Instructions to prepare and submit your homework

1. Explain the general plan of the program in Q. 1 using a readme file.
2. Be generous about using comments to improve readability.
3. To submit, zip (or tar) all files into a single file, and drop it to ICON drop box.

Question 1. (40 points) Create an exponent function: float \( \text{exp}(\text{float } x) \) that accepts an input \( x \) from the user, and returns \( e^x \), (using the MIPS floating point co-processor).
Recall that \( e = 2.71828183... \) Use Taylor Series expansion to compute the exponential function:

\[
e^x \approx 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + ... + \frac{x^{10}}{10!}
\]

(It is an infinite series, but you can stop after computing up to the 10\textsuperscript{th} term)

To facilitate this, you may create two functions, \( \text{power} \) and \( \text{factorial} \), that may have the signatures: float \( \text{power}(\text{float } x, \text{int } n) \) and int \( \text{factorial}(\text{int } n) \). Here, \( \text{power}(x, n) \) would return \( x^n \) for \( n \geq 0 \) and \( \text{factorial}(n) \) will return \( n! \). For computing the factorial, you may write either a recursive program or a simple iterative program.

A helpful SPIM instruction is \( \text{cvt.s.w } Fd Fs \) that converts an integer in the source register \( Fs \) to a single precision floating-point number in the destination register \( Fd \). Here is an example of its usage:

\[
\begin{align*}
\text{mtc1 } &\$v0, \$f1 \quad \# \text{move to register } \$f1 \text{ (in coprocessor C1) from register } \$v0 \\
\text{cvt.s.w } &\$f1, \$f1 \quad \# \text{convert the integer in } \$f1 \text{ to single precision floating point format} \\
\text{div.s } &\$f0, \$f0, \$f1 \quad \# \text{divide } \$f0 \text{ by } \$f1 \text{ and store the result in } \$f0
\end{align*}
\]

Here is another example of a program that computes the polynomial \( ax^2 + bx + c \).
A summary of some useful floating-point instructions is available in Appendix B of your textbook.
Question 2. (10 points) Let $Y, Z$, be two 32-bit registers, $X$ be a single bit. Draw a circuit so that by applying a single pulse in the clock line, the following operation can be performed:

\[
\text{if } X = 0 \text{ then } Y := Y + Z \text{ else } Y := Y - Z
\]

You can use an **Add / Subtract** unit with a function control input $f$, such that when $f=0$, the unit acts as an adder, and when $f=1$, it acts as a subtractor. Briefly explain why your circuit will work.