Tutorial 05: Floating-point Numbers

Computer Science Department
CS2208: Introduction to Computer Organization and Architecture
Fall 2022-2023
Instructor: Mahmoud R. El-Sakka
Office: MC-419
Email: elsakka@csd.uwo.ca
Phone: 519-661-2111 x86996

Example of Decimal to IEEE-754 Floating-point Conversion

- ***Example 1***: *Convert* $5.877472_{10}\times10^{-39}$ into a *32-bit single-precision IEEE-754 FP* value.

$\text{Log}_2(10) = 1 / \log_{10}(2)$

$10^{-39}=2^{z}$ → $\log_2(10^{-39}) = z$ → $-39\times\log_2(10) = z$ → $z = -129.5551957$

$10^{-39} = 2^{-129.5551957} = 2^{-129} \times 2^{-0.5551957} = 2^{-129} \times 0.680564734_{10}$

$5.877472_{10} \times 10^{-39} = 5.877472_{10} \times 0.680564734_{10} \times 2^{-129}$

$= 4_{10} \times 2^{-129} = 2^{2} \times 2^{-129} = 2^{-127} = 1_2 \times 2^{-127}$

- Convert 1_2 into a fixed-point binary
  - $1_2 = 1.0_2$ (*already normalized*)
- *True exponent is less than -126 → underflow case*
  - *The exponent needs to be -126:* $-127_{10} = -126 - 1$
  - *Hence, the significant needs to be adjusted to compensate the -1*
  - After moving the radix point backward by 1 position → 0.1_2

    i.e., $1.0_2 \times 2^{-127} = 0.1_2 \times 2^{-126}$
  - After Taking 23 bits → $0.\ 100\ 0000\ 0000\ 0000\ 0000\ 0000_2$
- The sign bit, S, is 0 because the number is positive
- The final number is $\underbrace{0000\ 0000\ 0}$100 0000 0000 0000 0000 0000 or 00400000_{16}

↑

zero/underflow.

Example of Decimal to IEEE-754 Floating-point Conversion

- *Example 2*: *Convert* $9.0_{10} \times 10^{-44}$ into a *32-bit single-precision IEEE-754 FP* value.

$\mathrm{Log}_2(10) = 1 / \log_{10}(2)$

$10^{-44}=2^z$ ➔ $\log_2(10^{-44}) = z$ ➔ $-44\times\log_2(10) = z$ ➔ $z = -146.164836175$

$10^{-44} = 2^{-146.164836175} = 2^{-146} \times 2^{-0.164836175} = = 2^{-146} \times 0.892029808_{10}$

$9.0_{10} \times 10^{-44} = 9.0_{10} \times 0.892029808_{10} \times 2^{-146} = 8.028268272_{10} \times 2^{-146}$

- Convert 8.028268272_{10} into a fixed-point binary
  - $8_{10} = 1000_2$ and
  - $0.028268272_{10} = 0.0000011100111100100 1\ldots_2$.
  - Therefore, $8.028268272_{10} = 1000.000001110011110010 01\ldots_2$.
- *Normalization:* $9.0_{10} \times 10^{-44} = 8.028268272_{10} \times 2^{-146} =$ $1000.00000111001111001001\ldots_2\times2^{-146}= 1.00000000111001111001001\ldots_2\times2^{-143}$
- *True exponent is less than -126 ➔ underflow case*
  - *The exponent needs to be -126:* $-143_{10} = -126 -17$
  - *Hence, the significant needs to be adjusted to compensate the -17*
  - After moving the radix point backward by 17 position

    ➔ 0. **0000 0000 0000 0000** 1000 0000 0111001111001001…$_2$
  - After Taking only 23 bits ➔ 0. **000 0000 0000 0000 01**00 0000 0011…$_2$

Rounded to the nearest FP

- The sign bit, S, is 0 because the number is positive
- The final number is 0000 0000 0000 0000 0000 0000 0100 0000 or 00000040_{16}

Example of Decimal to IEEE-754 Floating-point Conversion

- ***Example 3***: *Convert* 3.6_{10} into a *32-bit single-precision IEEE-754 FP* value.
  - ○ Convert 3.6_{10} into a fixed-point binary
    - $3_{10} = 11_2$ and
    - $0.6_{10} = 0.1001\ 1001 \ldots {}_2$.
    - Therefore, $3.6_{10} = 11.1001\ 1001 \ldots {}_2$

0.6 × 2 = 1.2
0.2 × 2 = 0.4
0.4 × 2 = 0.8
0.8 × 2 = 1.6
0.6 × 2 = 1.2
…

  - ○ Normalize $11.1001\ 1001 \ldots {}_2$ to
    $1.11001\ 1001 \ldots {}_2 \times 2^1$.
  - ○ The sign bit, S, is 0 because the number is positive
  - ○ The *biased exponent* is the *true exponent* plus 127; that is,
    $1 + 127 = 128_{10} = 1000\ 0000_2$
  - ○ The fractional significand is 110 0110 0110 0110 0110 0110 0110 0110 …
    - *the leading 1 was stripped* and
    - *to be rounded to 23 bits (rounded to nearest FP number)*.
  - ○ The final number is 0100 0000 0110 0110 0110 0110 0110 0110,
    or 40666666_{16}. ➔ 3.599999046325684_{10}

Example of Decimal to IEEE-754 Floating-point Conversion

- *Example 4*:
  Convert 16777216.75_{10} into a *32-bit single-precision IEEE-754 FP* value.

  - Convert 16777216.75_{10} into a fixed-point binary
    - 16777216_{10} = 1 0000 0000 0000 0000 0000 0000_2 and
    - $0.75_{10} = 0.11_2$.
    - Therefore, 16777216.75_{10} = 1 0000 0000 0000 0000 0000 0000.11_2.
  - Normalize 1 0000 0000 0000 0000 0000 0000.11_2 to
    1.0000 0000 0000 0000 0000 0000 $11_2 \times 2^{24}$.
  - The sign bit, S, is 0 because the number is positive
  - The *biased exponent* is the *true exponent* plus 127; that is,
    $24 + 127 = 151_{10}$ = 1001 0111_2
  - The fractional significand is 000 0000 0000 0000 0000 0000 011
    - *the leading 1 was stripped* and
    - *to be rounded to 23 bits (rounded to nearest FP number)*.
  - The final number is 0100 1011 1000 0000 0000 0000 0000 0000,
    or $4B800000_{16}$ ➔ 16777216_{10} *(i.e., there is 0.75 rounding error)*

Example of Decimal to IEEE-754 Floating-point Conversion

❑ *Example 5*:
Convert 16777219_{10} into a *32-bit single-precision IEEE-754 FP* value.

- Convert 16777219_{10} into a fixed-point binary
  - 16777219_{10} = $1\ 0000\ 0000\ 0000\ 0000\ 0000\ 0011_2$ and
- Normalize $1\ 0000\ 0000\ 0000\ 0000\ 0000\ 0011_2$ to
  $1.0000\ 0000\ 0000\ 0000\ 0000\ 0011_2 \times 2^{24}$.
- The sign bit, S, is 0 because the number is positive
- The *biased exponent* is the *true exponent* plus 127; that is,
  $24 + 127 = 151_{10} = 1001\ 0111_2$
- The fractional significand is 000 0000 0000 0000 0000 0001 1
  - *the leading 1 was stripped* and
  - *to be rounded to 23 bits (rounded to nearest FP number)*.
- The final number is 0100 1011 1000 0000 0000 0000 0000 0010,
  or $4B800002_{16}$ → 16777220_{10} *(i.e., there is 1.0 rounding error)*

Mid-way → round to even significand

Example of IEEE-754 FP to Decimal to IEEE-754 FP Conversion

- ***Example 6***:
  Convert $4B800002_{16}$ from the *32-bit single-precision IEEE-754 FP* representation into decimal representation. ***Then*** add 1.0_{10} to the result. And ***finally*** convert it back to the *32-bit single-precision IEEE-754 FP* representation.

  - Convert the hexadecimal number ($4B800002_{16}$) into binary form

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0	1	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

  - Unpack the number into *sign bit*, *biased exponent*, and *fractional significand*.
    - S = 0
    - E = 1001 0111
    - F =000 0000 0000 0000 0000 0010
  - As the sign bit is 0, the number is positive.
  - We subtract 127 from the *biased exponent* $1001\ 0111_2$ to get the *true exponent* ➔ $1001\ 0111_2 - \ 0111\ 1111_2 = 0001\ 1000_2 = 24_{10}$.
  - The fractional significand is $.000\ 0000\ 0000\ 0000\ 0000\ 0010_2$.
  - Reinserting the leading one gives $1.000\ 0000\ 0000\ 0000\ 0000\ 0010_2$.
  - The number is $+(1 + 2^{-22}) \times 2^{24} = 2^{24} + 2^{2} = 1024_{10} \times 1024_{10} \times 16_{10} + 4_{10}$
    $= 16777216_{10} + 4_{10} = 16777220_{10}$

Example of IEEE-754 FP to Decimal to IEEE-754 FP Conversion

- ***Example 6 (continution)***:

  Adding 1.0_{10} to the result ➔ $16777220_{10} + 1.0_{10} = 16777221_{10}$

  Converting the result back to the *32-bit single-precision IEEE-754 FP* format

  - Convert 16777221_{10} into a fixed-point binary
    - $16777221_{10} = 1\ 0000\ 0000\ 0000\ 0000\ 0000\ 0101_2$ and
  - Normalize $1\ 0000\ 0000\ 0000\ 0000\ 0000\ 0101_2$ to
    $1.0000\ 0000\ 0000\ 0000\ 0000\ 0101_2 \times 2^{24}$.
  - The sign bit, S, is 0 because the number is positive
  - The *biased exponent* is the *true exponent* plus 127; that is, $24 + 127 = 151_{10} = 1001\ 0111_2$
  - The fractional significand is 000 0000 0000 0000 0000 0010 1
    - *the leading 1 was stripped* and
    - *to be rounded to 23 bits (rounded to nearest FP number)*.
  - The final number is 0100 1011 1000 0000 0000 0000 0000 0010, or $4B800002_{16}$ ➔ 16777220_{10}

Mid-way ➔ round to even significand

$16777220_{10} + 1.0_{10} = 16777220_{10}$!!!

(This is due to the rounding error)

This is the same FP number that we started with!!

Example of IEEE-754 FP to Decimal to IEEE-754 FP Conversion

- ***Example 6 (continution)***:

○ Run the following C program to verify ***Example 6***:

```
#include <stdio.h>
int main()
{
  float f = 16777220, ff;
  ff = f + 1;
  printf("%f %f \n", f, ff);
  return 0;
}
```

```
The output will be:
16777220.000000 16777220.000000
```

Change the "`float`" to "`int`" and the "`%f`" to "`%d`" and repeat executing the program again.

```
The output after the "float" to "int" change will be:
16777220 16777221
```

Change the "`float`" to "`double`" and the "`%f`" to "`%lf`" and repeat executing the program again.

```
The output after the "float" to "double" change will be:
16777220.000000 16777221.000000
```

Example of IEEE-754 Floating-point to Decimal Conversion

- ***Example 7***: *Convert* 00200000_{16} from *32-bit single-precision IEEE-754 FP* value into a decimal value.
  - ○ Convert the hexadecimal number (00200000_{16}) into binary form

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

  - ○ Unpack the number into *sign bit*, *biased exponent*, and *fractional significand*.
    - ▪ S = 0
    - ▪ E = 0000 0000 ← underflow .
    - ▪ F =010 0000 0000 0000 0000 0000
  - ○ As the sign bit is 0, the number is positive.
  - ○ We subtract 126 from the *biased exponent* 0_2 to get the *true exponent* ➔ $0_2 - 0111\ 1110_2 = -126_{10}$.
    As the true exponent is -126, then the F is not normalized
  - ○ The fractional significand is $.010\ 0000\ 0000\ 0000\ 0000\ 0000_2$.
  - ○ The number is $.01_2 \times 2^{-126} = 2^{-2} \times 2^{-126} = 2^{-128}$

We are subtracting 126, not 127, from the biased exponent, because the biased exponent = 0.

$2^{-128} = 10^{z}$ ➔ $\log_{10}(2^{-128}) = z$ ➔ $z = -38.53183944$

$2^{-128} = 10^{-38.53183944} = 10^{-38} \times 10^{-0.\ 53183944} = 10^{-38} \times 0.293873587$

$2^{-128} = 0.293873587 \times 10^{-38} = 2.9387358 \times 10^{-39}$

Final Word!!

- **How can you verify your FP conversion results?**
- There are many online converters between IEEE FP format to float and vice versa.
  - For example, https://www.h-schmidt.net/FloatConverter/IEEE754.html