22C:060: Computer Organization

Spring 2011

Assignment 3

Total points = 50

Assigned March 22, due March 29, 2011, 11:59:59 PM

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 **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:

$$e^x \sim= 1 + x + (x^2)/2! + (x^3)/3! + \ldots + (x^{10})/10!$$

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

To facilitate this, you may create two functions, *power* and *factorial*, that may have the signatures: float power (float x, int n) and int factorial (int n). Here, *power (x* ,n) would return x^n for $n \geq 0$ and *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 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 integer 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                # Return to OS.

## Data Segment
##   .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.

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:

if X= 0 **then** Y:= Y+Z **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.