COMPUTER ORGANIZATION AND ARCHITECTURE
UNIT –III
TOPIC- DATA REPRESENTATION-FLOATING POINT REPRESENTATION

Floating-Point Representation

Using 32 bits to represent a number, positive or negative, the range of possible values is large but there are circumstances when bigger number representations are needed. The way to do this is to use floating point numbers. The reason for using floating point representation is that the range of possible values is much greater.

The floating-point representation of a number has two parts:
- The first part represents a signed, fixed-point number – the mantissa
- The second part designates the position of the binary point – the exponent
- The mantissa may be a fraction or an integer

Example: decimal number +6132.789 is
Mantissa: +6123789
Exponent: +04
Equivalent to +0.6132789 x 10^{+4}

A floating-point number is always interpreted to represent ***m* x *r***$\boldsymbol{m \times r^e}$
Example: binary number+1001.11 (with 8-bit fraction and 6-bit exponent)
Mantissa: 01001110
Exponent: 000100
Equivalent to $+(.1001110)_2 \text{x } 2^{+4}$

Normalisation of Floating point numbers

Normalisation, in floating-point number systems, is a process of standardizing these numbers into a consistent format. This procedure is crucial in computing for ensuring uniformity, accuracy, and efficiency in arithmetic operations and data storage.

Normalised form for floating point binary numbers is the equivalent of **standard form** for denary numbers. For example, the denary number 9257 can be represented as:

- 9257×10^0
- 925.7×10^1
- 92.57×10^2
- 9.257×10^3

Only the last example is represented in standard form. In standard form, the mantissa must be between 1 and 10 (>=1 and <10).

In the normalized form, there is only a single non-zero digit before the radix point.

Similarly, suppose you want to represent +12.510 as a floating point binary number. Here are some examples of the way that the number can be represented:

- 01100.1×2^0
- $0110.01 \text{ x } 2^1$
- 011.001×2^2
- 01.1001×2^3
- 0.11001×2^4

All of these examples represent the same value (+12.510), but only the final example is normalized.

For example, decimal 1234.567 is normalized as $1.234567 \text{ x } 10^3$

- by moving the decimal point so that only one digit appears before the decimal.
- The exponent expresses the number of positions the decimal point was moved left (positive exponent) or moved right (negative exponent).

To normalise a floating point number, you need to first determine whether the number is positive or negative by checking the most significant bit of the mantissa

Adjusting the mantissa

If the number is **positive**, you need to move through the mantissa from left to right until you find the first bit whose value is 1. The binary point will be positioned immediately in front of this bit.

A normalised positive number always starts as 0.1

For example, the floating point number 0.1101×2^3 **is normalised**, whilst the floating point number 0.01101×2^4 **is not**.

If the number is **negative**, you need to move through the mantissa from left to right until you find the first bit whose value is 0. The binary point will be positioned immediately in front of this bit.

A normalised negative number always starts as 1.0

For example, the floating point number 1.0011×2^3 **is normalised**, whilst the floating point number 100.11×2^1 **is not**.

Adjusting the exponent

If you move the binary point in the mantissa, you must adjust the exponent so that the value of the number remains the same.

- Every time you move the binary point **one place to the right** you must **reduce the value of the exponent by 1**.
- Every time you move the binary point **one place to the left** you must **increase the value of the exponent by 1**.

Ex 1:

- the positive floating point number 0.00011×2^5 is not normalised since it does not start with 0.1.
- To normalise this number, move the binary point 3 places to the right and reduce the value of the exponent by 3 to give 0.11×2^2.

Ex 2:

- the positive floating point number 010.01×2^{-3} is not normalised since it does not start with 0.1.
- To normalise this number, move the binary point 2 places to the left and increase the exponent by 2 to give 0.1001×2^{-5}.