Assignment 2 – SOLUTION

Assignment 2 - Objective:

In this assignment, you will gain familiarity with:

- IEEE floating point representation

Remember to

- show your work (as illustrated in lectures), and
- to make sure the pdf or jpeg documents you upload are of good quality, i.e., easy to read, therefore easy to mark! :)

Marking scheme:

- This assignment will be marked as follows:
  - Questions 1 and 2 will be marked for correctness.
- The amount of marks for each question is indicated as part of the question.
- A solution will be posted on Monday after the due date.

Due: Friday January 27 at 23:59:59 on Crowdmark.

Late assignments will receive a grade of 0, but they will be marked (if they are submitted before the solutions are posted on Monday) in order to provide feedback to the student.

Enjoy!

Q1 a. (8 points)

Floating point conversion and Rounding

Convert the following real numbers R to IEEE (Standard 754) floating point representation (single precision), clearly showing the effect of rounding on the **frac** (mantissa) if rounding occurs. Then express your final answer in binary and in hexadecimal form.

I. 0.001111111_2

II. 3.1416015625_{10}

III. -0.9_{10}

IV. $1/3_{10}$ (a third)

a. i. 0.001111111_2

$V = (-1)^s \cdot M \cdot 2^E$
$M = 1 + \text{frac}$
$E = \exp - \underbrace{\text{bias}}_{127}$

- Normalized: $1.111111_2 \times 2^{-3}$ (implied 1)
- $s = 0$ (the number is positive)
- $M = 1 + \text{frac}$
- $\text{frac} = 111111\underbrace{00..00}_{17 \times 0\text{'s}}$ (padding)

No rounding occurred

- $E = \exp - 127$ (bias $= 2^{k-1} - 1 = 2^{8-1} - 1 = 2^7 - 1 = 127$)

  $-3 + 127 = \exp$

  $\exp = 124_{10} \rightarrow 01111100_2$

$\therefore$ 0 01111100 11111100000000000000000

0x3E7E0000

a. ii. 3.1416015625_{10}

- in binary: 11.0010010001_2

  normalized: $1.10010010001_2 \times 2^{+1}$

- $S = 0$
- $M = 1 + \text{frac}$
- $\text{frac} = 10010010001\underbrace{\overset{\text{padding}}{00..0}}_{12 \times 0\text{'s}}$

$E = \exp - 127$

$1 + 127 = \exp$

$\exp = 128_{10} \rightarrow 10000000_2$

No rounding occurred

$.1416015625 \times 2 = 0 + 0.283203125 \rightarrow 0$

$.283203125 \times 2 = 0 + 0.56640625 \rightarrow 0$

$.56640625 \times 2 = 1 + 0.1328125 \rightarrow 1$

$.1328125 \times 2 = 0 + 0.265625 \rightarrow 0$

$.265625 \times 2 = 0 + 0.53125 \rightarrow 0$

$.53125 \times 2 = 1 + 0.0625 \rightarrow 1$

$.0625 \times 2 = 0 + 0.125 \rightarrow 0$

$.125 \times 2 = 0 + 0.25 \rightarrow 0$

$.25 \times 2 = 0 + 0.5 \rightarrow 0$

$.5 \times 2 = 1 + 0.0 \rightarrow 1$

$\therefore$ 0 1000000 0100 1001 0001 0000 0000 0000

0x40491000

2. iii. -0.9_{10}

- in binary: $-0.1\overline{1100}_2$
- normalized: $-1.\overline{1100}_2 \times 2^{-1}$
- $S = 1$ (the number is negative)
- $M = 1 + \text{frac}$

$.9 \times 2 = 1 + 0.8 \rightarrow 1$
$.8 \times 2 = 1 + 0.6 \rightarrow 1$
$.6 \times 2 = 1 + 0.2 \rightarrow 1$
$.2 \times 2 = 0 + 0.4 \rightarrow 0$
$.4 \times 2 = 0 + 0.8 \rightarrow 0$
$.8 \times 2 = 1 + 0.6 \rightarrow 1$
$.6 \times 2 = 1 + 0.2 \rightarrow 1$
$.2 \times 2 = 0 + 0.4 \rightarrow 0$
$.4 \times 2 = 0 + 0.8 \rightarrow 0$
$.8 \times 2 = 1 + 0.6 \rightarrow 1$
⋮

repeating pattern

→ Rounding frac:

23rd bit

rounding position

- frac = 11001100110011001100110 | ~~0 ... $\overline{1100}$ ...~~

$< \frac{1}{2}$ of rounding position (i.e. $\frac{1}{2^{23}}$)

∴ **Round down** -> discard the bits to the right-hand side of rounding position

- $E = exp - 127$
  $-1 + 127 = exp$
  $exp = 126_{10} \rightarrow 01111110_2$

∴ 1 0111 1110 1100 1100 1100 1100 1100 110

0x BF666666

2\. iv. $\frac{1}{3_{10}} = 0.\dot{3}$

to convert into binary, let's expand then round $\frac{1}{3}$ into $0.333 \rightarrow$

$0.333 \times 2 = 0 + 0.666 \rightarrow 0$
$0.666 \times 2 = 1 + 0.332 \rightarrow 1$
$0.332 \times 2 = 0 + 0.664 \rightarrow 0$
$0.664 \times 2 = 1 + 0.328 \rightarrow 1$
$0.328 \times 2 = 0 + 0.656 \rightarrow 0$
$0.656 \times 2 = 1 + 0.312 \rightarrow 1$
$0.312 \times 2 = 0 + 0.624 \rightarrow 0$
$0.624 \times 2 = 1 + 0.248 \rightarrow 1$
...

repeating pattern

- in binary: $0.\overline{01}_2$
- normalized: $0.010101\ldots \times 2^0$
  $1.0101\ldots \times 2^{-2}$
- $S = 0$
- $M = 1 + frac$
- $frac = 0101010101010101010101\phi|\underbrace{\cancel{01\ldots}}_{> \frac{1}{2} \text{ of rounding position}}$

23rd bit

$\therefore$ **Round up** -> add 1 to the bit at rounding position then discard the bits to the right-hand side of rounding position

- $E = exp - 127$
  $-2 + 127 = 125_{10}$
  $exp = 125_{10} \rightarrow 01111101_2$

$\therefore$ 0 01111101 01010101010101010101011

0x3EAAAAAB

Q1 b. (2 points)

Floating point conversion and Rounding

Convert 0x4AEA4C1A from IEEE floating point representation (single precision) to a fractional decimal number (i.e., a real number R).

b. 0x4AEA4C1A

$\underset{s}{0}\,\underbrace{10010101}_{exp}\,\underbrace{11010100100110000011010}_{frac}$

∴ positive number

- $E = exp - bias = 149 - 127 = 22 \quad \therefore 2^{22}$

10010101
$2^7 + 2^4 + 2^2 + 2^0$
$= 128 + 16 + 4 + 1$
$= 149_{10}$

- $M = 1 + .11010100100110000011010$
$= 1.\underbrace{11010100100110000011010}_{} \cong 1.830447197$

0.5 + 0.25 + 0.0625 + 0.015625 + 0.001953125 +
0.000244140625 + 0.0001220703125 +
0.000001907348633 + 0.0000009536743164
0.0000002384185791 ≅ 0.83044743844

$V = (-1)^0 \; 1.83044743844 \cdot 2^{22}$

R => 7677453.0

Q1 c. (2 points)

Floating point conversion and Rounding

Round the following binary numbers (rounding position is **bolded** – it is the bit at the 2^{-4} position) following the rounding rules of the IEEE floating point representation.

I. $1.001\mathbf{1}111_2$

II. $1.100\mathbf{1}001_2$

III. $1.011\mathbf{1}100_2$

IV. $1.011\mathbf{0}100_2$

For each of the four (4) resulting rounded binary numbers, indicate

- what type of rounding you performed and
- compute the value that is either added to or subtracted from the initial binary number (listed above) as a result of the rounding process. In other words, compute the error introduced by the rounding (approximation) process.

rounding position

i. $1.0011\cancel{111}_2$

> half way (half way is $1.0011\underline{100}_2$)

So we **round up** -> add 1 to the bit at rounding position then discard the bits to the right-hand side of rounding position

$\boxed{\therefore 1.0100_2}$

The error introduced by the rounding process is

$$\begin{array}{rl} 1.0100000_2 \rightarrow & 1.2500000_{10} \\ -\ 1.0011111_2 \rightarrow & \underline{-\ 1.2421875_{10}} \\ & 0.0078125_{10} \end{array}$$

i.e., 0.0078125_{10} has been **added** to the original value 1.0011111_2 as part of the rounding process.

---

ii. $1.1001\underline{001}_2$ ← rounding position

< half way (half way is $1.1001\underline{100}_2$)

So we **round down** -> discard the bits to the right-hand side of rounding position

$\boxed{\therefore 1.1001_2}$

The error introduced by the rounding process is

$$\begin{array}{l} 1.1001001_2 \\ \underline{-\ 1.1001000_2} \\ 0.0000001_2 \rightarrow \ 7.8125_{10} \times 10^{-3} \end{array}$$

i.e., 0.0078125_{10} has been **subtracted** from the original value 1.1001001_2 as part of the rounding process.

iii. $1.0111|100_2$ ← rounding position

is exactly half way

so we **round to even number** -> we add 1 to the bit at rounding position (this has the same effect as rounding to closest even number, i.e., a number with a zero (0) bit in the rounding position) then we discard the bits to the right-hand side of rounding position

$\boxed{\therefore 1.1000_2}$

The error introduced by the rounding process is

$$\begin{array}{rcr} 1.1000000_2 & \rightarrow & 1.50000_{10} \\ -\ 1.0111100_2 & \rightarrow & \underline{-\ 1.46875_{10}} \\ & & 0.03125_{10} \end{array}$$

i.e., 0.03125_{10} has been **added** to the original value 1.0111100_2 as part of the rounding process.

<u>NOTE</u>: Could 1.0111100_2 be rounded to 1.0110_2 instead of 1.1000_2?

No, because the error ($1.0111100_2 - 1.0110_2$) is larger ($1.46875_{10} - 1.375_{10} = 0.09375_{10}$) So 1.0110_2 is not the closest even number to our original number 1.0111100_2.

rounding position

iv. $1.0110\underline{100}_2$

is exactly half way

so we **round to even number**. An even binary number is a number with a zero (0) bit. Since the bit at the rounding position is already zero (0), i.e., even, the only thing we do is discard the bits to the right-hand side of the rounding position

$\therefore \boxed{1.0110_2}$

The error introduced by the rounding process is

$$\begin{array}{r} 1.0110100_2 \\ \underline{-\ 1.0110000_2} \\ 0.0000100_2 \end{array} \rightarrow 0.03125_{10}$$

i.e., 0.03125_{10} has been **subtracted** from the original value 1.0110100_2 as part of the rounding process.

Q2 a. (4 points)

IEEE Standard 754 Encoding Scheme with smaller w

Creating smaller hypothetical floating-point representations based on the IEEE floating point format allows us to investigate this encoding scheme more easily, since the numbers are easier to compute and manipulate.

Download the **A2_Q2_Table.pdf** or **A2_Q2_Table.docx** from our course web site (under Assignment 2) and open the file with the format you would like to work with: pdf or Word.

The table lists several fractional decimal numbers represented as 6-bit IEEE-like floating-point numbers (w = 6). The format of these 6-bit floating-point numbers is as follows:

- One (1) bit is used to express for the sign (**s**),
- Three (3) bits are used to express **exp** (k = 3),
- Two (2) bits are used to represent **frac** (n = 2),
- in the following order: **s exp frac**.

Complete the table (the same way as in Figure 2.35 in our textbook), i.e., fill in the white table cells then answer the questions Q2 b. to i.

Tip: Have a look at Figure 2.35 in our textbook, which illustrates a similar table for a hypothetical 8-bit IEEE-like floating-point format. This will give you an idea of how to complete the table. Also, Figure 2.34 displays the complete range of these 6-bit IEEE-like floating point numbers as well as their values between -1.0 and 1.0. This diagram may be helpful when you are checking your work.

* Notice the smooth transition from 3/16 to 4/16

		Exponent			Fraction		Value		
Description	**Bit representation**	**exp**	**E**	**2^E**	**frac**	**M**	**M 2^E**	**V**	**Decimal**
zero	0 000 00	0	-2	1/4	0/4	0/4	0/16	0	0.0
Smallest positive denormalized	0 000 01	0	-2	1/4	¼	¼	1/16	1/16	0.0625
	0 000 10	0	-2	1/4	2/4 = ½	2/4 = ½	2/16	2/16	0.125
Largest positive denormalized	**0 000 11**	**0**	**-2**	**1/4**	**¾**	**¾**	**3/16**	**3/16**	**0.1875**
Smallest positive normalized	**0 001 00**	**1**	**-2**	**1/4**	**0/4**	**4/4 = 1**	**4/16**	**4/16**	**0.25**
	0 001 01	1	-2	1/4	¼	5/4	5/16	5/16	0.3125
	0 001 10	1	-2	1/4	2/4 = ½	6/4	6/16	6/16	0.375
	0 001 11	1	-2	1/4	¾	7/4	7/16	7/16	0.4375
	0 010 00	2	-1	1/2	0/4	4/4 = 1	4/8	4/8	0.5
	0 010 01	2	-1	1/2	¼	5/4	5/8	5/8	0.625
	0 010 10	2	-1	1/2	2/4 = ½	6/4	6/8	6/8	0.75
	0 010 11	2	-1	1/2	¾	7/4	7/8	7/8	0.875
One	0 011 00	3	0	1	0/4	4/4 = 1	4/4	4/4	1.0
	0 011 01	3	0	1	¼	5/4	5/4	5/4	1.25

	0 011 10	3	0	1	2/4 = ½	6/4	6/4	6/4	1.5
	0 011 11	3	0	1	¾	7/4	7/4	7/4	1.75
	0 100 00	4	1	2	0/4	4/4 = 1	8/4	8/4	2
	0 100 01	4	1	2	¼	5/4	10/4	10/4	2.5
	0 100 10	4	1	2	2/4 = ½	6/4	12/4	12/4	3
	0 100 11	4	1	2	¾	7/4	14/4	14/4	3.5
	0 101 00	5	2	4	0/4	4/4 = 1	16/4	16/4	4
	0 101 01	5	2	4	¼	5/4	20/4	20/4	5
	0 101 10	5	2	4	2/4 = ½	6/4	24/4	24/4	6
	0 101 11	5	2	4	¾	7/4	28/4	28/4	7
	0 110 00	6	3	8	0/4	4/4 = 1	32/4	32/4	8
	0 110 01	6	3	8	¼	5/4	40/4	40/4	10
	0 110 10	6	3	8	2/4 = ½	6/4	48/4	48/4	12
Largest positive normalized	0 110 11	6	3	8	¾	7/4	56/4	56/4	14
+ Infinity	0 111 00							∞	

NaN	**Bonus: 0.5 marks!** 0 111 01 0 111 10 0 111 11							NaN	

Q2 b. (1 point)

What is the value of the bias?

bias = $2^{k-1}-1$ and since $k=3$ then bias $= 2^{3-1}-1 = 2^2-1 = 4-1 = 3$

Q2 c. (1 point)

Consider two adjacent denormalized numbers. How far apart are they? Expressed this difference as a fractional decimal number (i.e., a real number R).

$\Delta_d = \frac{1}{16} = 0.0625$

Q2 d. (1 point)

Consider two adjacent normalized numbers with the **exp** field set to 001. How far apart are they? Expressed this difference as a decimal number.

The answer is listed as part of the answer to question Q2 f.

Q2 e. (1 point)

Consider two adjacent normalized numbers with the **exp** field set to 010. How far apart are they? Expressed this difference as a decimal number

The answer is listed as part of the answer to question Q2 f.

Q2 f. (1 point)

Consider two adjacent normalized numbers with the **exp** field set to 011. How far apart are they? Expressed this difference as a decimal number.

$$\left.\begin{array}{l} \Delta_{001} \rightarrow \frac{1}{16} = 0.0625 \\ \Delta_{010} \rightarrow \frac{2}{16} = 0.125 \\ \Delta_{011} \rightarrow \frac{4}{16} = 0.25 \end{array}\right\} \text{obtained by calculation}$$

Q2 g. (1 point)

Without doing any calculations, can you guess how far apart are two adjacent normalized numbers ...

- with the **exp** field set to 100?
- with the **exp** field set to 101?
- with the **exp** field set to 110?

$$\left.\begin{array}{l}\Delta_{100} \rightarrow \frac{8}{16} = 0.5 \\ \Delta_{101} \rightarrow \frac{16}{16} = 1 \\ \Delta_{110} \rightarrow \frac{32}{16} = 2\end{array}\right\}$$

$M \cdot 2^E$
↓
increases by a power of 2

Q2 h. (1.5 point)

What is the "range" (not contiguous) of fractional decimal numbers that can be represented using this 6-bit IEEE-like floating-point representation?

"range" of real numbers → [-14.0 .. 14.0] not considering ± ∞ and NaN
(since it is not a continuous range)

Q2 i. (1.5 point)

What is the range of the normalized exponent **E** (E found in the equation $v = (-1)^s M 2^E$) which can be represented by this 6-bit IEEE-like floating-point representation?

Range of the normalized exponent **E**

range of E -> [-2 .. 3]

⇓

exp -> [001 .. 110]

denormalized exponent **E**

exp = 000 -> E = -2 } smooth transition

001 -> E = -2

010 -> E = -1

011 -> E = 0

100 -> E = 1

101 -> E = 2

110 -> E = 3

111 -> ± ∞, NaN

Range of the normalized exponent **E**

Q2 j. (1 point)

Give an example of a fractional decimal numbers that cannot be represented using this 6-bit IEEE-like floating-point representation, but is within the “range” of representable values, which you expressed as your answer to Q2 h. above.

11.0 cannot be represented but it is within the range

Q2 k. (2 points)

Give an example of a real number that would overflow if we were trying to represent it using this 6-bit IEEE-like floating-point representation.

Then convert this real number into a 6-bit IEEE-like floating-point representation and clearly indicate that it overflows.

16.0_{10} → in binary: 10000
normalized: $1.0000 \times 2^{+4}$
$S = 0$
$E = exp - bias \Rightarrow 4 + 3 = 7 \therefore exp = 7 \rightarrow 111_2$
However, exp 111 is outside exp's range and therefore would overflow.

Q2 l. (2 points)

How close is the value of the **frac** of the largest normalized number to 1? In other words, how close is **M** to 2? Yet another way of phrasing this question would be to ask: what is the value of ε (epsilon) in this expression 1 <= M <= 2 - ε ? Express your answer as a fractional decimal number (i.e., a real number R).

<u>Answer:</u>

The value of the **`frac`** of the largest normalized number (14) is .11 -> ¾ = 0.75_{10}

How close is the value of the **`frac`** of the largest normalized number (14) to 1 -> $1.00_2 – 0.11_2 = 0.01_2$ = ¼ = 0.25_{10}

So, ε (epsilon) is ¼ = 0.25_{10}