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: _____

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	lw $t2, 356($0)	8c0a0164	32	2ffc	0	0	0	0
038	lw $t3, 376($0)	8c0b0178	32	2ffc	0	0	0	0
03c	add $t2, $t2, $t3	014b5020	32	2ffc	0	0	0	0
040	addiu $t3, $0, 30	240b001e	32	2ffc	0	0	0	0
044	div $t2, $t3	014b001a	32	2ffc	0	0	0	0
048	mflo $a1	00002812	5	2ffc	0	0	0	0
04c	addi $v0, $0, 1	20020001	5	2ffc	0	1	0	0
050	jal factorial	0c000c17	5	2ffc	054	1	0	0
054	add $s0, $v0, $0	00408020	5	2ffc	054	78	0	0
058	j end	080000c28	5	2ffc	054	78	0	0
05c	factorial: addi $sp, $sp, -8	23bdfff8	5	2ff4	054	1	0	0
060	sw $a1, 4($sp)	afa50004	5	2ff4	054	1	0	0
064	sw $ra, 0($sp)	afbf0000	5	2ff4	054	1	0	0
068	addi $t4, $0, 2	200c0002	5	2ff4	054	1	0	0
06c	slt $t4, $a1, $t4	00ac602a	5	2ff4	054	1	0	0
070	beq $t4, $0, else	11800003	5	2ff4	054	1	0	0
074	addi $v0, $0, 1	20020001	1	2fd4	088	1	0	0
078	addi $sp, $sp, 8	23bd0008	1	2fdc	088	1	0	0
07c	jr $ra	03e00008	1	2fdc	088	1	0	0
080	else: addi $a1, $a1, -1	20a5ffff	4	2ff4	054	1	0	0
084	jal factorial	0c000c17	4	2ff4	088	1	0	0
088	lw $ra, 0($sp)	8fbf0000	1	2fdc	088	1	0	0
08c	lw $a1, 4($sp)	8fa50004	2	2fdc	088	1	0	0
090	addi $sp, $sp, 8	23bd0008	2	2fe4	088	1	0	0
094	mult $v0, $a1	00450018	2	2fe4	088	1	0	0
098	mflo $v0	00001012	2	2fe4	088	2	0	0
09c	jr $ra	03e00008	2	2fe4	088	2	0	0
100	end: sw $a1, 0($0)	ac050000	5	2ffc	054	78	5	0
104	sw $s0, 16($0)	ac100010	5	2ffc	054	78	5	78