EE 2310 Test Review #3 – Assembly Language and Computer Architecture

Name _____

1. **Construct a brief loop program to count all the letters in the phrase shown. Note that the first letter of the phrase was made lower case, so that all you have to do is count lower-case letters. The program should output the number of letters only; no leader is needed.**

   .

```
        .text
main:
        .data
str:    .asciiz "hello, world!\"
```

2. **Write a program that calculates the function x^y. Locations "x" and "y" hold operands x, y. The result of the calculation is stored in location z when completed. You need only write the text section, as the data declaration is already done. <u>You must use a loop to do the calculation</u>. Use the data declaration shown at the right for w, y, and z.**

   What is the decimal answer? **<u>100,000,000</u>.**

```
        .data
x:      .word 100
y:      .word 4
z:      .space 4
```

3. **Write a brief loop program to look for 0's in the data word "data1" given in the data statement to the right. Note that since the word has 8 hex digits, the loop needs to be traversed eight times. You will also need to create a "mask" or set of "mask bits" so that you can isolate each hex number to test whether or not it is 0.**

   Print out the number of 0's found when the loop ends and then exit the program with a syscall 10 as usual.

```
        .data
data1:  .word 0x9f07ec03
```

```
        .text
main:
```

4. **Create a timer program that will output a 30-second sequence of numbers (i.e., 1 2 3 4… at one number per second). You will have to experiment a bit, but start with a loop that counts down to 0 from 250,000, then change the number to get as close to one second per cycle as you can.**

5. **In the early 13th century, the mathematician Fibonacci developed a formula for a number series that predicted the reproduction rate of rabbits. This series turned out to be an important mathematical development.**

   The general formula for a Fibonacci number F(n) is: F(n) = F(n-1) + F(n-2), n an integer > 0, where F(0) is defined as 0, and F(1) is defined as 1.

   Write a program that inputs a number n from the user and then calculates F(n) using a recursive routine, where F(n) is the Fibonacci number for the integer n. Due to fixed point calculation, restrict the input n to 40 or less. Use the data declarations shown to the right.

   Hints:
   (1) Note that the first F(n) that would ever have to be calculated is F(2), as F(0) and F(1) are defined.
   (2) You should write this program using a recursive loop as follows: Write a jal routine that subtracts 1 from the integer n (for which it is desired to calculate F[n]) each time it goes through the loop and tests if the resulting number is 2. If not, jal to the routine again and subtract again. Each time through the jal loop, store $ra on the stack. When n is counted down to 2, go to the second recursive loop, the jr loop. Calculate F(2), pop $ra off the stack and do a jr $ra. This should take you through the jr loop again and you will calculate F(3). When the second loop has done as many jr's as there were jal's in the first loop, the second loop will have calculated F(n).

   When the program is working properly, calculate F(5) and F(10). [F(5)=5; F(10)=55.]

   Note: Like the programming problems in Homework 7, this problem is challenging, and requires some thought and a well-planned approach.

```
        .text
main:




        .data
inpt:   .asciiz "Input 2-digit number,
40 or less, for Fibonacci calculation: "
ans1:   .asciiz "F ("
ans2:   .asciiz ") is "
endl:   .asciiz ".\n"
n:      .word 0
```