Computer Arithmetic Part 1: Understanding Data Representation

Data Representation

- Data Types
- Complements
- Fixed Point Represntations
- Floating Point Representations

Computer-Dealing what kind of Information?

- Data
    - 1, 3.14, -9, ...
    - A, B, C, &, %
- Relationship among data elements
    - Data Structure.
        - Linear List, Trees, Rings, etc.
- Program
    - Set of instructions

More about Numbers

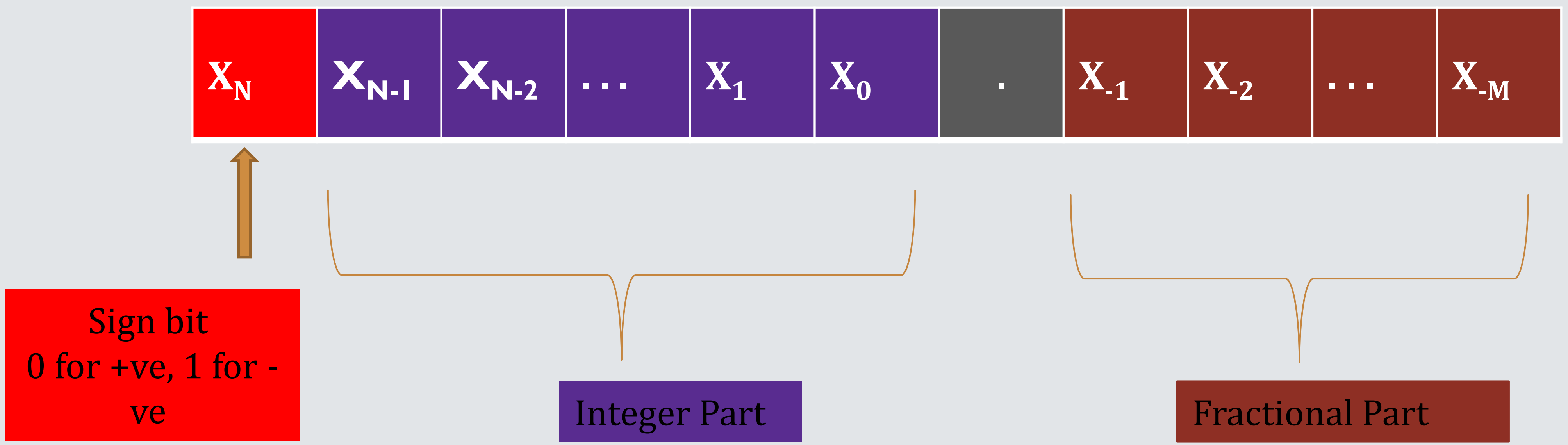
Number System can be.

- Non-Positional – Roman
- Positional – Decimal, Hexadecimal, Octal, Binary

Base or Radix – Uses R distinct symbol

- Example-
  - Binary – 0 and 1,
  - Decimal- 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Fixed-Point Representation



Examples

1011.1	➔ $1x2^3 + 0x2^2 + 1x2^1 + 1x2^0 + 1x2^{-1}$	=	11.5
101.11	➔ $1x2^2 + 0x2^1 + 1x2^0 + 1x2^{-1} + 1x2^{-2}$	=	5.75
10.111	➔ $1x2^1 + 0x2^0 + 1x2^{-1} + 1x2^{-2} + 1x2^{-3}$	=	2.875

Limitations

- More number of bits require to achieve more precision.
- $(1/3)*3 \neq 1$

Signed Numbers

To represent both positive and negative numbers

3

There are three representation

- Signed Magnitude
- Signed 1's Complement
- Signed 2's Complement

Signed Magnitude

Representation of +12 and –12 in an 8-bit binary number

+12 = 0000 1100

-12 = 1000 1100

Simple

0 ? There are two representation

255 different numbers for an 8-bit representation

Sign and Magnitude

1's Complement

Representation of +12 and –12 in an 8-bit binary number

+12 = 0000 1100

-12 = 1111 0011

0 ? There are two representations of 0

255 different numbers for an 8-bit representation

Complexity in performing addition and subtraction

2's Complement

Representation of +12 and –12 in an 8-bit binary number

+12 = 0000 1100

-12 = 1111 0100

Only one representation for 0

256 different number for an 8-bit representation

Arithmetic works easily

Negating is fairly easy

Floating Point Representation (IEEE-754)

- ✓ A number F is represented as a triplet <s, E, M>
- ✓ $F = (-1)^{S} M * 2^{E}$

s_n	$e_k e_{k-1} \dots e_0$	$m_{n-1} m_{n-2} \; \dots \; m_0 . m_{-1} \; \dots \; m_{-m}$
Sign	Designates the positions of the radix point	Signed fixed point number, either an integer or a fractional number

- ✓ Sign bit indicating negative =1 or positive =0
- ✓ M is called the Mantissa, and is normally a fraction in the range of [1.0-2.0]
- ✓ E is called the exponent, which weights the number by power of 2.

Encoding:

- Single-precision numbers: total 32 bits, E 8 bits, M 23 bits
- Double-precision numbers: total 64 bits, E 11 bits, M 52 bits

s	E	M

- Range of E: $1 \leq E \leq 254$ (all 0s and all 1s are reserved for special number)

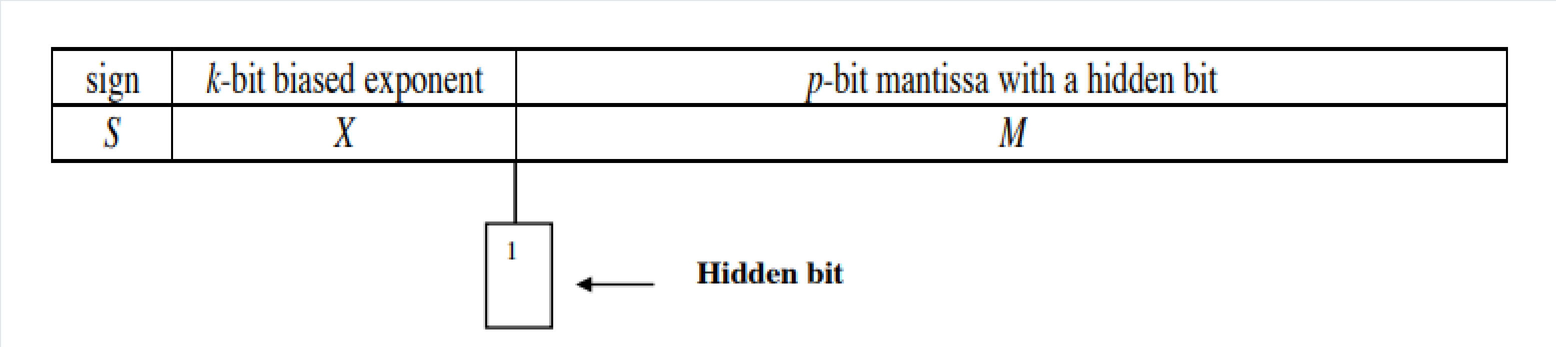
- Encoding Exponent with bias value: E = Exponent + Bias
  - (Bias : Single Precision = 127, Double Precision = 1023)

- Encoding Mantissa M
  - The mantissa is coded with an implied leading 1 (i.e. in 24 bits).

    $M = 1 \,.\, xxxx...x$
  - Here, *xxxx…x* denotes the bits that are stored for the mantissa. We get the extra leading bit for free.

Bias

- The value stored is offset from the actual value by the exponent bias, also called a biased exponent
- Biasing is done so that exponents can be +ve or –ve, in two's complement



- The true exponent, x, is found by subtracting a fixed number from the biased exponent, X. This fixed number is called the bias. For a k-bit exponent, the bias is $2^{k-1}-1$, and the true exponent, x and X are related by

$$x = X - (2^{k-1}-1)$$

Example: In single precision, if exponent bias X = 134, then x = 134 -127= 7

Example: IEEE 754 Representation

F= -3.75

Consider the number F = -3.75

$-3.75_{10} = -11.11_2 = -1.111 \times 2^1$

Mantissa will be stored as: $M = 11100000000000000000000_2$

Here, EXP = 1, BIAS = 127. ➔ $E = 1 + 127 = 128 = 10000000_2$

1	10000000	11100000000000000000000

40700000 in hex

Thank You