MATH50003 (2022–23)

Problem Sheet 1

This problem sheet tests the representation of numbers on the computer, using modular and floating point arithmetic.

Please complete the problems using pen-and-paper, though some can be verified using Julia.

Problem 1 With 8-bit signed integers, what are the bits for the following: $10, 120, -10$.

SOLUTION We can find the binary digits by repeatedly subtracting the largest power of 2 less than a number until we reach 0, e.g. $10 - 2^3 - 2 = 0$ implies $10 = (1010)_2$. Thus the bits are:

In [1]:
```julia
using ColorBitstring
printlnbits(Int8(10))
```

```
00001010
```

Similarly,

$$120 = 2^6 + 2^5 + 2^4 + 2^3 = (1111000)_2$$

Thus the bits are (meant to be deduced by hand but we use Julia to confirm):

In [2]:
```julia
printlnbits(Int8(120))
```

```
01111000
```

For negative numbers we perform the same trick but adding 2^p to make it positive, e.g.,

$$-10 = 2^8 - 10 (\mathrm{mod} 2^8) = 246 = 2^7 + 2^6 + 2^5 + 2^4 + 2^2 + 2 = (11110110)_2$$

This the bits are:

In [3]:
```julia
printlnbits(Int8(-10))
```

```
11110110
```

END

Problem 2 What is π to 5 binary places? Hint: recall that $\pi \approx 3.14$.

SOLUTION Note that

In [4]:
```julia
3 + 1/8  + 1/64
```

Out[4]: 3.140625

which has the binary representation $(11.001001)_2$. Indeed:

In [5]:
```julia
printbits(Float16(π))
```

```
0100001001001000
```

Instead of simply guessing the above representation we can instead continuously subtract the largest powers 2 which do not result in a negative number. For π the procedure then finds that we can write

$$\pi - 1 * 2^1 - 1 * 2^0 - 1 * 2^{-3} - 1 * 2^{-6} \ldots$$

END

Problem 3 What are the single precision F_{32} (`Float32`) floating point representations for the following:

$$2, 31, 32, 23/4, (23/4) \times 2^{100}$$

SOLUTION Recall that we have `σ,Q,S = 127,8,23`. Thus we write

$$2 = 2^{128-127} * (1.00000000000000000000000)_2$$

The exponent bits are those of

$$128 = 2^7 = (10000000)_2$$

Hence we get

In [6]:
```julia
printlnbits(2f0)
```

```
01000000000000000000000000000000
```

We write

$$31 = (11111)_2 = 2^{131-127} * (1.1111)_2$$

And note that $131 = (10000011)_2$ Hence we have:

In [7]:
```julia
printlnbits(31f0)
```

```
01000001111110000000000000000000
```

On the other hand,

$$32 = (100000)_2 = 2^{132-127}$$

and $132 = (10000100)_2$ hence:

In [8]:
```julia
printlnbits(32f0)
```

```
01000010000000000000000000000000
```

Note that

$$23/4 = 2^{-2} * (10111)_2 = 2^{129-127} * (1.0111)_2$$

and $129 = (10000001)_2$ hence we get:

In [9]:
```
printlnbits(23f0/4)
```

```
01000000101110000000000000000000
```

Finally,

$$23/4 * 2^{100} = 2^{229-127} * (1.0111)_2$$

and $229 = (11100101)_2$ giving us:

In [10]:
```
printlnbits(23f0/4 * 2f0^100)
```

```
01110010101110000000000000000000
```

END

Problem 4 Let $m(y) = \min\{x \in F_{32} : x > y\}$ be the smallest single precision number greater than y. What is $m(2) - 2$ and $m(1024) - 1024$?

SOLUTION The next float after 2 is $2 * (1 + 2^{-23})$ hence we get $m(2) - 2 = 2^{-22}$:

In [11]:
```
nextfloat(2f0) - 2, 2^(-22)
```

Out[11]:
```
(2.3841858f-7, 2.384185791015625e-7)
```

similarly, for $1024 = 2^{10}$ we find that the difference $m(1024) - 1024$ is $2^{10-23} = 2^{-13}$:

In [12]:
```
nextfloat(1024f0) - 1024, 2^(-13)
```

Out[12]:
```
(0.00012207031f0, 0.0001220703125)
```

END

Problem 5 Suppose $x = 1.25$ and consider 16-bit floating point arithmetic (F_{16}). What is the error in approximating x by the nearest float point number $\mathrm{fl}(x)$? What is the error in approximating $2x$, $x/2$, $x + 2$ and $x - 2$ by $2 \otimes x$, $x \oslash 2$, $x \oplus 2$ and $x \ominus 2$?

SOLUTION None of these computations have errors since they are all exactly representable as floating point numbers. **END**

Problem 6 For what floating point numbers is $x \oslash 2 \neq x/2$ and $x \oplus 2 \neq x + 2$?

SOLUTION

Consider a normal $x = \pm 2^{q-\sigma}(1.b_1 \ldots b_S)_2$. Provided $q > 1$ we have

$$x \oslash 2 = x/2 = \pm 2^{q-\sigma-1}(1.b_1 \ldots b_S)_2$$

However, if $q = 1$ we lose a bit as we shift:

$$x \oslash 2 = \pm 2^{1-\sigma}(0.b_1 \ldots b_{S-1})_2$$

and the property will be satisfy if $b_S = 1$. Similarily if we are sub-normal, $x = \pm 2^{1-\sigma}(0.b_1 \ldots b_S)_2$ and we have

$$x \oslash 2 = \pm 2^{1-\sigma}(0.0b_1 \ldots b_{S-1})_2$$

and the property will be satisfy if $b_S = 1$. (Or `NaN`.)

Here are two examples:

In [13]:
```julia
# normal number with q = 1 and last bit 1
x = reinterpret(Float16, 0b0000010000000011)
@test x/2 ≠ Float64(x)/2 # Float64 can exactly represent x/2
# normal number with q = 1 and last bit 0
x = reinterpret(Float16, 0b0000010000000010)
@test x/2 == Float64(x)/2
# sub-normal number with q = 1 and last bit 1
x = reinterpret(Float16, 0b0000000000000011)
@test x/2 ≠ Float64(x)/2 # Float64 can exactly represent x/2
# sub-normal number with q = 1 and last bit 0
x = reinterpret(Float16, 0b0000000000000010)
@test x/2 == Float64(x)/2 # Float64 can exactly represent x/2
```

Out[13]: **Test Passed**

For the second part, first assume $x > 0$ and write

$$x = 2^j(1.b_1 \ldots b_S)_2$$

Lets begin with the case $j = 0$. We then get:

$$2 \oplus x = \mathrm{fl}(2(1.1b_1 \ldots b_S)_2) = 2 + x$$

if and only if $b_S = 0$:

In [14]:
```julia
x =reinterpret(Float16, 0b0011110000000010) # b_S = 0
@test x+2 == Float64(x)+2
x =reinterpret(Float16, 0b0011110000000011) # b_S = 1
@test x+2 ≠ Float64(x)+2
```

Out[14]: **Test Passed**

If $j < 0$ then we have

$$2 \otimes x = \mathrm{fl}(2(1.\underbrace{0\ldots0}_{-j \text{ zeros}}1b_1 \ldots b_S)_2)$$

hence we need $b_{S-j} = \cdots = b_S = 0$:

In [15]:
```julia
# this has j = -1
x =reinterpret(Float16, 0b0011100000000100) # b_S = B_{S-1} = 0
@test x+2 == Float64(x)+2
x =reinterpret(Float16, 0b0011100000000110) # b_S = 0, B_{S-1} ≠ 0
@test x+2 ≠ Float64(x)+2
```

Out[15]: **Test Passed**

If $j = 1$ then $b_S = 0$ iff $x \oplus 2 = x$ as we are incrementing the exponent and shifting the significand. If $1 < j < S + 2$ then $b_k = 0$ for some $1 \leq k \leq j - 1$ implies that $x \oplus 2 = x$. Otherwise if $b_1 = \cdots = b_{j-1} = 1$ then $b_S = 0$ iff $x \oplus 2 = x$ as we are incrementing the exponent and shifting the significand. If $j \geq S + 2$ then

$$x + 2 = 2^j (1.b_1 \ldots b_S \underbrace{0 \ldots 0}_{j-S-2 \text{ times}} 10)_2$$

and when rounded the trailing 10 will be removed. (Of course `Inf + 2 == Inf` and `NaN + 2 == NaN` as well.)

Now consider $x < 0$ and it helps to write $2 = (1.111111\ldots)_2$. If $j = 0$ define $\bar{b}_k = 1 - b_k$ (that is, 1 if $b_k = 0$ and 0 otherwise.) Then we have

$$2 + x = (0.\bar{b}_1 \ldots \bar{b}_S 1111\ldots)_2 = (0.\bar{b}_1 \ldots \bar{b}_S)_2 + 2^{-S} \in F$$

as we have at most S non-zero bits after the first decimal point. If $j = -1$ we have

$$2 + x = 1.0\bar{b}_1 \ldots \bar{b}_S 1111\ldots = (1.0\bar{b}_1 \ldots \bar{b}_S)_2 + 2^{-1-S}$$

and $\bar{b}_S = 1$ (i.e. $b_S = 0$) iff $2 + x = 2 \oplus x$ (since then the the last bit is zero). Similarly, if $j < -1$ we have

$$2 + x = 1.\underbrace{1\ldots1}_{-j-1 \text{ times}} 0\bar{b}_1 \ldots \bar{b}_S 1111\ldots = (1.\underbrace{1\ldots1}_{-j-1 \text{ times}} 0\bar{b}_1 \ldots \bar{b}_S)_2 + 2^{-j-S}$$

and $\bar{b}_{S+j+1} = \cdots = \bar{b}_S = 1$ (i.e. $b_{S+j+1} = \cdots = b_S = 0$) iff $2 + x = 2 \oplus x$. For $j = 1$ we have

$$2 \oplus x = -\mathrm{fl}((1b_1.b_2 \ldots b_S)_2 - (10)_2) = -\mathrm{fl}(b_1.b_2 \ldots b_S)_2 = -b_1.b_2 \ldots b_S = 2 + x.$$

For $j = 2$ and $b_1 = 1$ we have

$$2 \oplus x = -\mathrm{fl}((11b_2.b_3 \ldots b_S)_2 - (10)_2) = (10b_2.b_3 \ldots b_S)_2 = 2 + x$$

where if $b_1 = 0$ we have

$$2 \oplus x = -\mathrm{fl}((10b_2.b_3 \ldots b_S)_2 - (10)_2) = (1b_2.b_3 \ldots b_S)_2 = 2 + x.$$

By similar arguments, whenever $2 \leq j \leq S + 1$ we have $2 \oplus x = 2 + x$. If $j = S + 2$ the only case where $2 \oplus x = 2 + x$ is if $x = 2^{S+2}$:

$$2 \oplus x = -\mathrm{fl}((1\underbrace{0\ldots0}_{S\text{ zeros}}00)_2 - (10)_2) = -\mathrm{fl}((\underbrace{1\ldots1}_{S-1\text{ ones}}\ 10)_2) = 2^{S+2}(1.\underbrace{1\ldots1}_{S\text{ ones}})_2 = 2 + x$$

If $j \geq S + 3$ we then have $2 \oplus x \neq 2 + x$.

Finally, if $x = 0$ then $x \oplus 2 = \mathrm{fl}(2) = 2$.

END

Problem 7 What are the exact bits for $1 \oslash 5$, $1 \oslash 5 \oplus 1$ computed using half-precision arithmetic (`Float16`) (using default rounding)?

SOLUTION

From Problem 2.1 in Lab 2 we know that

$$1/5 = 2^{-3} * (1.10011001100\ldots)_2 \approx 2^{-3} * (1.1001100110)_2$$

where the $\approx$ is rounded to the nearest 10 bits (in this case rounded down). This can be shown using Geometric series:

$$(0.00110011001100\ldots)_2 = (2^{-3} + 2^{-4})(1.00010001000\ldots)_2 = (2^{-3} + 2^{-4})\sum_{k=0}^{\infty} \frac{1}{16^k}$$
$$= \frac{2^{-3} + 2^{-4}}{1 - \frac{1}{2^4}} = \frac{3}{15} = \frac{1}{5}$$

We write $-3 = 12 - 15$ hence we have $q = 12 = (01100)_2$. so we get the bits:

In [16]:
```julia
printbits(Float16(1)/5)
```

```
0011001001100110
```

Adding `1` we get:

$$1 + 2^{-3} * (1.1001100110)_2 = (1.001100110011)_2 \approx (1.0011001101)_2$$

Here we write the exponent as $0 = 15 - 15$ where $q = 15 = (01111)_2$. Thus we get:

In [17]:
```julia
printbits(1 + Float16(1)/5)
```

```
0011110011001101
```

END

Problem 8 Explain why the following does not return `1`. Can you compute the bits explicitly?

In [18]:
```julia
Float16(0.1) / (Float16(1.1) - 1)
```

Out[18]:
```
Float16(1.004)
```

SOLUTION Note that

$$\frac{1}{10} = \frac{1}{2}\frac{1}{5} = 2^{-4} * (1.10011001100\ldots)_2$$

hence we have

$$\mathrm{fl}(\frac{1}{10}) = 2^{-4} * (1.1001100110)_2$$

and

$$\mathrm{fl}(1 + \frac{1}{10}) = \mathrm{fl}(1.0001100110011\ldots) = (1.0001100110)_2$$

Thus

$$\mathrm{fl}(1.1) \ominus 1 = (0.0001100110)_2 = 2^{-4}(1.1001100000)_2$$

and hence we get

$$\mathrm{fl}(0.1) \oslash (\mathrm{fl}(1.1) \ominus 1) = \mathrm{fl}(\frac{(1.1001100110)_2}{(1.1001100000)_2}) \neq 1$$

To compute the bits explicitly, write $y = (1.10011)_2$ and divide through to get:

$$\frac{(1.1001100110)_2}{(1.10011)_2} = 1 + \frac{2^{-8}}{y} + \frac{2^{-9}}{y}$$

We then have

$$y^{-1} = \frac{32}{51} = 0.627\ldots = (0.101\ldots)_2$$

Hence

$$1 + \frac{2^{-8}}{y} + \frac{2^{-9}}{y} = 1 + (2^{-9} + 2^{-11} + \cdots) + (2^{-10} + \cdots) = (1.00000000111\ldots)_2$$

Therefore we round up (the $\ldots$ is not exactly zero but if it was it would be a tie and we would round up anyways to get a zero last bit) and get:

In [19]:
```
printlnbits(Float16(0.1) / (Float16(1.1) - 1))
```

```
0011110000000100
```

END

Problem 9 Find a bound on the *absolute error* in terms of a constant times machine epsilon ϵ_{m} for the following computations

$$(1.1 * 1.2) + 1.3$$
$$(1.1 - 1)/0.1$$

implemented using floating point arithmetic (with any precision). That is, each number is rounded first using fl and each operation is replaced by its floating point analogues $\oplus, \otimes, \ominus, \oslash$.

SOLUTION

The first problem is very similar to what we saw in lecture. Write

$$(\mathrm{fl}(1.1) \otimes \mathrm{fl}(1.2)) \oplus \mathrm{fl}(1.3) = (1.1(1+\delta_1)1.2(1+\delta_2)(1+\delta_3) + 1.3(1+\delta_4))(1+\delta_5)$$

We first write

$$1.1(1+\delta_1)1.2(1+\delta_2)(1+\delta_3) = 1.32(1+\delta_6)$$

where

$$|\delta_6| \leq |\delta_1| + |\delta_2| + |\delta_3| + |\delta_1||\delta_2| + |\delta_1||\delta_3| + |\delta_2||\delta_3| + |\delta_1||\delta_2||\delta_3| \leq 4\epsilon_{\mathrm{m}}$$

Then we have

$$1.32(1+\delta_6) + 1.3(1+\delta_4) = 2.62 + \underbrace{1.32\delta_6 + 1.3\delta_4}_{\delta_7}$$

where

$$|\delta_7| \leq 7\epsilon_{\mathrm{m}}$$

Finally,

$$(2.62 + \delta_7)(1+\delta_5) = 2.62 + \underbrace{\delta_7 + 2.62\delta_5 + \delta_7\delta_5}_{\delta_8}$$

where

$$|\delta_8| \leq 10\epsilon_{\mathrm{m}}$$

For the second part, we do:

$$(\mathrm{fl}(1.1) \ominus 1) \oslash \mathrm{fl}(0.1) = \frac{(1.1(1+\delta_1) - 1)(1+\delta_2)}{0.1(1+\delta_3)}(1+\delta_4)$$

Write

$$\frac{1}{1+\delta_3} = 1 + \delta_5$$

where

$$|\delta_5| \leq \left|\frac{\delta_3}{1+\delta_3}\right| \leq \frac{\epsilon_{\mathrm{m}}}{2}\frac{1}{1-1/2} \leq \epsilon_{\mathrm{m}}$$

using the fact that $|\delta_3| < 1/2$. Further write

$$(1+\delta_5)(1+\delta_4) = 1+\delta_6$$

where

$$|\delta_6| \leq |\delta_5| + |\delta_4| + |\delta_5||\delta_4| \leq 2\epsilon_{\mathrm{m}}$$

We also write

$$\frac{(1.1(1+\delta_1)-1)(1+\delta_2)}{0.1} = 1 + \underbrace{11\delta_1 + \delta_2 + 11\delta_1\delta_2}_{\delta_7}$$

where

$$|\delta_7| \leq 12\epsilon_{\mathrm{m}}$$

Then we get

$$(\mathrm{fl}(1.1) \ominus 1) \oslash \mathrm{fl}(0.1) = (1+\delta_7)(1+\delta_6) = 1 + \delta_7 + \delta_6 + \delta_6\delta_7$$

and the error is bounded by:

$$(12+2+34)\epsilon_{\mathrm{m}} = 48\epsilon_{\mathrm{m}}$$

This is quite pessimistic but still captures that we are on the order of ϵ_{m}.

END