Assignment 11: Floating Point Addition

KAPIL VERMA
ENTRY NO-2018CS10348

About My Implementation:

In this assignment, we have to build addition calculator for IEEE754 floating point addition calculator. We are taking input through another file(.txt). The format for that is like:

Input1 input2

As we know that any IEEE 754 formatted floating point contains three parts:

- Sign [first bit of number.]
- Exponent [Next 8 bits after sign]
- Mantissa [Next 23 bits]

So total 32 bit lies in single precision floating point number.

So, first of all, we are dividing the given input in their respective part i.e. signs, exponent and mantissa. Now we make exponent of both the numbers equals. After that we check that signs of both number if they are opposite than we do subtraction otherwise go for addition.

After addition or subtraction, we go for normalization. There we shift the mantissa right or left according to situations so that overall, we can get normalized number. During shifting we add or subtract the exponent by which we have shift the number.

- For shifting right, we add in exponent and for shifting left, we subtract in exponent.

During addition in exponent, we check that does the unsigned value of exponent exceed that 254, it does than overflow. If it goes less than 0 that underflow.

Cycles:

Number of cycles goes like:

- One cycle for shifting to make sure that exponent of both are equal.
- One cycle for addition or subtraction.
- One cycle for Normalization.
- One cycle for Rounding.

Rounding:

During normalization, we check that what is the value is last digit. If value is 1 then we will add that 1 to the mantissa and again go for checking normalization, if it’s normalized then we are done.

If the last digit is 0 then we don’t need to add anything to mantissa.

Functions:

- Int binarytodecimal(int exponent):

  This function takes a binary number in int form then return its decimal form.

- Char* shift(char input [], int shift, int right, int len):

  This function take binary no store in char array, shift denoting how much we have to shift, int right denote that do we have to shift right or left and len denote the len of binary no.

- Char readfile(FILE *file):

  This takes a FILE pointer that is file. This function read the input file and Call the checking function for execution.

- Int checking (char input1[], char input2[]):

  This function takes two inputs Nd decode the number. Here we r dividing number in their respective parts like sign, mantissa and exponent.

According to sign we will call addition and subtraction functions. After getting result of addition or subtraction, we go for Normalization.

- Char addition (char input1[], char input2[]):

  This takes two input and add them on the basis of binary addition.

- Char subtraction (char input1[], char input2[]):

  This takes two input and subtract input2 from input1 them on the basis of binary subtraction.

- Char Normalization (char* input, int exp, int k):

  This take an input which is going to be normalized. Exp denote the final exponent for this no and k denote that are we calling this function after addition or subtraction. If addition just before it then k=1 otherwise 0.

Cases:

We handled all the cases using suitable approaches. The cases are:

- Underflow:

  If final exponent (not bias) of result is greater than 254.

- Overflow

  If final exponent (not bias) of result is less than 0.

- Infinity

  If final exponent of result is 255 and mantissa is equal to 0 then it infinity.

- Nan:

  If final exponent of result is 255 and mantissa is not equal to 0.