainder method: convert $(23)_{10}$ to base 2
 - Step 2. Multiplication method: convert $(.375)_{10}$ to base 10

$$.375 \times 2 = \mathbf{0} .75$$

$$.75 \times 2 = \mathbf{1} .5$$

$$.5 \times 2 = \mathbf{1} .0$$

- $(.375)_{10} = (.011)_2$



Base 10 to Base 2

- Convert $(23.375)_{10}$ to base 2.
 - Step 1. Remainder method: convert $(23)_{10}$ to base 2
 - Step 2. Multiplication method: convert $(.375)_{10}$ to base 2
 - Step 3. Total: $(23.375)_{10}$
 - $(10111.011)_2$



Base 2 to Base 10

- Number in base 2:
 - $b_n b_{n-1} \dots b_1 b_0 . b_{-1} b_{-2} \dots b_{-m}$
- Value in base 10
 - $\sum_{i=-m}^{n-1} b_i * 2^i$
- Weighted Position Code
- Polynomial method

Base 2 to Base 10

- Convert $(1010.01)_2$ to base 10

- $(1010.01)_2$

$$= (1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 0 * 2^0 + 0 * 2^{-1} + 1 * 2^{-2})_{10}$$

$$= (8 + 0 + 2 + 0 + 0 + .25)_{10}$$

$$= (10.25)_{10}$$

Converting to other bases

- Conversion between base k and base 10
 - Remainder method
 - Divide by k
 - Multiplication method
 - Multiply by k
 - Polynomial method
 - $\sum_{i=-m}^{n-1} b_i * k^i$

Negative Numbers

- Signed Magnitude
 - First bit as sign
 - $(+12)_{10} = (00001100)_2$
 $(-12)_{10} = (10001100)_2$

Negative Numbers

- Signed Magnitude
- One's Complement
 - Convert all zero's in 1's and 1's in zero's
 - $(+12)_{10} = (00001100)_2$
 $(-12)_{10} = (11110011)_2$

Negative Numbers

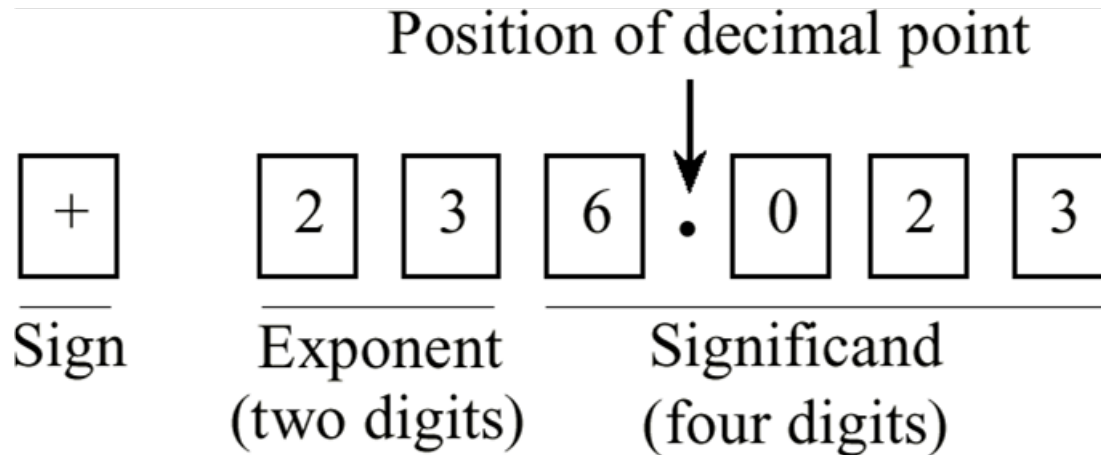
- Signed Magnitude
- One's Complement
- Two's Complement
 - Take the one's complement, and add 1
 - $(+12)_{10} = (00001100)_2$
 - $(-12)_{10} = (11110100)_2$

Negative Numbers

- Signed Magnitude
- One's Complement
- Two's Complement
- Excess
 - Add bias to every number and convert it as a positive number
 - $(+12)_{10} = (100001100)_2$ $(12 + 128 = 140)$
 - $(-12)_{10} = (01110100)_2$ $(-12 + 128 = 116)$

Floating Point

- Wide range of numbers in a limited number of bits
- Example: $+6.023 * 10^{23}$



Normalization

- Different representations for the same number
 - $254 * 10^0$
 - $25.4 * 10^1$
 - $0.0254 * 10^4$
 - $254000 * 10^{-3}$
- Normalization: the dot is placed right of the leftmost digit
 - $2.54 * 10^2$
- Hidden bit: in base 2 the leftmost bit can be discarded
 - $1.1010 \rightarrow 1010$

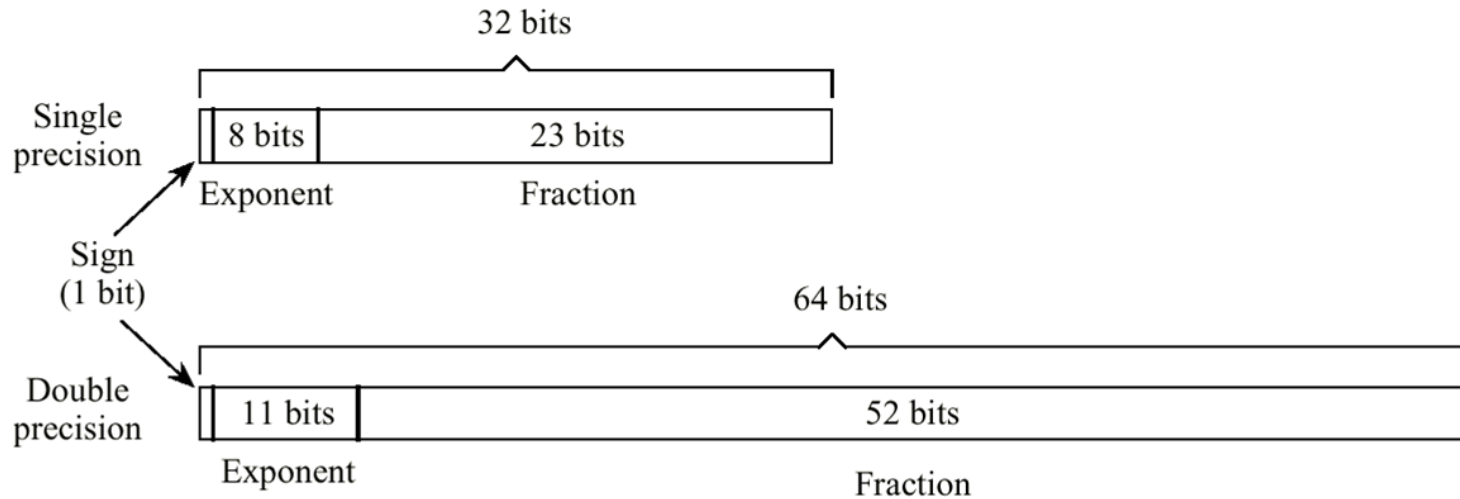


Conversion

- Convert $(9.375 * 10^{-2})_{10}$ to base 2 (scientific notation)
 1. Fixed point: $(.09375)_{10}$
 2. Multiplication method: $(.0011)_2$
 3. Normalize: $.0011 = 0.0011 * 2^0 = 1.1 * 2^{-3}$

IEEE-754

- Sign bit
- Exponent: Excess 127 – 1023
- Number representation with hidden bit



IEEE-754

- Represent $(-12.625)^{10}$ in IEEE-754 Single Precision format
 1. Convert to base 2
 - $(-12.625)_{10} = (-1100.101)_2$
 2. Normalize
 - $(-1100.101)_2 = (-1.100101)_2 * 2^3$
 3. Calculate exponent in excess 127
 - $3 + 127 = 130 \rightarrow (10000010)_2$

1 1000 0010 100 1010 0000 0000 0000 0000



IEEE-754 – Special Cases

- +0
 - 0 0000 0000 000 0000 0000 0000 0000 0000
- +∞
 - 0 1111 1111 000 0000 0000 0000 0000 0000
- + NaN
 - 0 1111 1111 001 0010 0000 0100 1000 0000
- 2^{-128} (Denormalized): $(1 * 2^{-2}) * 2^{-126}$
 - 0 0000 0000 010 0000 0000 0000 0000 0000

Exercises

- Blackboard
- Course webpage
 - <http://msdl.cs.mcgill.ca/people/hv/teaching/ComputerSystemsArchitecture/#CS4>

