ARM Architecture Assignment 3

MT2016050

Problem-1

In Floating point representation we have three components

1. The Sign Bit
2. Exponent
3. Fractional Part.

Q1. Precession is one the prime attribute of any Floating point Representation. Does any of the above three components play a role in the defining the Precession of the number ? If so which are the component or Components which play the role in defining precession and how ? Explain this with example in your own words.

A1. Precision of a floating point number is defined by number of bits, not any of above. Sign bit, Exponent and fraction determines the value of the number. According to IEEE 754 standard, precision ranges are,

half-Precision 16-bit : 1 Sign bit + 5 exponent bit + 10 Fraction bit

Single-Precision 32-bit : 1 Sign bit + 8 exponent bit + 23 Fraction bit

Double-Precision 64-bit : 1 Sign bit + 11 exponent bit + 52 Fraction bit

Q2.What is Normal and de-normal values as per IEEE 754 standards explain this with the help of number line.

Here, examples are demonstrated using single precision (32-bit) format.

S	Exponent (8-bit)	Fraction (23-bit)

Normalized Form:

$$\boldsymbol{Decimal = (sign)2^{(E-127)}X\ (1 +\ 2^{-(F)})}$$

Ex. For The sequence <u>1</u> <u>01111010</u> <u>101 0000 0000 0000 0000 0000</u>,

S = 1

E = 0111 1010 = 122

F = 1 + 2^{-1} + 2^{-3} = 1.725

Decimal = $-2^{(122-127)}$ x 1.725 = -0.05390625

The problem with normalized form is, we can not represent zero, as the fraction is always add to 1.So, de-normalized form is introduced where E = 0 and

$$\boldsymbol{Decimal = (sign)2^{(-126)}X\ (0 +\ 2^{-(F)})}$$

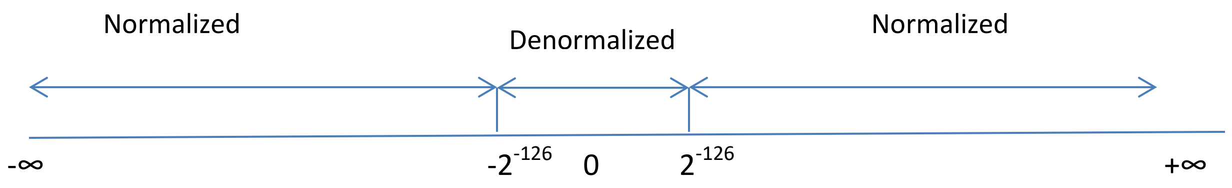
Ex. For The sequence <u>0</u> <u>0000000</u> <u>000 1000 0000 0000 0000 0000</u>,

S = 1

E = 0 ➔ denormalized form.

F = 0 + 2^{-4} = 0.0625

Decimal = $2^{(-126)}$ x 0.0625 ~= 0



Q3. In IEEE 754 specifications, which are the rounding methods ?

When the FPU performs its calculations, more digits are used than are necessary to store the numbers involved. Calculations are performed in what is called 'full working precision'. When a result is presented, the FPU converts the special full working form into a value of the desired precision. This is called rounding.

In IEEE 754, four methods are defined for rounding a floating point number.

1. **Round to nearest :** System chooses nearer of the two possible outputs. If the correct answer is exactly halfway between the two, the system chooses the where the least significant bit of Fraction is zero.

   Ex: 0.987654321 → 0.9876543

2. **Round toward plus infinity:** The system chooses the larger of the two possible outputs

   Ex: 0.987654321 → 0.9876544

3. **Round toward minus infinity:** The system chooses the smaller of the two possible outputs

   Ex: 0.987654321 → 0.9876543

4. **Round toward zero or truncate:** The system chooses the output that is closer to zero, in all cases.

   Ex: 0.987654321 → 0.9876543