22C:060: Computer Organization
Homework 3
Total points = 50
Due Tuesday October 22, 2013, 5:00 AM (not PM)

(Ideally you should finish the work on the previous night and submit it by midnight.

Late submissions will not be accepted)

1. Do not consult others. You must solve the problems on your own.
2. Be generous about using comments to improve readability. This includes a comment at the beginning specifying the purpose of the program.
3. To submit the program, zip (or tar) them into a single file that has your last name as the prefix. Use ICON drop box to submit your assignment.

The Question
Create an exponent function: float exp (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:

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

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

Part 1 (15+15 = 30 points) To facilitate this, create two functions: (1) float power (float x, int n) and (2) factorial, that will use the two functions: float power (float x, int n) and int factorial (int n). Here, power (x, n) would return x^n for n ≥ 0 and factorial n will return n!. For computing the factorial, you can write either a recursive program or a simple iterative program.

Part 2 (20 points) Use the two functions to compute e^x from a given value of x.
A helpful SPIM instruction is `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:

```
mtc1 $v0, $f1    # move to register $f1 (in coprocessor C1) from register $v0
cvt.s.w $f1, $f1  # convert the int in $f1 to single precision floating point format
div.s $f0, $f0, $f1 # divide $f0 by $f1 and store the result in $f0
```

Here is another example of a program that computes the polynomial $ax^2 + bx + c$

```
## float1.s -- compute ax^2 + bx + c for user-input x
.text
.globl main
##
# Register Use Chart
# $f0  --  x
# $f2  --  sum of terms
main:    # read input
   la  $a0,prompt       # prompt user for x
   li  $v0,4            # print string
   syscall
   li  $v0,6            # read single
   syscall             # $f0 <-- x
   # evaluate the quadratic
   l.s  $f2,a           # sum = a
   mul.s $f2,$f2,$f0     # sum = ax
   l.s  $f4,b           # get b
   add.s $f2,$f2,$f4     # sum = ax + b
   mul.s $f2,$f2,$f0     # sum = (ax+b)x = ax^2 +bx
   l.s  $f4,c           # get c
   add.s $f2,$f2,$f4     # sum = ax^2 + bx + c
   # print the result
   mov.s $f12,$f2       # $f12 = argument
   li  $v0,2            # print single
   syscall
   la  $a0,newl         # new line
   li  $v0,4            # print string
   syscall
   li  $v0,10           # code 10 == exit
   syscall             # end the program
```
## Data Segment

```al
## .data
    a: .float 1.0
    b: .float 1.0
    c: .float 1.0
    prompt: .asciiz "Enter x: \\
    blank: .asciiz " \\
    newl: .asciiz "\n"
```

A summary of some useful floating-point instructions is available in Appendix B of your textbook.