Assignment 1: Number Representations

INTRODUCTION:

Converting decimal integer to binary

To convert integer to binary, start with the integer in question and divide it by 2 keeping notice of the quotient and the remainder. Continue dividing the quotient by 2 until you get a quotient of zero. Then just write out the remainders in the reverse order.

Here is an example of such conversion using the integer 12.
First, let's divide the number by two specifying quotient and remainder:

$$\begin{aligned} 12 : 2 &= 6 + 0 \\ 6 : 2 &= 3 + 0 \\ 3 : 2 &= 1 + 1 \\ 1 : 2 &= 0 + 1 \end{aligned}$$

Now, we simply need to write out the remainder in the reverse order –**1100**. So, **12** in decimal system is represented as **1100** in binary.

Converting decimal fraction to binary

To convert fraction to binary, start with the fraction in question and multiply it by **2** keeping notice of the resulting integer and fractional part. Continue multiplying by 2 until you get a resulting fractional part equal to zero. Then just write out the integer parts from the results of each multiplication.

Here is an example of such conversion using the fraction **0.375**.

$$\begin{aligned} 0.375 \cdot 2 &= 0 + 0.75 \\ 0.75 \cdot 2 &= 1 + 0.5 \\ 0.5 \cdot 2 &= 1 + 0 \end{aligned}$$

Now, let's just write out the resulting integer part at each step – **0.011**. So, **0.375** in decimal system is represented as **0.011** in binary.

INT TO HEX

The concept is exactly the same as INT to BINARY conversion.
You can either convert the BINARY to HEX, or you can directly start with base 16 instead of 2.
Example: 2545 => 000009F1

FP TO HEX

The concept is similar to the above one.

There are several ways to represent floating point numbers but IEEE 754 is the most efficient in most cases. IEEE 754 has 3 basic components:

1. **The Sign of Mantissa:**
   This is as simple as the name.
   0 represents a positive number while 1 represents a negative number.
2. **The Biased exponent**:
   The exponent field needs to represent both positive and negative exponents.
   So, a *bias*(127 for Single Precision FP) is added to the actual exponent in order to get the stored exponent.
   So, if the exponent is 4, it stores 4+127=131.
3. **The Normalised Mantissa:**
   The mantissa is part of a number in scientific notation or a floating-point number, consisting of its significant digits. Here we have only 2 digits, i.e. 0 and 1. So a normalised mantissa is one with **only one 1 to the left of the decimal**.
   Valid examples: 1.01001, 1.00001 etc.
   Invalid example: 11.001, 0.101 etc.

Example: Let's work out the **representation of -17 in FP.**

1. **Sign bit** = 1 as it is negative.
2. **Exponent** is decided by the nearest smaller or equal to 2^n number. For 17, 16 is the nearest 2^n. Hence the exponent of 2 will be 4, since $2^4 = 16$.
   So, the biased exponent that needs to be stored is 4+127=131 i.e. 10000011.
3. **Mantissa:** 17 in binary = 10001.
   Move the binary point so that there is only one bit from the left. Adjust the exponent of 2 so that the value does not change. This is normalizing the number = 1.0001×2^4. Now, consider the *fractional part(after decimal place i.e. 0001)* and represent it as 23 bits by adding zeros. => 00010000000000000000000

Thus the floating point representation of -17 is 1 10000011 00010000000000000000000.
Now convert it to 8-bit Hexadecimal.

Another Example: Number = 85.125

1. **sign = 0**
2. Binary of 85 = 1010101, Binary of 0.125 = 001
   => 85.125 = 1010101.001 = 1.010101001 x 2^6
   => Exponent = 6
   => **Biased exponent = 6+127 = 133 = 10000101**
3. **Mantissa = 010101001** (after decimal place)
   We will add 0's to complete the 23 bits.

So, the IEEE 754 Single Precision is:
= **0 10000101 010101001**00000000000000
This can be written in hexadecimal form **42AA4000**