EE 2310 Homework #6 – Writing Loop Programs

Name:_______________________________________ EE ______ CE ______

Write the following programs as directed. Once again, you need to write and run the programs under your SPIM emulator. MAKE SURE THAT THEY RUN CORRECTLY! On the homework due date, email the programs as NotePad attachments, directly to your TA. The questions can be listed and answered in the email itself.

1. **In the box at right, compose a loop program to store the value 0x89ab672f in data locations w, x, y, and z. Note that the data words w-z are declared with value 0, as a place holders for each number. Remember: the program MUST be a loop! That means that you must write a loop that will execute four times to store the data in locations w-z. End the program as usual with a syscall 10. Hint: This would be a good program to use the branch instruction beqz.**

   After completing the program answer these questions:

   1.1. What are the memory addresses of w, x, y, z?

   1.2. How do you decide when you are done (i.e., how do you know when you have gone through the loop four times)?

```
        .text
main:




        .data
w:      .word 0
x:      .word 0
y:      .word 0
z:      .word 0
```

2. In the space at the right, compose a program to examine the string “Hello, world!\n” looking for a d. Each time a letter is not a d, add one to a counter register. When a d is finally found, output the number of characters examined prior to the d.
   Note that this program will take a loop, but this time you do not need a counter to know when to quit (since you stop after a d). Remember, punctuation (even spaces!) <u>are characters</u>.

   What is the character count before d is encountered?

```
        .text
main:

        .data
str:    .asciiz "Hello, world!\n"
```

3. In the area to the right, construct a program that will print out the data words in memory in the reverse order that they are declared in the data statement. To do this, use the stack to reverse the data sequence. You will need two small loops: one to store the four data words on the stack, and a second to print the data (which is conveniently in reverse order on the stack!) to the console. Yes, you WILL need a counter in both loops! When addressing the stack, use the “points to the last filled location” convention for the stack pointer ($29).

   Remember to output a CR/LF between the numbers to keep them on separate lines.

```
        .data
num1:   .word 0x47
num2:   .word 0xf8
num3:   .word 0x3c
num4:   .word 0xad
```

4. **In the space to the right, construct a program that outputs only the capital letters, spaces, exclamation points, and carriage return/line feeds in the data words "w," "x," and "y" declared in the data statements. Note that this means you will need a loop that analyzes bytes in the three words one at a time, determines if the byte is the ASCII code for a capital letter (ASCII codes 0x41-0x5a), space (0x20), exclamation point (0x21) or CR/LF (0x0a). If the byte is one of these characters, the program will print it out using syscall 11. If not, the byte is discarded and the next byte is loaded and analyzed.**

   Note that your loop will need a counter to determine when you have analyzed all the bytes. When the analysis is complete, end the program as usual.

   In your email to the TA with the attached programs, write out the contents of the print-out.

```
        .data
w:      .word 0x44305455
x:      .word 0x43314520
y:      .word 0x0a213753
```