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:
    addi $a0, $0, 0x100     # array base address = 0x100
    addi $a1, $0, 0         # i = 0
    addi $t0, $0, 3
    addi $t1, $0, 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 the 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: _____

Checked by: _____, Date: _____

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								
038								
03c								
040								
044								
048								
04c								
050								
054								
058								
05c								
060								
064								
068								
06c								
070								
074								
078								
07c								
080								
084								
088								
08c								
090								
094								
098								
09c								
100								
104								