COMP.3050 Computer Architecture
Homework #2 September 14, 2023

- This assignment is due on **Tuesday, September 26.**

- Your submission must be made electronically using the submit command as described below.

- **All** of your submissions must include a minimum of **four** separate files:

1. **File 1:** A short **write-up** that **first** specifies what you think your **degree of success** with a project is (**from 0% to 100%**), followed by a brief discussion of your approach to the project along with a **detailed description** of any problems that you were **not** able to resolve for this project. **Failure to specifically provide this information will result in a 0 grade** on your assignment. If you do **not disclose** problems in your write-up and problems are detected when your program is tested, you will receive a grade of 0. **Make sure that you include your email address in your write-up so that the corrector can email you your grade.**
2. File(s) 2(a, b, c, …): Your complete source code, in one or more .c and/or .h files
3. **File 3**: A **make file** to build your assignment. This file must be named **Makefile**.
4. **File 4:** A file that includes your **resulting output** run(s) from your project. This is a simple text file that shows your output, but make sure that you annotate it so that it is self-descriptive and that all detailed output is well identified.

- The files described above should be the only files placed in one of your subdirectories, and this subdirectory should be the target of your submit command.
- The files described above should be the only files placed in one of your subdirectories, and this subdirectory should be the target of your submit command (see the on-line file **Assignment_Submit_Details.pdf** for your section (201 or 203) for specific directions).

- This problem requires you to add two **positive only** IEEE floating point numbers together by emulating the floating point hardware in software:

1. You are only allowed to use the **integer operations in C** (i.e. shift, integer add, bitwise logical operators, etc.) to do the addition.
2. You will need to scan in two floating point numbers into unions which allows access to the mantissa, exponent and sign components as shown in class.
3. Using the bit field components of the two numbers you must compute the result of their addition and rebuild a new floating point number which you can printf as output. As shown below, **you must include printed bit output** which shows the steps you're using to complete the addition.

- Your program interface **must look like**:

```
**********************************************************************
```

This program will emulate the addition of two IEEE 754 floating point numbers

```
Please enter two positive floating point values each with:
    - no more than 6 significant digits
    - a value between   +   10**37 and 10**-37

Enter Float 1:  34.5
Enter Float 2:   1.250

Original pattern of Float 1:  0 1000 0100 000 1010 0000 0000 0000 0000
Original pattern of Float 2:  0 0111 1111 010 0000 0000 0000 0000 0000
Bit pattern of result      :  0 1000 0100 000 1111 0000 0000 0000 0000

EMULATED FLOATING RESULT FROM PRINTF ==>>> 35.75
HARDWARE FLOATING RESULT FROM PRINTF ==>>> 35.75
**********************************************************************

Your program output should look like the above example.  The following
output is not required but is shown here to provide a step-by-step list
of what you have to do to produce the above output example.  To get the
result of the addition constructed into the union for the example shown
above you must:

  -COPY THE MASNTISSA PARTS INTO THEIR OWN HELPER INT VARIABLES AND
  -EXPOSE HIDDEN BITS INTO MANTISSA HELPER VARIABLES TO GET 24 BITS:
      Slam hidden bit into Float 1:    1000 1010 0000 0000 0000 0000
      Slam hidden bit into Float 2:    1010 0000 0000 0000 0000 0000

  -SHIFT MANTISSA OF SMALER VALUE FOR COMMON EXPONENT:
      Post shift pattern of mant. 1:     1000 1010 0000 0000 0000 0000
      Post shift pattern of mant. 2:     0000 0101 0000 0000 0000 0000

  -ADD AND ADJUST FINAL MANTISSA OF RESULT:
      Bit sums before adjustment:        1000 1111 0000 0000 0000 0000
  -SINCE HIDDEN BIT IS PERFECT FOR RESULT JUST REMOVE IT
      Final 23 bit pattern for result:    000 1111 0000 0000 0000 0000
  -NOW PUT RESULT INTO THE MANTISSA BIT PART OF THE FINAL ANSWER
  -WITH COMMON EXPONENT AND CORRECT SIGN PLACED IN THEIR BIT FIELDS
   AND PRINT THE UNIT AS A FLOAT
```

- You must generate output for at least the numbers that are found in the file:
  ~bill/cs305/as2testdata (the file is also on the class webpage**)**