CS 240 – Computer Organization

Lab 3 – Logic Handling and File I/O – 110 points

The primary goal of this lab is to learn how to use low-level MIPS assembly instructions to implement *logical* and *conditional statements*. The lab also introduces you to File I/O operations. This lab comes in three parts. There are 110 points in total.

In this lab, you will learn how to use MIPS assembly instructions to:

1. Implement FOR and WHILE loop constructs
2. Read/write from/to memory
3. Read from a file

Download the lab files and implement three subroutines "64-bit Unsigned Addition", "Fibonacci", and "file_read". Please look at comments in the provided source code for more information.

Important note: ***Do not use the registers $s0 to $s7 in any of your codes.***

Your tasks:

1. Complete the function **Unsigned_Add_64bit** to perform sum of two 64-bit numbers. (30 points)

- You are given two 64-bit numbers A and B located across 4 registers
- $t0 and $t1 registers are preloaded with lower and upper 32-bits of number A
- $t2 and $t3 registers are preloaded with lower and upper 32-bits of number B
- *You need to store the result of the unsigned addition in $t4 and $t5 for lower and upper 32-bits.*

2. Complete the function **Fibonacci** in the template to compute the Fibonacci series (40 points)

The Fibonacci Sequence is the series of numbers:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

The next number is found by adding up the two numbers before it.

The 2 is found by adding the two numbers before it (1+1)

The 3 is found by adding the two numbers before it (1+2),

The 5 is (2+3),

and so on!

- You should compute and return the nth digit of the Fibonacci sequence.
- The digit you need to compute will be in $a0.
- *Return your digit in $a1.*

3. Complete the function **ReadFile** to read data from a file and print them (40 points)

- Use syscalls to open and read content from the file to an input buffer
    - For example, refer to the class slides on File I/O and syscall
    - For full reference of syscalls, go to the link below: http://courses.missouristate.edu/KenVollmar/Mars/Help/SyscallHelp.html
- Print out the valid characters from the input buffer. Valid characters include **space, period(.), newline, a-z, A-Z and 1-9.** All other characters should not be printed.
    - *Use ascii values to determine if a character is valid or invalid.*
    - Refer to syscalls data sheet or class slides for info regarding printing characters.

Strictly adhere to the following information:

- Opening the file:
    - Address of the null-terminated string containing filename (the $a0 argument): **file_name**
    - Set the flag for read_only. Mode should be hardcoded to 0.

- Reading from the file:
    - Address of input buffer (the $a1 argument) : read_buffer
    - Max number of characters to read (the $a2 argument) : 300
    - Preserve the return value $v0 to use a loop exit condition.

- DO NOT FORGET TO CLOSE YOUR FILE!!!

Submissions:

- Complete the provided lab3.s template and submit it to Gradescope.
- **Submit only lab3.s by itself (not in a folder/zip and with no other files).**

Important Things to Consider

1) Don't forget to update the student name and id variables. These are used for grading purposes. FAILURE TO ADD NAME AND ID RESULTS IN NO SUBMISSION

```
student_name: .asciiz "Your Name"
student_id: .asciiz "Your Student ID"
```

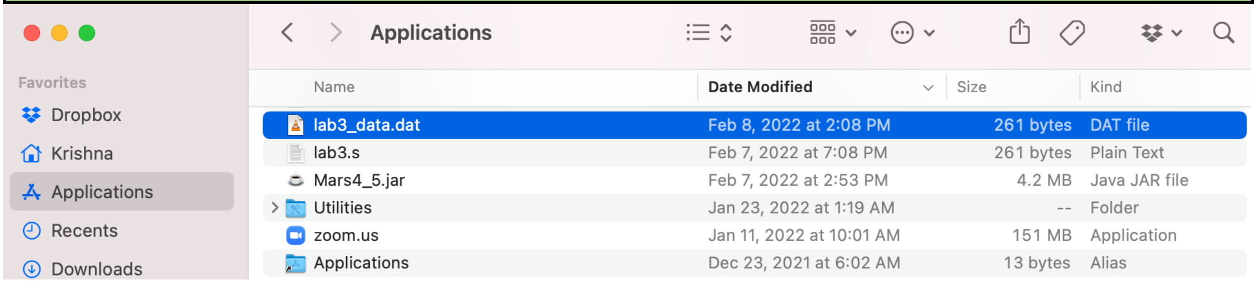
2) Since we use a computer program to test your code, you are **ONLY** allowed to modify the highlighted area in the source code.
For example:

```
count_ones:
move $t0, $a0
############### Part 1: your code begins here ##################
Your stuff goes here :D
############### Part 1: your code ends here  ###################
move $v0, $t0
jr $ra
```

3) Do not use the $s registers in your code (subroutine) as they are used by the main program.

4) The programs do not require use of additional subroutines. However, if you end up creating a new subroutine in your code, make sure you preserve the original value in $ra register.

5) In order for the file reading to work, your lab3 file, .dat file, and mars have to be in the same folder. Please do not modify the path inside the lab 3 file as it will mess with grading. See image for example:



Name	Date Modified	Size	Kind
lab3_data.dat	Feb 8, 2022 at 2:08 PM	261 bytes	DAT file
lab3.s	Feb 7, 2022 at 7:08 PM	261 bytes	Plain Text
Mars4_5.jar	Feb 7, 2022 at 2:53 PM	4.2 MB	Java JAR file
Utilities	Jan 23, 2022 at 1:19 AM	--	Folder
zoom.us	Jan 11, 2022 at 10:01 AM	151 MB	Application
Applications	Dec 23, 2021 at 6:02 AM	13 bytes	Alias

Gradescope will immediately show you your auto grading score, and you can resubmit as much as you need (grades will be synced to Canvas later). Note that we will use additional test cases other than the ones provided so make sure to test your code yourself. 5 points of each problem will be awarded for making a serious attempt at each task AND for adding comments.

CHEATING:

Submit your own work. While you may collaborate with other students, you are required to submit your own code. Any students found submitting identical/very similar code will be evaluated for academic dishonesty. DO NOT share/send your code to other students, this is also considered cheating. We use a similarity checker to evaluate all submissions.

New Anti-Cheat Measure:

The instructor and the TA reserve the right to call upon any student to explain their code after submission deadline.

Failure to comply AND/OR if the student is unable to defend their work/code, the work will be considered as plagiarized; The student will earn no credit for the lab and may be reported for academic dishonesty.