CS 231 – Lab #4 – Function practice

In this lab, you are given two incomplete MIPS assembly programs that you need to finish. In each case, there will be a function for you to implement, and possibly some other code as well.

Create a new folder called lab4 on your USB drive or account, and do all of today's work in there. From the class Web site, you should see a folder called lab4. Download divisor.s and sort.s. These are the programs to modify.

Part #1 – divisor.s

Invoke MARS as in the previous lab. Open the source file divisor.s. The purpose of this program is to ask the user for a positive integer, and then calculate the sum of its proper divisors. A proper divisor of n is a number less than n that divides evenly into n. For example, the proper divisors of 10 are 1, 2 and 5. After finding the sum of proper divisors, the program will classify the user's input value as being one of the following: prime, deficient, perfect or abundant.

- Prime: the only proper divisor of n is 1
- Deficient: the sum of proper divisors is less than n
- Perfect: the sum of proper divisors equals n
- Abundant: the sum of proper divisors exceeds n

Technically, all prime numbers are deficient as well, so when we report a number as prime, we will also mention it is deficient. In the data segment, all of the output strings you will need have already been created for you.

The program is missing two pieces that you need to fill in. Use the comments in the program to guide your solution.

1. At the bottom of the program there should be a function called proper that returns the sum of the proper divisors in the number stored in register $a0.

2. The main program calls proper, and then prints the value being returned from that function. Next, the program determines if the user's input value is prime. You need to handle the other 3 cases: deficient, perfect, abundant.

When you are done with the implementation, test your code with some suitable input. For example, make sure your program recognizes that 2520 is abundant, 2521 is prime, and 8128 is perfect. Can you think of a number that is deficient but not prime? _____√

Part #2 – sort.s

This simple program is designed to sort 3 numbers in ascending order. This program is not intended to show you an efficient sorting algorithm (it's actually a form of Stooge sort!). Rather, the purpose of this program is to have you practice with an assembly program that exhibits nested function calls.

In this program, the main program calls a function called sort3. The job of sort3 is to sort 3 numbers. It delegates its work to another function called sort2. Believe it or not, sort3 will call sort2 three times. The comments in the program should explain the details.

Your job is to complete the implementation of sort3. Take note of how the main program calls sort3, and how sort2 communicates parameters with sort3 to guide your solution. √

Part #3 – Encoding practice

Finally, let's have some fun with the binary representation of MIPS instructions. Note that if you select the Execute tab, you can see your program's details. The "Code" column shows binary (machine) language instructions, while the "Basic" column shows "true" MIPS instructions. Let's investigate the correspondence between the two.

1. From either program you wrote today, select one instruction that uses at least two register operands. Work out the binary representation by hand.

2. Select some other instruction in either program. Write down its binary representation. Give this binary instruction to someone else in the class, and ask them to decode it – in other words, figure out what the human readable MIPS assembly instruction is. Write down the binary and assembly versions of the instruction here for reference:

   _____

   Now, in the space below, write down the binary instruction that someone else gave you:

   _____

   Decode (disassemble) this binary code to determine the corresponding MIPS instruction. √