

# MIPS Assignment 1

ECE 4612

Leo Battalora

2021/09/21

## Table of Contents

<b>Objectives</b>	<b>2</b>
<b>Tools/Equipment</b>	<b>2</b>
<b>Analysis/Algorithm/Procedure</b>	<b>2</b>
Task 1: Copy a string of characters	2
Task 2: Find factorial of a number	3
Task 3: Print a diamond shape to the I/O window	3
<b>Results/Observations</b>	<b>3</b>
<b>Conclusions</b>	<b>4</b>

# Objectives

This lab asked students to complete three tasks to practice the fundamentals of programming using the MIPS32 Instruction Set. These tasks included copying a string of characters from one memory location to another, calculating the factorial of a number inputted by the user, and drawing a diamond shape to the I/O window with the dimensions specified by the user.

Skills practiced include: moving data between registers, reading and writing to and from memory, issuing system calls, performing basic arithmetic, branching, and calling functions and subroutines.

## Tools/Equipment

Mips Assembler and Runtime Simulator (MARS) version 4.5 was used to write, assemble, execute, and debug MIPS32 based programs. Developed by Pete Sanderson and Kenneth Vollmar, MARS is a compiled .jar which was executed using JDK 11, the open-source reference implementation of version 11 of the Java SE Platform.

## Analysis/Algorithm/Procedure

This heading is sectioned into three parts, one for each of the tasks specified in the assignment. Each subsection will define the task, explain any algorithms used, and detail the program procedure.

### Task 1: Copy a string of characters

Task 1 asked students to write MIPS code to copy the content of a text array 'y' which contains their full name, to array 'x'.

A string copy (strcpy) subroutine was written which assumes registers \$a0 and \$a1 hold the address for where the first character of the source and destination strings are stored. The subroutine will loop through each character in the source string, copying it to the destination until the null terminator is reached. The implemented approach does have the potential to run into an infinite loop in the case that the source string never null terminates, or if the destination address intersects with the source string's memory space.

The main function loads the addresses of the two text array locations, then calls the strcpy subroutine. Once the subroutine returns, the program exits using system call code 10 (exit).

## Task 2: Find factorial of a number

Task 2 asked students to write MIPS code to calculate the factorial of a given input number. The program should prompt the user for an input before calculating the result.

A factorial function was written which takes the argument in register \$a0 to be the integer to calculate the factorial of, then returns the result in register \$v0. The function initializes the result to 1, then multiplies it repeatedly with the natural numbers. This process is stopped when the sentinel counter has exceeded the input number. This process will return 1 for all inputs less than or equal to 0.

The main function prompts the user for an input number, reads in the next integer, then passes this value to the factorial function. Once the function returns, the result is displayed and the program exits.

## Task 3: Print a diamond shape to the I/O window

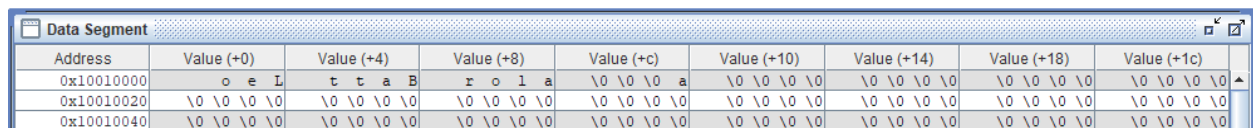
Task 3 asked students to write MIPS code to take an integer input and draw a diamond using asterisks with side lengths of the read integer.

Two subroutines were written to handle the logic of printing the diamond (print\_diamond), and printing each line of the diamond (print\_line). The print\_diamond subroutine uses the diamond's side length (argument \$a0) and a row counter to calculate the position of each asterisk on each line of the diamond. Once these positions are calculated, it passes the positions to the print\_line subroutine which prints spaces at the character indexes where no asterisk is specified. The print\_diamond subroutine runs two loops: the first to print the top half of the diamond as it widens, and a second to print the bottom half of the diamond as it tapers.

The main function prompts the user for an input number in the range [1,11], then passes this value to the print\_diamond subroutine. After the subroutine completes, the program exits.

## Results/Observations

Task 1 assembled without errors. Figure 1 below shows the Data segment analyzer before the program is executed. One can see one copy of the string "Leo Battalora" in memory. After execution, the data segment analyzer showed the copied string at both the source and destination addresses.



Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	o e L	t t a B	r o l a	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010020	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010040	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0

Figure 1: Data segment analyzer before executing task\_01.asm

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	o e L	t t a B	r o l a	e L \0 a	a B o	l a t t	\0 a r o	\0 \0 \0 \0
0x10010020	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010040	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0

Figure 2: Data segment analyzer after executing task\_01.asm

Task 2 assembled without errors. Figure 3 below shows the Run I/O window after the program is executed with an input of 4. The result 24 is equal to  $4 * 3 * 2 * 1$ , or  $4!$ .

```

Mars Messages  Run I/O
Enter a number to find factorial: 4
Result: 24
-- program is finished running --

```

Figure 3: Run I/O window after executing task\_02.asm

Task 3 assembled without errors. Figure 4 below shows the Run I/O window after the program is executed with an input of 4. The result is a diamond shape with side lengths of 4 asterisks.

```

Mars Messages  Run I/O
Input [1,11]: 4
  *
 * *
*  *
*   *
 *   *
  *   *
   *   *
    *
-- program is finished running --

```

Figure 4: Run I/O window after executing task\_03.asm

## Conclusions

This assignment served as a great introduction to programming in the MIPS32 instruction set. It exposed students to a variety of operations including declaring data segments, pushing to and popping off the stack, moving data between registers, reading and writing to and from memory, issuing system calls, performing basic arithmetic, branching, and calling functions and subroutines.