Floating-point arithmetic and error analysis (AFAE)

Tutorial nº 2 - Introduction to computer arithmetic

Exercise 1 (Representation of signed integers)**.**

1. Explain 3 ways to represent a 8-bits signed integer. Give the representation of 19 and -19 for each of this 3 ways.

Exercise 2 (Representation of floating-point numbers)**.**

1. Give the representation in IEEE-754 single precision of the following numbers:
   - 13
   - 0.4375
   - -0.4375
   - $1 + 2^{-24}$
   - $1 + 2^{-24} - 2^{-25}$
   - $1 + 2^{-24} + 2^{-25}$
   - $1/7$
   - 2^{-130}
2. Let $a = 4097 = 2^{12} + 1$ and $b = 8449 = 2^{13} + 2^8 + 1$ be 2 single precision floating-point numbers. Let $c = a \otimes b$ be the floating-point number obtained by computing the product of a and b in single precision with rounding to nearest. Give the representation of c in single precision.

Exercise 3 (Problem with double rounding)**.** Let `x=0x3ff6a09e6fffcafe` and `y=0x3d8a80ffffffffff` be 2 floating-point numbers. Represented in binary, we obtain:

```
x = 1.0 1 1 0 1 0 1 0 0 0 0 0 1 0 0 1 1 1 1 0 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 1 0 1 1 1 1 1 1 1 0 _2
y = 1.1 0 1 0 1 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 _2  *  2^(-39)
```

1. Compute $x + y$ exactly.
2. Derive from question 1 what is the rounding to nearest of this sum in double precision, more precisely the significand of the representation in double precision of $x + y$.
   Derive the rounding in single precision of this double precision number.
3. Derive from question 1 the rounding in single precision of $x + y$.
   What do you notice? Explain.

Exercise 4 (Computation of square root and division)**.**

1. We recall the Newton-Raphson algorithm: $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$ to find the root of the function f from a reasonable approximation x_0.
   Apply this algorithm to compute a square root.
2. If we assume that the intial point has 4 bits of accuracy, how many iterations are needed to obtain an accuracy of 24 bits? 53 bits?
3. Explain how to use the same method to compute the division of 2 floating-point numbers?