CS1022 Tutorial #7
Floating-Point Numbers

1 Floating-Point Representation

(a) Normalise the following decimal floating point numbers:

(i) 456.789×10^3

(ii) 0.000425×10^{-2}

(b) Convert the following binary floating point numbers to decimal:

(i) 1.1

(ii) 1000.0101

(iii) 0.1111

(c) Convert the following decimal floating point numbers to binary:

(i) 15.25

(ii) 12.1875

(iii) 8.9

(iv) 0.08

(d) Normalise each of the binary floating point numbers in part (c).

(e) Show how you would store each of the normalised floating point numbers from part (d) as a 32-bit word using the IEEE 754 standard.

2 Floating-Point Representation Subroutines

(a) Write an ARM Assembly Language subroutine that will accept as parameters the fraction and exponent of a floating point number of the form $f \times 2^e$ and return a 32-bit IEE754 representation of the number. Assume that the original number is not normalised and that the fraction, f and exponent, e are signed 2's Complement numbers.

(b) Write an ARM Assembly Language subroutine that will accept as a parameter a 32-bit IEE754 encoded floating-point number. Your subroutine should decode the number and return the values f and e used to represent the number in the form $f \times 2^e$.

3 Floating-Point Addition

(a) Decode, align, add, normalise and re-encode each of the following floating point numbers encoded using the IEEE 754 standard, using the approach outlined in lectures.

(i) a=0x3FA00000, b=0x3F400000

(ii) a=0x41C40000, b=0x41960000

(iii) a=0x41A60000, b=0x3E400000

(b) Write an ARM Assembly Language subroutine that will add two IEEE754 encoded values, returning the result as another IEEE754 value. Your program should be capable of re-normalising the result but you need not consider rounding.