CS2323 HW3

Roll: CO22BTECH11006
Name: Om Dave

Q1 (Decimal to floating point representation)

(a) -13.25 (Single Precision)

Sign Bit: 1

Binary Conversion:

- Integer (13): `1101`
- Fraction (0.25): `.01`
- Combined: `1101.01`

Normalization: `1.10101 * 2^3`

Exponent Calculation:

- Actual Exponent: 3
- Bias (Single): 127
- Biased Exponent: 3 + 127 = 130
- Binary (8-bit): `10000010`

Mantissa Calculation:

- From Normalization: `10101`
- Padded (23-bit): `10101000000000000000000`

Final Binary Representation:

- `1 10000010 10101000000000000000000`

Hexadecimal Conversion:

- Grouped: `1100 0001 0101 0100 0000 0000 0000 0000`
- Final Hex: `0xC1540000`

(b) 0.1 (Single Precision)

Sign Bit: 0

Binary Conversion:

- `0.0001100110011...` (repeating `0011`)

Normalization: `1.100110011.* 2^-4`

Exponent Calculation:

- Actual Exponent: -4
- Bias (Single): 127
- Biased Exponent: -4 + 127 = 123
- Binary (8-bit): `01111011`

Mantissa Calculation:

- Repeating Pattern: `100110011...`
- Truncated/Rounded (23-bit): `10011001100110011001101`

Final Binary Representation:

- `0 01111011 10011001100110011001101`

Hexadecimal Conversion:

- Grouped: `0011 1101 1100 1100 1100 1100 1100 1101`
- Final Hex: `0x3DCCCCCD`

(c) 156.75 (Double Precision)

Sign Bit: 0

Binary Conversion:

- Integer (156): `10011100`
- Fraction (0.75): `.11`
- Combined: `10011100.11`

Normalization: `1.001110011 * 2^7`

Exponent Calculation:

- Actual Exponent: 7
- Bias (Double): 1023
- Biased Exponent: 7 + 1023 = 1030
- Binary (11-bit): `10000000110`

Mantissa Calculation:

- From Normalization: `001110011`
- Padded (52-bit): `001110011000...000`

Final Binary Representation:

- `0 10000000110 0011100110000000000000000000000000000000000000000000`

Hexadecimal Conversion:

- Grouped: `0100 0000 0110 0011 1001 1000 0000 .0000`
- Final Hex: `0x4063980000000000`

(d) -0.0078125 (Double Precision)

Sign Bit: 1

Binary Conversion:

- `0.0000001`

Normalization: `1.0 * 2^-7`

Exponent Calculation:

- Actual Exponent: -7
- Bias (Double): 1023
- Biased Exponent: -7 + 1023 = 1016
- Binary (11-bit): `01111111000`

Mantissa Calculation:

- From Normalization: `0`
- Padded (52-bit): All zeros

Final Binary Representation:

- `1 01111111000 0000000000000000000000000000000000000000000000000000`

Hexadecimal Conversion:

- Grouped: `1011 1111 1000 0000 0000 .0000`
- Final Hex: `0xBF8000000000000`

Q2 (Floating point to Decimal representation)

(a) 0xC1200000 (Single Precision)

Hex to Binary:

- `1100 0001 0010 0000 0000 0000 0000 0000`

Deconstruct Bits (1-8-23):

- **Sign:** `1` (Negative)
- **Exponent:** `10000010` (Decimal: 130) -> Actual Exponent: 130 - 127 = 3
- **Mantissa:** `01000000000000000000000`

Decimal Conversion:

- **Normalized Binary (1.Mantissa):** `1.01`
- **Non-normalized Binary (Shift by Exponent 3):** `1.01 * 2^3` = `1010.0`
- **Binary to Decimal:** `8 + 0 + 2 + 0` = `10.0`
- **Final Value (With Sign): -10.0**

(b) 0x3F800000 (Single Precision)

Hex to Binary:

- `0011 1111 1000 0000 0000 0000 0000 0000`

Deconstruct Bits (1-8-23):

- **Sign:** `0` (Positive)
- **Exponent:** `01111111` (Decimal: 127) -> Actual Exponent: 127 - 127 = 0
- **Mantissa:** `00000000000000000000000`

Decimal Conversion:

- **Normalized Binary (1.Mantissa):** `1.0`
- **Non-normalized Binary (Shift by Exponent 0):** `1.0 * 2^0` = `1.0`
- **Binary to Decimal:** `1.0`
- **Final Value (With Sign): 1.0**

(c) 0xBFF0000000000000 (Double Precision)

Hex to Binary:

- `1011 1111 1111 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000`

Deconstruct Bits (1-11-52):

- **Sign:** `1` (Negative)
- **Exponent:** `01111111111` (Decimal: 1023) -> Actual Exponent: 1023 - 1023 = 0
- **Mantissa:** `000...` (All zeros)

Decimal Conversion:

- **Normalized Binary (1.Mantissa):** `1.0`
- **Non-normalized Binary (Shift by Exponent 0):** `1.0 * 2^0` = `1.0`
- **Binary to Decimal:** `1.0`
- **Final Value (With Sign): -1.0**

(d) 0x4024000000000000 (Double Precision)

Hex to Binary:

- `0100 0000 0010 0100 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000`

Deconstruct Bits (1-11-52):

- **Sign:** `0` (Positive)
- **Exponent:** `10000000010` (Decimal: 1026) -> Actual Exponent: 1026 - 1023 = 3
- **Mantissa:** `010000...`

Decimal Conversion:

- **Normalized Binary (1.Mantissa):** `1.01`
- **Non-normalized Binary (Shift by Exponent 3):** `1.01 * 2^3` = `1010.0`

- **Binary to Decimal:** `10.0`
- **Final Value (With Sign): 10.0**

Q3

Given (hex): A = `0x41480000` (single) B = `0xC0700000` (single)

Decode A (Single Precision)

- **Hex → Binary (32):** `0100 0001 0100 1000 0000 0000 0000 0000`
- **Deconstruct (1–8–23):** `0 | 10000010 | 10010000000000000000000`
- **Sign Bit:** `0`
- **Exponent:** $10000010_2 = 130$ ⇒ Actual $e = 130 - 127 = 3$
- **Mantissa (with hidden 1):** `1.1001`
- **Value:** $1.1001_2 \times 2^3 = 12.5$

Decode B (Single Precision)

- **Hex → Binary (32):** `1100 0000 0111 0000 0000 0000 0000 0000`
- **Deconstruct (1–8–23):** `1 | 10000000 | 11100000000000000000000`
- **Sign Bit:** `1`
- **Exponent:** $10000000_2 = 128$ ⇒ Actual $e = 1$
- **Mantissa (with hidden 1):** `1.111`
- **Value:** $-(1.111_2 \times 2^1) = -3.75$

Result (Normalized)

Value: $12.5 + (-3.75) = 8.75 = 1.00011_2 \times 2^3$

Encode Result as **Double Precision**

- **Sign Bit:** `0`

- **Exponent Calculation:** Actual $e = 3$, Bias (double) `1023` ⇒ $E = 1026$ ⇒ `10000000010`

- **Mantissa (52-bit):** from `1.00011` ⇒ fraction `00011` + pad to 52 `000110..`

- **Final Binary Representation (1–11–52):** `0 10000000010 0001100..`

- **Hexadecimal Conversion (64, grouped):** `0100 0000 0010 0001 1000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000`

- **Final Hex: 0x4021800000000000**

Q4

Decode A (Double Precision)

- **Hex → Binary (64):** `0100 0000 0011 1001 0000 .0000`
- **Deconstruct (1–11–52):** `0 | 10000000011 | 1001000…(zeros)…`

- **Sign Bit:** `0`
- **Exponent:** $10000000011_2 = 1027$ ⇒ Actual $e = 1027 - 1023 = 4$
- **Mantissa (with hidden 1):** `1.1001`
- **Value:** $1.1001_2 \times 2^4 = 25.0$

Decode B (Double Precision)

- **Hex → Binary (64):** `1100 0000 0000 1000 0000 .0000`
- **Deconstruct (1–11–52):** `1 | 10000000000 | 1000000…(zeros)…`
- **Sign Bit:** `1`
- **Exponent:** $10000000000_2 = 1024$ ⇒ Actual $e = 1024 - 1023 = 1$
- **Mantissa (with hidden 1):** `1.1`
- **Value:** $-(1.1_2 \times 2^1) = -3.0$

Multiply (Real)

- **Value:** `25.0 * -3 = -75.0`

Normalize Product

- **Value:** $-75.0 = = -(1001011_2) = -(1.001011_2 \times 2^6)$

Encode Result as **Single Precision**

- **Sign Bit:** `1`
- **Exponent Calculation:** Actual $e = 6$, Biased exponent (single) ⇒ $E = 127 + 6 = 133$ ⇒ `10000101`
- **Mantissa (23-bit):** from `1.001011` ⇒ fraction `001011` + pad to 23

Final Binary Representation (1–8–23): `1 10000101 00101100000000000000000`

Hexadecimal Conversion (32, grouped): `1100 0010 1001 0110 0000 0000 0000 0000`

- **Final Hex: 0xC2960000**

Q5

Take the integer:

$$ N = 2^{24} + 1 = 16{,}777{,}217 $$

- A 32-bit signed integer can represent all values from -2^{31} to $2^{31}-1$. So $16{,}777{,}217$ can be represented.

- A 32-bit float (IEEE 754 single precision) has:

  - 23 fraction bits + 1 hidden bit = 24 bits of precision.
  - This means every integer up to 2^{24} can be exactly represented.
  - Once an integer requires more than 24 bits to write in binary, precision is lost.

Floating-point representation of $2^{24}+1$

- Binary form:

  $$ 2^{24}+1 = 1.0000\ldots000001 \times 2^{24} $$

  (23 zeros then a trailing 1).

- Mantissa: only 23 fraction bits are stored. The trailing 1 at the 24th place does not fit.

Thus the stored value becomes:

$$ 1.0000\ldots0000 \times 2^{24} = 2^{24} = 16{,}777{,}216 $$

$16{,}777{,}217$ exists in 32-bit signed int but not in float. In float it is rounded off to $16{,}777{,}216$, losing the $+1$.

Q6

Let float32 numbers:

$$ a=-2^{24},\quad b=2^{24},\quad c=1. $$

- **Left-associate** $(a+b)+c$:

  $$ ((-2^{24})+2^{24})+1 = 0+1 = 1\quad(\text{exact}). $$

- **Right-associate** $a+(b+c)$:

  $$ b+c = 2^{24}+1 ;\xrightarrow{\text{round}}; 2^{24}\quad(\text{As seen previously}), $$

  so

  $$ a+(b+c)=(-2^{24})+2^{24}=0. $$

Hence,

$$ (a+b)+c = 1 ;\neq; a+(b+c) = 0. $$

Hence, addition in floating point numbers is not always associative.

Q5

Take the integer:

$$ N = 2^{24} + 1 = 16{,}777{,}217 $$

- A 32-bit signed integer can represent all values from -2^{31} to $2^{31}-1$. So $16{,}777{,}217$ can be represented.

- A 32-bit float (IEEE 754 single precision) has:

  - 23 fraction bits + 1 hidden bit = 24 bits of precision.
  - This means every integer up to 2^{24} can be exactly represented.
  - Once an integer requires more than 24 bits to write in binary, precision is lost.

Floating-point representation of $2^{24}+1$

- Binary form:

  $$ 2^{24}+1 = 1.0000\ldots000001 \times 2^{24} $$

  (23 zeros then a trailing 1).

- Mantissa: only 23 fraction bits are stored. The trailing 1 at the 24th place does not fit.

Thus the stored value becomes:

$$ 1.0000\ldots0000 \times 2^{24} = 2^{24} = 16{,}777{,}216 $$

So $16{,}777{,}217$ exists in 32-bit signed int but not in float.
In float it is rounded off to $16{,}777{,}216$, losing the $+1$.

Q6

Let float32 numbers:

$$ a=-2^{24},\quad b=2^{24},\quad c=1. $$

- **Left-associate** $(a+b)+c$:

  $$ ((-2^{24})+2^{24})+1 = 0+1 = 1 \quad(\text{exact}). $$

- **Right-associate** $a+(b+c)$:

  $$ b+c = 2^{24}+1 ;\xrightarrow{\text{round}}; 2^{24}\quad(\text{As seen previously}), $$

  so

  $$ a+(b+c)=(-2^{24})+2^{24}=0. $$

Hence,

$$ (a+b)+c = 1 ;\neq; a+(b+c) = 0. $$

Thus, addition in floating point numbers is not always associative.