 **California State University Long Beach**

**College of Engineering**

**Computer Engineering Computer Science Department**

**Thursday December 6, 2016**

**Computer Architecture**

**Lab Project -3**

**Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Table of Contents

[Problem Definition: 3](#_Toc466962712)

[1) Program it in MARS 3](#_Toc466962713)

[2) Explain its behavior 3](#_Toc466962714)

[MARS Commented Source Code: 4](#_Toc466962715)

[Snapshot Outputs (For each part above): 5](#_Toc466962716)

[1) Register Picture 5](#_Toc466962717)

[2) Memory Picture 6](#_Toc466962718)

[3) Console Picture: 7](#_Toc466962719)

[Problem Explanation and behavior. 8](#_Toc466962720)

Problem Definition:

# Problem Definition:

In MARS do the following:

1. Create an array in memory that contains the following numbers:

1 – 2 – 5- 7- 10

1. Write a program that iterate through the array and calculate the factorial
   1. Create a function that recursively calculates the factorial for that number.
2. Analyze and describe the solution.

Problem Solution:

# MARS Commented Source Code:

# Snapshot Outputs (For each part above):

## Register Picture

## Memory Picture

## Console Picture:

# Problem Explanation and behavior.

We start off the program initializing the values whose factorials we are going to calculate, and the spaces we are going to store the values in. The we take the addresses of these allocated memory spaces and store them in $s0 and $s1 respectively. Second, we initialize a value which will be how many times we will loop which will be 5 due to having 5 elements in our array. So, in our loop we load the element in currently pointed to by $s0, and store it in $a0 which is the argument register. Then we use the Jump and Link (jal) instruction to call our factorial function which will give us a result stored in register $v0. Then both $s0 and $s1 addresses are incremented by 1 which will be the size of 4 bytes in MIPS so we do an add immediate (e.g. addi $s1, s1, 4), and decrement $t3 then loop back to the beginning until our $t3 reaches 0 in which the program will fall through and jump to the end: function which is at the bottom of the program.

So, the factorial function is a recursive function and starts out with setting the stack pointer back 8 bytes to store a return address and store an argument. Then it tests for the base case of n < 1 if it is less than 1 it sets $t0 to 1 making the next instruction not jump to the L1 subroutine. If this is the case, the program will continue and return a 1 and pop it’s saved return address and argument. This is the base case.

If the value of a0 is > 1 the function L1 is called which subtracts a0 – 1 and calls the “fact” function with the new value, and until when the base case is reached in which all the calls into L1 will continue each loading what was stored in the stack, and multiplying them storing the final answer in v0.