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. (CLO 5—Assy Lang.) Compose a loop program to put the value 0x73a8ef64 in $t0. Then, store each of the bytes in the word (0x73, 0xa8, 0xef, 0x64) in data locations w, x, y, and z, consecutively, starting with the 0x64 byte. Note that the data words w-z are declared with value 0, as a place holders for each number. Also, note that you are storing each BYTE as a separate WORD. The program must be a loop. That means that you must write a loop that will execute four times to mask off each byte, move it to another register, and store it in one of the locations, w, x, y, or z. End the program as usual with a syscall 10.

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)?
2. (CLO 5—Assy Lang.) Compose a program to examine the string “Hello, world!\n”, calculating the total numeric value of all the characters in the string (including punctuation marks). The program should load each letter, add that numbers numeric value to the running total, and finally produce a total sum. The program will take a loop, but in this case, you do not need a counter, since the phrase is null terminated. Simply look for a null (0) and then terminate the program by printing out the total. Remember, punctuation (even spaces!) have a numeric value as well.

What is the total character numeric sum?

3. (CLO 5—Assy Lang.) Construct a program that will print out the data words in memory AS ASCII CHARACTERS 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). Note 1: In the first loop, use a counter to count up to 4; then count DOWN from 4 in the second loop. Note 2: If you pop each word off the stack into $a0, then execute a syscall 11, you will print out the least significant byte of the word as an ASCII character.

End the program with a syscall 10, as usual.

What is printed out?
4. (CLO 5—Assy Lang.) Construct a program that compares the five numbers in the list below and outputs only the smallest one.

Note that your loop will need a counter to determine when you have analyzed the five words. When the analysis is complete, end the program as usual. Note: This must be a loop program.

Output the leader shown below before printing the number.

```
.text
main:

.data
v: .word 0x6966abce
w: .word 0x89c7d12e
x: .word 0x7ff9348c
y: .word 0x445e279a
z: .word 0x123ac45d
ldr: .asciiz "The smallest number = "
```