CMSC 133

Introduction to Computer Organization, Architecture and Assembly Language

Laboratory Report

Names of all Authors and Student IDs	Kienth Matthew B. Tan
Lab No.	1
Task/Step No.	2
Lab Title	MIPS
Email Addresses of all Authors	kbtan@up.edu.ph
Date of Document Issue	03/30/2022
Document Version Number	1.0
Address of Organisation Preparing this report:	Department of Computer Science University of the Philippines Cebu Gorordo Avenue, Lahug, Cebu City Tel: (032) 232 8185
Course Name	CMSC 133 Introduction to Computer Organization, Architecture and Assembly Language
Course Units	3

Summary.

The Laboratory activity is all about putting into action what we learned in our previous lessons and video lectures. It aims to help us learn the basics of MIPS assembly language from basic programs while task by task progressing into more complex problems.

The **second task** which will be featured in this laboratory report is on how to find the length of a string. The task is actually quite simple as we simply had to loop through the string to get the length. Through this task, I learned through practice on how to use numerous other instructions such as lb, li, and beqz. The task also taught me how to access the next character in a string.

In approaching my method to solving this task, I first created a C program that would solve the laboratory problem. Things would then be much simpler after that as I only had to translate my C code into MIPS assembly code. I then tried to apply what I learned on how to indicate when the loop ends. After that, the steps are quite simple as I just had to loop the code up until it detects a terminating NULL, and for each loop we add the length by 1.

TABLE OF CONTENTS

1 The Lab Problem

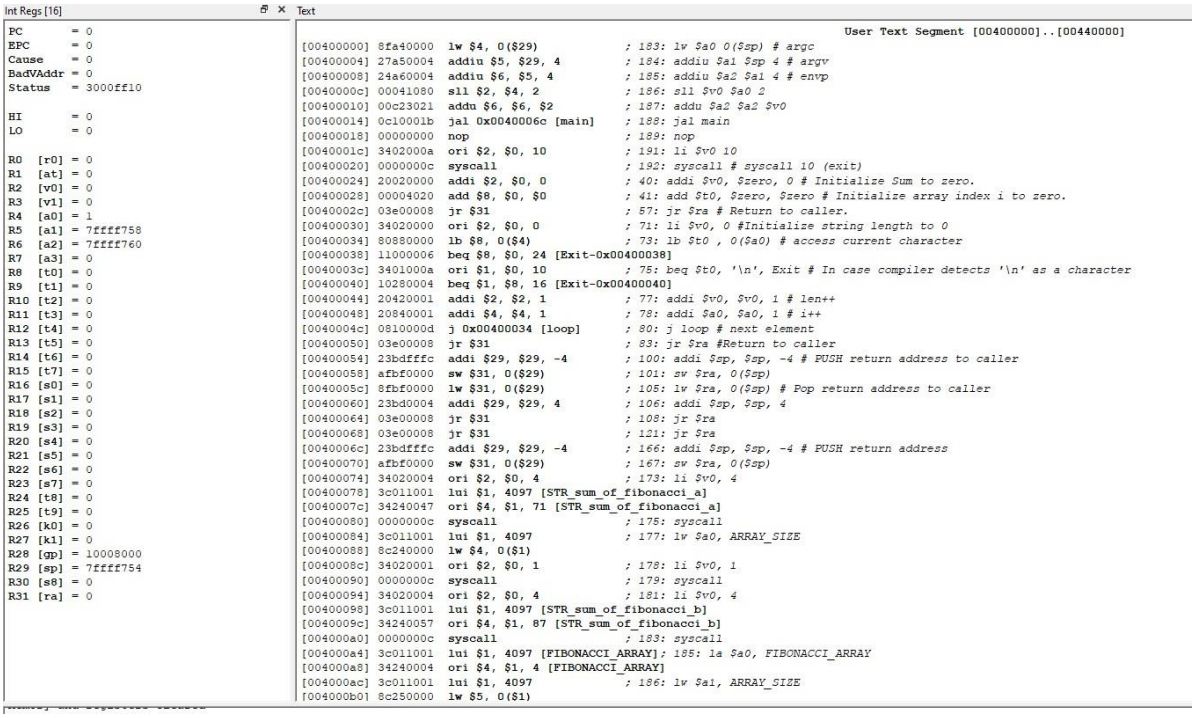
The problem to be solved in this second task of the laboratory activity is to create a subroutine wherein it has to take the memory address of a string and then return the number of characters in that given string. This means that through an iteration or loop, it keeps count of the number of characters in the string until it detects an exit code or a terminating NULL at the end of the string. The length of the string would then be returned to the caller to be printed in the console.

2 The Assembly Source Code

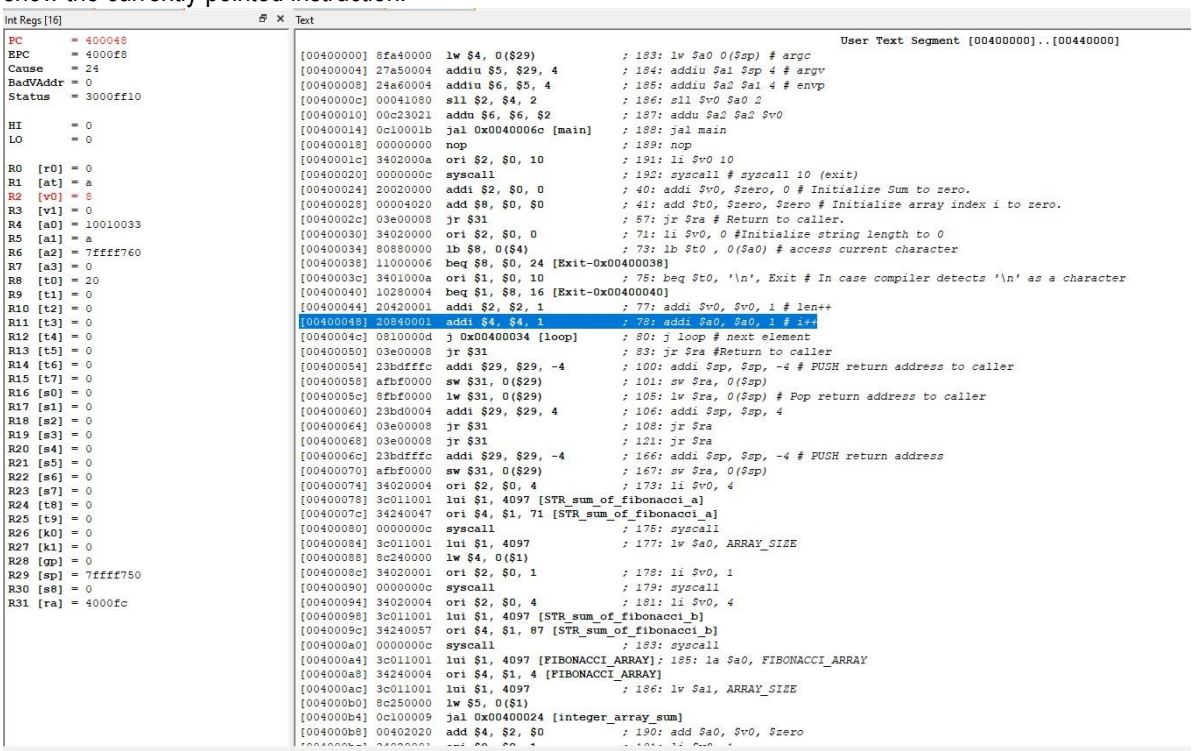
```
###############################################################################
#
# DESCRIPTION: Gives the length of a string.
#
#       INPUT: $a0 - address to a NUL terminated string.
#
#      OUTPUT: $v0 - length of the string (NUL excluded).
#
#    EXAMPLE:  string_length("abcdef") == 6.
#
###############################################################################
string_length:
    #### Write your solution here ####
    li $v0, 0               #Initialize string length to 0
loop:
    lb   $t0 , 0($a0)       # access current character
    beqz $t0, Exit          # check for null character
    beq  $t0, '\n', Exit    # In case compiler detects '\n' as a character

    addi $v0, $v0, 1        # len++
    addi $a0, $a0, 1        # i++

    j    loop               # next element

Exit:
    jr  $ra                 #Return to caller

###############################################################################
```

3 Screenshots

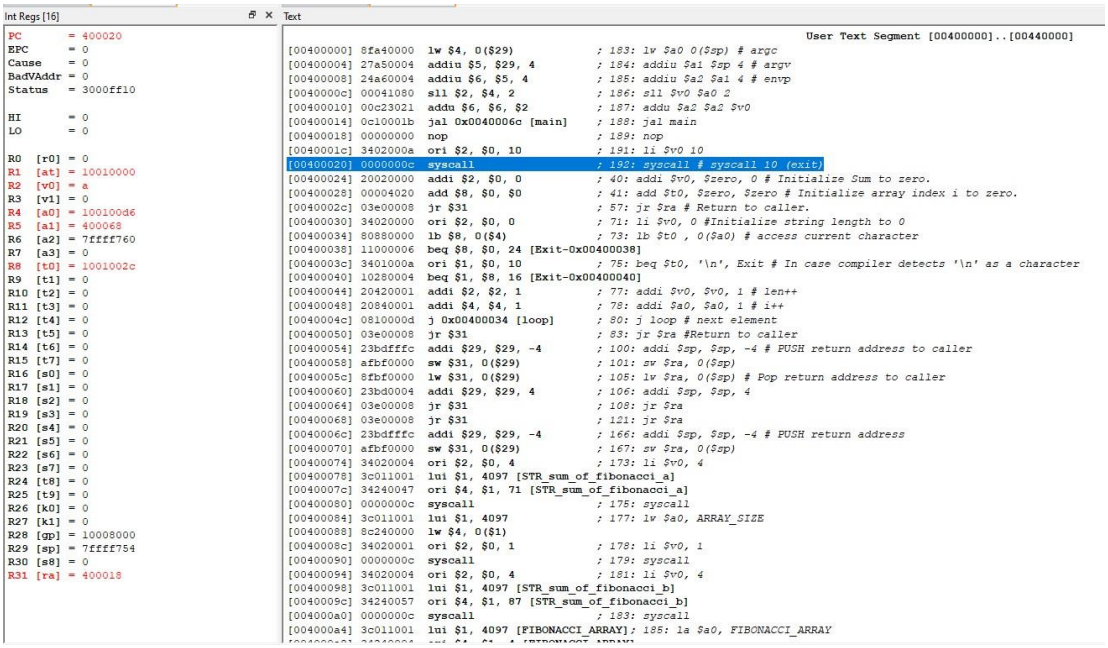
A. **Start of Execution** – Show the current state of the registers, code/program/text segment and data and stack segments in the memory after the program is loaded. Show the initial values of the registers. Show that your program and data has been loaded in the code and the data segments respectively.



B. **Middle of Execution** – Show the current state of the registers and data and stack segments mid-execution. Also show the currently pointed instruction.



C. **End of Execution** – Show the current state of registers and data and stack segments after the execution of the last instruction of your program. Also show the currently pointed instruction after the execution of the last instruction.



4 Learning & Insights

Through this laboratory activity, I learned how to deal with strings in MIPS assembly specifically in the case of how to end the loop when it comes to checking every character in the string, in this case it would be the terminating NULL at the end of the string depicted by ".asciiz" when we first initialize the string.

In looping through the characters of a string, we first had to find a way to access those characters. Through this task, I learned that instead of using "lw" or load word, we had to use "lb" or load byte as each character in a string is one byte compared to words which are 4 bytes. To access the next character, we simply had to increment the address by 1 in each loop.

I also learned how to use "li" or load immediate which I think is just a way to initialize values. There was also "beqz" which is different from "beq" since it only takes one address instead of two. It helps if you are trying to determine if that address equates to 0 or NULL.

5 Comparison to a High-Level Implementation

Here is an implementation of the lab problem in the language C.

```c
#include <stdio.h>

int string_length(char str[]){
    int length = 0;

    for (int i = 0; str[i] != '\0'; ++i){
        length += 1;
    }
    return length;
}
int main(){
    char Str[] = "Hunden, Katten, Glassen";
    printf("str = %s\n", Str);
    printf("string_length(str) = %d", string_length(Str));
    return 0;
}
```

Similar to our assembly source code, the main acts as the caller and the function sum acts as the callee. In looping through a string, compared to C where we had to check if the “i” in that array of characters equates to “\0” which is treated as a NULL character, in MIPS we didn’t have to explicitly state the NULL character as the string is already a NULL terminated string since it was initialized with .asciiz. To detect the NULL character in the string, we simply had to use “beqz” to check if the current character equated to 0 or NULL.

In terms of printing the output, it is also a bit different since in C we didn’t have to do instructions such as add, load immediate, or syscall, as we simply had to write printf and enclosed within are the strings or values we want to print out.

6 Conclusion

Overall, this laboratory activity helped polish what I learned from the lecture videos provided, it enabled me to understand MIPS assembly by hand in regards to strings. Similar to other programming languages, I realized that it is important to understand MIPS Assembly as it will assist me in future projects especially those that includes the specific use of addresses, memories and registers. This particular task helped in discovering how the program works around the manipulation of string variables in MIPS and it exposed me to other instructions that are very useful when it comes to strings such as “beqz” for detecting NULL’s and “lb” loading a character in a string.