HW2 Elijah Williams

Problem 1:

Recall that double-precision format can be used to represent any real number $x \in \mathbb{R}$ of the form:

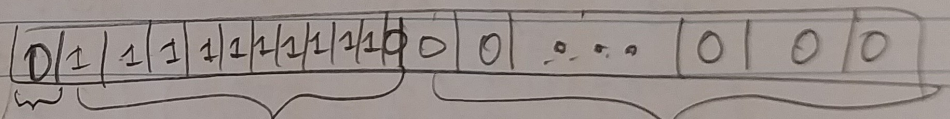
$$x = (-1)^s \times 2^{c-1023} \times (1.f)_2$$

where $\begin{cases} s \in \{0,1\} \\ 0 < c < 2047 \rightarrow c - 1023 \in [-1022, 1023] \\ 1 \leq 1.f \leq 2 - 2^{52} \end{cases}$

(a) The largest machine number we can represent is:

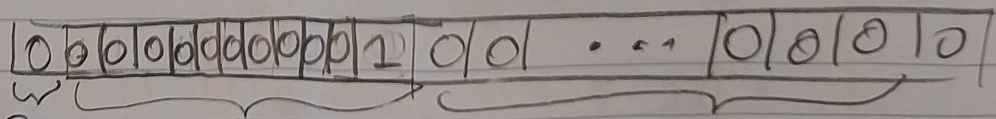
$$(1) \times 2^{1023} \times (2 - 2^{52}) < 2^{1024}$$

So $\boxed{m_1 = 1023}$

$2^{1023} \equiv$ 

(b) Similarly smallest sized machine number is

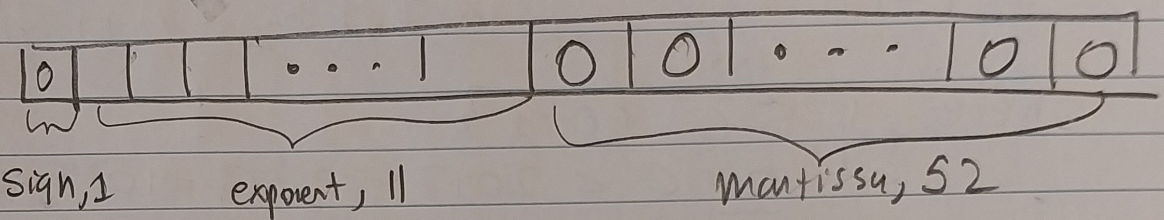
$$(1) \times 2^{-1022} \times (1.0)_2 = 2^{-1022}$$

$2^{-1022} \equiv$ 

So $\boxed{m_s = -1022}$

(c) Suppose $-1022 \leq m \leq 1023$

- Then 2^m is a machine number: that looks like:



- If $m \neq m_g$, then the machine number immediately to the right of 2^m is

$$r = \boxed{(1+2^{-52})2^m}$$

- If $m \neq m_s$, then the machine number to the left of 2^m is:

$$l = \boxed{2^{m-1}(2-2^{-52})} = 2^m - 2^{-53+m}$$

$$\begin{cases} 2^m - l = 2^{-53+m} \\ r - 2^m = 2^{-52+m} \end{cases}$$

- The number immediately to the left is closer than the number immediately to the right by a factor of 2

(2)

Idea: When we order the numbers in a list in terms of magnitude, we can decrease the roundoff error of the sum.

(a) See Pictures Below

(b) This is not always the case.
Suppose we are working on a decimal machine with two digits allocated to the mantissa.

$$\text{List} = [.25\times10^{-2}, .34\times10^{-4}, .51\times10^{-5}, .61\times10^{-}$$

$$\text{Sum(List)} = .31491$$

Suppose addition take place in a 4 digit accumulator. Then,

Unordered Sum:

$$\begin{array}{r} .2500\times10^{-2} \\ +\ .0034\times10^{-2} \\ \hline .2534\times10^{-2} \end{array}$$

$$= fl() \Rightarrow .25\times10^{-2}$$

$$\begin{array}{r} .25\times10^{-2} \\ +\ .0005\times10^{-2} \\ \hline .2505\times10^{-2} \end{array}$$

$$fl() \Rightarrow .2500\ 10^{-2}$$

$$\begin{array}{r} .2500\ 10^{-2} \\ +\ .0610\times10^{-2} \\ \hline .3110\times10^{-2} \end{array}$$

$$fl() .31\times10^{-2}$$

Rounding Error $.31491 - .31 = \boxed{.491}$

Ordered Sum: $[.51\times10^{-5}, .34\times10^{-4}, .61\times10^{-3}, .25\times10^{-2}]$

$$
\begin{array}{rl}
 & .0510 \times 10^{-4} \\
+ & .3400 \times 10^{-4} \\
\hline
 & .3910 \times 10^{-4} \\
fl() \Rightarrow & .0390 \times 10^{-3} \\
+ & .6100 \times 10^{-3} \\
\hline
 & .6490 \times 10^{-3} \\
fl() \Rightarrow & .0650 \times 10^{-2} \\
+ & .2500 \times 10^{-2} \\
\hline
 & .3150 \times 10^{-2} \\
\Rightarrow & .32 \times 10^{-2}
\end{array}
$$

Rounding Error:

$.31491 - .32 = \boxed{-.509}$

The roundoff error for the ordered sum is larger in magnitude than the unordered sum.

3.) How many machine numbers are there in single-precision arithmetic?

Suppose $x \in \mathbb{R}$ is a single-precision machine number (non zero)

Then $x = (-1)^s \times 2^{c-127} \times (1.f)_2$

where $s \in \{0, 1\}$

$c \in \{1, 2, \ldots, 254\}$

$f = (.b_1 b_2 b_3 \ldots b_{52})$

Counting the possibilities:

$$\underbrace{2}_{} \cdot \underbrace{254}_{} \cdot \underbrace{2^{52}}_{}$$

$$= 2^{53} \cdot 254 \text{ single-preison numbers}$$

excluding $\pm 0, \pm \infty$

(9) In mathematics,

$$(a+b)+c = a+(b+c)$$

However this is not always true when adding machine numbers

ex)

- In my solution for problem 2, I ~~for~~ already found 3 floating point numbers for which this is true:

- Unsorted_list = [a, b, c]
- Sorted_list = [c, b, a]

$$\boxed{(a+b)+c \neq (c+b)+a}$$

(5) $\sin(x) + \cos(x) - 1$, x near zero

- When x is near zero,
  $\sin(x) \approx 0$
  $\cos(x) \approx 1$
  and we get subtractive cancellation.

We can try rationalization:

$$\left(\sin(x) + \cos(x) - 1\right) \cdot \left(\frac{\sin(x) + \cos(x) + 1}{\sin(x) + \cos(x) + 1}\right)$$

$$= \frac{\sin^2(x) + \sin(x)\cos(x) + \sin(x) + \cos^2(x) + \sin(x)\cos(x) + \cos(x) - 1 - \sin(x) - \cos(x) - 1}{\sin(x) + \cos(x) + 1}$$

$$= \boxed{\frac{\sin(2x)}{\sin(x) + \cos(x) + 1}}$$

No subtractive cancellation

Problem 6

- The purpose of this code is to find out what the smallest positive number we can add to 1 that causes the compiler to round up to another machine number.

  y = Machine epsilon = unit roundoff error

- I calculated

$$y = 2.2 \times 10^{-16}$$

This the same machine epsilon we calculated for double precision numbers

$$\varepsilon = 2^{-52} \approx 2.2 \times 10^{-16}$$

Problem 7

(a) Evaluate $f(x) = (x - \sin x)/x^3$, x close to zero

- See Program

(b) We can eliminate subtractive cancellation by dividing by x:

$$f(x) = \frac{x}{x^3} - \frac{\sin(x)}{x^3} = \frac{1}{x^2} - \frac{\sin(x)}{x^3}$$

```
>> HW2_P2
List: [678.7351548577735230, 0.7577401305783334, 0.0007431324681249]
Exact Sum: 679.49363812082003732939483597874641

Sorted Sum: 679.49363812082003732939483597874641
Sorted Error: 0.00000000000005600196955474201488

Unsorted Sum: 679.49363812081992364255711436271667
Unsorted Error: -0.00000000000005768486816687401486
>>
```

```
>> HW2_P7
x             f_old         abs_err         f_new         abs_err
1.000000000 0.158529015 0.008137651 0.158529015 0.008137651
0.100000000 0.166583353 0.000083313 0.166583353 0.000083313
0.010000000 0.166665833 0.000000833 0.166665833 0.000000833
0.001000000 0.166666658 0.000000008 0.166666658 0.000000008
0.000100000 0.166666661 0.000000005 0.166666667 0.000000000
0.000010000 0.166667285 0.000000618 0.166666667 0.000000000
0.000001000 0.166653732 0.000012934 0.166666667 0.000000000
0.000000100 0.172053567 0.005386901 0.166666667 0.000000000
0.000000010 0.000000000 0.166666667 0.166666667 0.000000000
0.000000001 0.000000000 0.166666667 0.166666667 0.000000000
0.000000000 0.000000000 0.166666667 0.166666667 0.000000000
0.000000000 0.000000000 0.166666667 0.166666667 0.000000000
0.000000000 0.000000000 0.166666667 0.166666667 0.000000000
0.000000000 0.000000000 0.166666667 0.166666667 0.000000000
0.000000000 0.000000000 0.166666667 0.166666667 0.000000000
0.000000000 0.000000000 0.166666667 0.166666667 0.000000000
```