EC413 Computer Organization
Lab 2b – MIPS Assembly Language Programming
DEMO REQUIRED

The purpose of this lab is for you to gain experience programming and debugging Assembly Language. The lab consists of a series of tasks exercising fundamental capabilities: I/O, loops, transfers, conditionals, nested loops.

Notes:

- Comment your code in the style of the tutorial .asm file.
- Be sure your program works for any array size. In particular, during the demo you should be able to change the inputs and your program should still work correctly.
- There is no need for additional write-ups beyond the code (with comments).

PRELAB

1. Enter the program outline (lab2b.asm). Run the program to confirm that everything is working correctly, that is, that the program loads and executes. You should see the message "Hello World!" and also a bunch numbers beginning with "17" get printed.

2. Find the code that prints the string. Note that it uses a sequence of three instructions used by this environment for I/O. These (i) load the data or a pointer to the data into a register(s), (ii) load the call type into $v0 (or $f0), and finally syscall, which transfers control to the run-time support system and executes the I/O operation. See pages 8-9 of *SPIM S20: A MIPS R2000 Simulator* for details.
Edit the program so that the message displays your name.

3. Find the code that prints the integer value of **AnInt**. Note that rather than loading a pointer to a string into $a0, the value is loaded.
Write code to print two numbers separated by a space (see the data segment). One way to do this is with three system calls: one to print the first integer (**AnInt**), one to print the string labeled **space**, and one to print the second integer (AnotherInt). Remember that printing integers and printing strings use two different types of syscalls.

4. The current code for Task 4 interprets the array **Input1** as a sequence of byte sized integers and prints them out. Note that it loads the user-determined array size before starting (**InLenB**).
(a) Modify the code so that it prints spaces between the integers.

LAB

4(b) Copy the code into the next section labels Task 4b. Modify the code to print the array backwards.

5. The current code for Task 5 copies the contents of **Input2** to **Copy**, byte at a time. Note that it assumes that the length of **Input2** is known ahead of time and is stored in **InlenB.**
Modify the code so that it copies the data word at a time (rather than byte at a time).

6. (Accumulation and simple arithmetic) Write code to compute and print the integer average of the contents of array **Input2** (in bytes).

7. (Conditionals) Write code to display the first 25 integers (0 to 24) that are divisible by 7. Again, print a space between the outputs.

8. (Nested loops or index arithmetic) Write code to transpose the 10x10 array **Input3** into the array **Transpose**. Assume bytes. Note that Input3 is initialized dynamically.