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

PREPRELAB

0. Read the entire lab (this document) and the program outline (lab2b.asm). Review lecture notes.

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 SW tasks and methods, including I/O, loops, transfers, conditionals, nested loops.

Notes:

- For this and the remaining labs you are encouraged to work with a partner
- For this and the remaining labs early completion is encourage (with bonus points)
- Comment your code in the style of the tutorial .asm file (with possible bonus points)
- 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)

General Advice:

- I have always found it good practice when learning a new Assembly Language to keep (or make) a reference sheet of the most common instructions. The textbook Appendix A10 has all of the instructions grouped by type. In Appendix A10, some instructions are labeled as "pseudoinstructions." This means they are usable in writing MIPS assembly language code (L4), but do not have direct tranlations to machine code (L3). For this part of the course you can use them freely.
- Proceed in small steps. Do not write more than a few instructions at a time without checking that they are doing what you expect.
- As always, the labs work the best when you fully understand the basic material before you start. This means reviewing the loop codes in the first part of the AL3 lecture notes.

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 (OLD_spim_Documentation)* 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**).

(4a) Modify the code so that it prints spaces between the integers.

LAB

(4b) 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.

Extra Credit (10 points)

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.

Grading Rubric

10pts	Prelab
90pts	Working code with demo and explanations as requested
Extra Credit	
10pts	good comments/code structure
10pts	lab part 8
8pts	Demo on T,W
5pts	Demo on Th