# CMPE 140 – Laboratory Assignment 4

**Dr. Donald Hung**
**Computer Engineering Department, San Jose State University**

## MIPS Programming (3):

## Array Processing, Stack and Recursive Procedure

### Purpose

Write MIPS assembly code to build a 50-entry array with the base address 0x100. You will need to access the array to perform some arithmetic calculations; the result of the calculation will be used as the input argument to a MIPS assembly program for the factorial function. This assignment should familiarize you with the MIPS implementation of arrays, stacks, procedures, and recursive procedures. Use this assignment to familiarize yourself with the MIPS ISA, assembly programming, as well as testing.

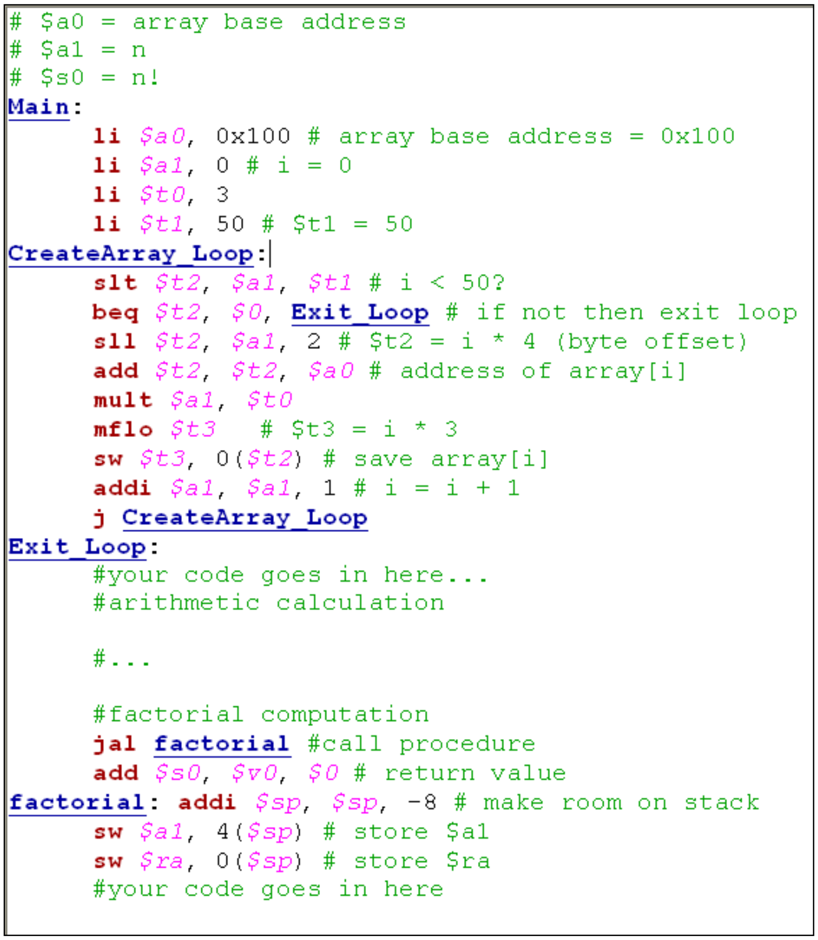
### Tasks

1) Write a MIPS assembly program to perform arithmetic expressions and compute the factorial of a number using a recursive procedure. The C++ pseudo code is given below:

```
void main()
{
      int n, f;
      int my_array[50];
      // Create the array
      for(i=0; i<50; i=i+1)
      {
            my_array[i] = i*3;
      }
      /*You will write MIPS code for the following parts*/
      // Arithmetic calculation
      n = (my_array[25]+ my_array[30])/30;
      // Factorial
      f = Factorial(n);
      return;
}

// Recursive factorial procedure
int Factorial(int n)
{
      if (n <= 1)
            return 1;
      else
            return (n*Factorial(n-1));
}
```

MIPS pseudo code (on next page):



```
# $a0 = array base address
# $a1 = n
# $s0 = n!
Main:
        li $a0, 0x100 # array base address = 0x100
        li $a1, 0 # i = 0
        li $t0, 3
        li $t1, 50 # $t1 = 50
CreateArray_Loop:
        slt $t2, $a1, $t1 # i < 50?
        beq $t2, $0, Exit_Loop # if not then exit loop
        sll $t2, $a1, 2 # $t2 = i * 4 (byte offset)
        add $t2, $t2, $a0 # address of array[i]
        mult $a1, $t0
        mflo $t3   # $t3 = i * 3
        sw $t3, 0($t2) # save array[i]
        addi $a1, $a1, 1 # i = i + 1
        j CreateArray_Loop
Exit_Loop:
        #your code goes in here...
        #arithmetic calculation

        #...

        #factorial computation
        jal factorial #call procedure
        add $s0, $v0, $0 # return value
factorial: addi $sp, $sp, -8 # make room on stack
        sw $a1, 4($sp) # store $a1
        sw $ra, 0($sp) # store $ra
        #your code goes in here
```

Requirements:

1. Your MIPS code should be under the line "#your code goes in here..." as shown in the figure above.
2. Register assignments:
   $a1 ← n
   $a0 ← array base addr
   $s0 ← n!
3. Your factorial function must be implemented as a **recursive procedure**.
4. The final value of *n* obtained from the arithmetic calculation must be written to the memory location at address 0x00.
5. The factorial *n*! must be written to the memory location at address 0x10.

2) Assemble your MIPS assembly code, single-step execute through all instructions, and verify the contents of the relevant registers. Sketch a stack status diagram that shows the addresses, stack pointer position, and values of $a1 and $ra after each iteration. Record the execution results using the test log table on page 3. Report the value at the following memory addresses when the entire program is executed:

- 0x00 – 0x03 (Word Adr 0x00);
- 0x10 – 0x13 (Word Adr 0x10);

3) Write a report including everything described in (2), as well as relevant screen shots and necessary discussions.

# CMPE 140 Lab 4 Test Log

**Programmer's Names:** Harmander Sihra and Vincent Van

**Checked by:** Ryan Lucus, **Date:** 9/27/17

Record the observed contents of registers and data memory after each instruction is executed.

| Addr | MIPS Instruction | Machine Code | Registers | | | | Memory Content | |
|---|---|---|---|---|---|---|---|---|
| | | | $a1 | $sp | $ra | $v0 | [0x00] | [0x10] |
| 034 | addi $t0, $t0, 30 | 0x2108001E | 100 | 200 | 0 | 0 | 0 | 0 |
| 038 | lw $t2, 356($zero) | 0x8C0A0164 | 100 | 200 | 0 | 0 | 0 | 0 |
| 03c | lw $t3, 376($zero) | 0x8C0B0178 | 100 | 200 | 0 | 0 | 0 | 0 |
| 040 | add $t3, $t3, $t2 | 0x016A5820 | 100 | 200 | 0 | 0 | 0 | 0 |
| 044 | div $t3, $t0 | 0x0168001A | 100 | 200 | 0 | 0 | 0 | 0 |
| 048 | mflo $a1 | 0x00002812 | 5 | 200 | 0 | 0 | 0 | 0 |
| 04c | sw $a1, 0($zero) | 0xAC050000 | 5 | 200 | 0 | 0 | 5 | 0 |
| 050 | jal 0x0017 | 0x0C000017 | 5 | 200 | 0 | 0 | 5 | 0 |
| 054 | sw $v0, 16($zero) | 0xAC020010 | 5 | 200 | 54 | 78 | 5 | 78 |
| 058 | j 0x0028 | 0x08000028 | 5 | 200 | 54 | 78 | 5 | 78 |
| 05c | addi $sp, $sp, -8 | 0x23BDFFF8 | 5 | 1F8 | 54 | 0 | 5 | 0 |
| 060 | sw $a1, 4($sp) | 0xAFA50004 | 5 | 1F8 | 54 | 0 | 5 | 0 |
| 064 | sw $ra, 0($sp) | 0xAFBF0000 | 5 | 1F8 | 54 | 0 | 5 | 0 |
| 068 | beq $zero, $a1, 8 | 0x10050008 | 5 | 1F8 | 54 | 0 | 5 | 0 |
| 06c | addi $a1, $a1, -1 | 0x20A5FFFF | 4 | 1F8 | 54 | 0 | 5 | 0 |
| 070 | jal 0x0017 | 0x0C000017 | 4 | 1F8 | 74 | 0 | 5 | 0 |
| 074 | lw $ra, 0($sp) | 0x8FBF0000 | 0 | 1D8 | 74 | 1 | 5 | 0 |
| 078 | lw $a1, 4($sp) | 0x8FA50004 | 1 | 1D8 | 74 | 1 | 5 | 0 |
| 07c | mult $a1, $v0 | 0x00A20018 | 1 | 1D8 | 74 | 1 | 5 | 0 |
| 080 | mflo $v0 | 0x00001012 | 1 | 1D8 | 74 | 1 | 5 | 0 |
| 084 | addi $sp, $sp, 8 | 0x23BD0008 | 1 | 1E0 | 74 | 1 | 5 | 0 |
| 088 | jr $ra | 0x03E00008 | 1 | 1E0 | 74 | 1 | 5 | 0 |
| 08c | lw $ra, 0($sp) | 0x8FBF0000 | 0 | 1D0 | 74 | 0 | 5 | 0 |
| 090 | lw $a1, 4($sp) | 0x8FA50004 | 0 | 1D0 | 74 | 0 | 5 | 0 |
| 094 | addi $sp, $sp, 8 | 0x23BD0008 | 0 | 1D8 | 74 | 0 | 5 | 0 |
| 098 | addi $v0, $v0, 1 | 0x20420001 | 0 | 1D8 | 74 | 1 | 5 | 0 |
| 09c | jr $ra | 0x03E00008 | 0 | 1D8 | 74 | 1 | 5 | 0 |
| 100 | | | | | | | | |
| 104 | | | | | | | | |