CSARCH Lecture Series: Binary Floating-Point format for Single Precision

Sensei RL Uy

College of Computer Studies

De La Salle University

Manila, Philippines

Copyright Notice

Overview

Reflect on the following questions:

- How is single-precision floating-point data stored in the memory?
- Given the code below, how are float data stored in the memory?

```
int main()
{
    float var, var1;
    var = 2.5;
    var1 = -1.28e2;
}
```

Overview

- This sub-module introduces the IEEE-754 single-precision floating-point format
- The objective is as follows:
  - ✓ Describe the process of representing single-precision floating-point data using IEEE-754 standard

Floating Point

- Scientists and engineers use scientific notation where a number is expressed as

$$+/-\ S \times 10^{\pm E}$$

- Where S is the significand (also known as mantissa), E is the exponent and 10 is the base.

$$\underbrace{6.022}_{\text{significand}} \times \underbrace{10}_{\text{base}}{}^{\underbrace{23}_{\text{exponent}}}$$

Floating Point

- Computers represent floating-point numbers using scientific notation in base 2.

$$\pm S \times 2^{\pm E}$$

$$\underbrace{1.101}_{\text{significand}} \text{ x } \underbrace{2}_{\text{base}}{}^{\underbrace{5}_{\text{exponent}}}$$

Floating Point

- Floating point standard for floating-point numbers in computer is the IEEE-754 (Institute of Electrical and Electronics Engineers Standard 754). Originally 1985, revised 2008, current version 2019
- IEEE-754 defines the following:
  - representation format
  - floating-point operations
  - rounding rules
  - exception handling (i.e., divide by zero)

IEEE-754 representation format

- Binary Half precision
- Binary Single precision
- Binary Double precision
- Binary Quadruple precision
- Decimal single precision
- Decimal double precision

IEEE-754 single-precision floating-point format

Sign	Exponent representation	Fraction part of significand
1	8	23
s	e’	f

normalized to this format before representation:

$1.f \times 2^{e}$

- IEEE-754 single-precision floating-point format is 32-bit in width
- The 32-bit is partition as 1 bit for sign bit; 8 bits for exponent representation and 23 bits for the fractional part of the significand
  - Significand in binary
  - Base 2
  - sign bit: 0→ positive; 1→ negative
  - e’ = e+127
  - significand normalized to 1.f (implied 1)

IEEE-754 single-precision floating-point format

Example: $+1.00111_2 \times 2^5$

normalized format: (same) $+1.00111_2 \times 2^5$

Significand in binary?	Yes
Base-2?	Yes
Normalized?	Yes
Sign bit	0
e' = e+127	5+127=132 [1000 0100]

Answer:

Sign	Exponent representation	Fraction part of significand
0	1000 0100	001 1100 0000 0000 0000 0000

Hex: 0x421C0000 [combine and group by 4]→ 0100 0010 0001 1100 0000 0000 0000 0000

IEEE-754 single-precision floating-point format

Example: $-100.111_2 x\ 2^{-7}$

normalized format: $-1.00111_2 x\ 2^{-5}$

Significand in binary?	Yes
Base-2?	Yes
Normalized?	No
Sign bit	1
e’ = e+127	-5+127=122 [0111 1010]

Answer:

Sign	Exponent representation	Fraction part of significand
1	0111 1010	001 1100 0000 0000 0000 0000

Hex: 0xBD1C0000 [combine and group by 4]→ 1011 1101 0001 1100 0000 0000 0000 0000

IEEE-754 single-precision floating-point format

Example: $-0.000100111_2 \text{x } 2^{15}$

normalized format: $-1.00111_2 \text{x } 2^{11}$

Significand in binary?	Yes
Base-2?	Yes
Normalized?	No
Sign bit	1
e' = e+127	11+127=138 [1000 1010]

Answer:

Sign	Exponent representation	Fraction part of significand
1	1000 1010	001 1100 0000 0000 0000 0000

Hex: 0xC51C0000 [combine and group by 4]→ 1100 0101 0001 1100 0000 0000 0000 0000

IEEE-754 single-precision floating-point format

Example: +4.0 $+100.0_2 \times 2^0$

normalized format: $+1.000_2 \times 2^2$

Significand in binary?	No
Base-2?	No
Normalized?	No
Sign bit	0
e' = e+127	2+127=129 [1000 0001]

Answer:

Sign	Exponent representation	Fraction part of significand
0	1000 0001	000 0000 0000 0000 0000 0000

Hex: 0x40800000 [combine and group by 4]→ 0100 0001 1000 0000 0000 0000 0000 0000

IEEE-754 single-precision floating-point format

Example: $-100.111_2 \times 2^{-7}$

normalized format: $-1.00111_2 \times 2^{-5}$

Significand in binary?	Yes
Base-2?	Yes
Normalized?	No
Sign bit	1
e' = e+127	-5+127=122 [0111 1010]

Answer:

Sign	Exponent representation	Fraction part of significand
1	0111 1010	001 1100 0000 0000 0000 0000

Hex: 0xBD1C0000 [combine and group by 4]→ 1011 1101 0001 1100 0000 0000 0000 0000

Try

label	Address	Memory data (binary)
var1	0008	
var	0000	

```
int main()
{
    float var, var1;
    var = 2.5;
    var1 = -1.28e2;
}
```

Try

```
int main()
{
    float var, var1;
    var = 2.5;
    var1 = -1.28e2;
}
```

label	Address	Memory data (binary)
var1	0008	1100 0011 0000 0000 0000 0000 0000 0000
var	0000	0100 0000 0010 0000 0000 0000 0000 0000

label	address	Memory data (hex)
var1	0008	4020 0000
var	0000	C300 0000

$+2.5 = 10.1_2 x\ 2^0 = 1.01_2 x\ 2^1$

Sign	Exponent representation	Fraction part of significand
0	1000 0000	010 0000 0000 0000 0000 0000

$-1.28e2 = -128.0 = 10000000.0_2 x\ 2^0 = 1.0_2 x\ 2^7$

Sign	Exponent representation	Fraction part of significand
1	1000 0110	000 0000 0000 0000 0000 0000

Try

IEEE-754 Single-Precision Floating-point	Decimal equivalent
1100 0000 0110 0000 0000 0000 0000 0000	
0100 0000 0111 0000 0000 0000 0000 0000	

Try

IEEE-754 Single-Precision Floating-point	Decimal equivalent
1100 0000 0110 0000 0000 0000 0000 0000	-3.5
0100 0000 0111 0000 0000 0000 0000 0000	+3.75

implied 1

Sign	Exponent representation	Fraction part of significand
1	10000000	110 0000 0000 0000 0000 0000

e = e'-127

= 128-127

= 1

significand: 1.110

= 1.75

Answer:

-1.75×2^1

= -3.5

Sign	Exponent representation	Fraction part of significand
0	1000000	111 0000 0000 0000 0000 0000

e = e'-127

= 128-127

= 1

significand: 1.111

= 1.875

Answer:

$+1.875 \times 2^1$

= +3.75

To recall …

- What have we learned:
  - ✓ Describe the process of representing single-precision floating-point data using IEEE-754 standard