***Design Methodology***

This lab has two major parts, first task was to build a 50-entry array, second being to step through each iteration and record the results. The array was to be used to calculate arithmetical calculations. The programs main purpose is to calculate the factorial using the recursive method. The pseudo code to be implemented required the final value to be located at address 0x00, and the n! to be written to memory location 0x10. Each instruction was executed and put into the log results. Source code, screenshots and log results are presented below. Lastly we had to find out the contents of 0x00 – 0x03 (Word Adr 0x00); and 0x10 – 0x13 (Word Adr 0x10). The outcome of the lab was to be learn the recursive algorithm using the MIPS assembly programs. After this lab it was easier to understand the stack pointer position, how to properly construct a stack status diagram, MIPS implementation of arrays, stacks, procedures, and recursive procedures.

Tasks which were completed in lab:

* MIPS program to perform arithmetic expressions of a factorial using an array.
* MIPS program compute the factorial of a number using a recursive procedure.
* Draw a stack status diagram which represents the addresses, stack pointer, including the values of $a1 and $ra values.
* Create a testing log for each execution iteration of the program.
* Obtain the values at memory addresses after full execution of the program.
  + 0x00 - 0x03
  + 0x10 - 0x13

Stack Status Diagram:

Address Data

|  |  |  |
| --- | --- | --- |
| FC |  | ←$sp $v0 = 78h |
| F8 | $a1 (0x5) |  |
| F4 | $ra (3054) | ←$sp $a1 = 5 , $v0 = 24 \* 5 |
| F0 | $a1 (0x4) |  |
| EC | $ra (3088) | ←$sp $a1 = 4, $v0 = 6 \* 4 |
| E8 | $a1 (0x3) |  |
| E4 | $ra (3088) | ←$sp $a1 = 3, $v0 = 3 \* 2 |
| E0 | $a1 (0x2) |  |
| DC | $ra (3088) | ←$sp $a1 = 2, $v0 = 2 \* 1 |
| D8 | $a1 (0x1) |  |
| D4 | $ra (3088) | ←$sp $a1 = 1, $v0 = 1 \* 1 |

Cmpe 140 Lab 4 Test Log:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 | 3054 | 1 | 0 | 0 |
| 054 | Add $s0, $v0, $0 | 00408020 | 5 | 2ffc | 3054 | 78 | 0 | 0 |
| 058 | J end | 08000c26 | 5 | 2ffc | 3054 | 78 | 0 | 0 |
| 05c | factorial: addi $s0, $v0, $0 | 00003098 | 5 | 2ff4 | 3054 | 1 | 0 | 0 |
| 060 | Sw $a1, 4($sp) | afa50004 | 5 | 2ff4 | 3054 | 1 | 0 | 0 |
| 064 | Sw $ra, 0($sp) | afbf0000 | 5 | 2ff4 | 3054 | 1 | 0 | 0 |
| 068 | Slti $t4, $a1, 2 | 28ac0002 | 5 | 2ff4 | 3054 | 1 | 0 | 0 |
| 070 | Addi $sp, $sp, 8 | 23bd0008 | 5 | 2ff4 | 3054 | 1 | 0 | 0 |
| 074 | Jr $ra | 03e00008 | 1 | 2fdc | 3080 | 1 | 0 | 0 |
| 078 | Addi $a1, $a1, -1 | 20a5ffff | 4 | 2ff4 | 3054 | 1 | 0 | 0 |
| 07c | Jal factorial | 0c000c17 | 4 | 2ff4 | 3080 | 1 | 0 | 0 |
| 080 | Lw $ra, 0($sp) | 8fbf0000 | 1 | 2fdc | 3080 | 1 | 0 | 0 |
| 084 | Lw $a1, 4($sp) | 8fa50004 | 2 | 2fdc | 3080 | 1 | 0 | 0 |
| 088 | Addi $sp, $sp, 8 | 23bd0008 | 2 | 2fe4 | 3080 | 1 | 0 | 0 |
| 08c | Mult $v0, $a1 | 00450018 | 2 | 2fe4 | 3080 | 1 | 0 | 0 |
| 090 | Mflo $v0 | 00001012 | 2 | 2fe4 | 3080 | 2 | 0 | 0 |
| 094 | Jr $ra | 03e00008 | 2 | 2fe4 | 3080 | 2 | 0 | 0 |
| 098 | Sw $a1, 0($0) | Ac050000 | 5 | 2ffc | 3054 | 78 | 5 | 0 |
| 09c | Sw $s0, 16($0) | Ac100010 | 5 | 2ffc | 3054 | 78 | 5 | 78 |

Source Code:

# $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

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:

#my code

lw $t2, 356($0) #t2 = my\_array[25]

lw $t3, 376($0) #t3 = my\_array[30]

add $t2, $t2, $t3 #t2 = t2 + t3

addiu $t3, $0, 30 #t3 = 30

div $t2, $t3 # (my\_array[25] + my\_array[30]) / 30

mflo $a1 # n = (my\_array[25] + my\_array[30]) / 30

addi $v0, $0, 1

jal factorial #call procedure

add $s0, $v0, $0 #return value

j end

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

slti $t4, $a1, 2

beq $t4, $0, else

addi $sp, $sp, 8

jr $ra

else: addi $a1, $a1, -1

jal factorial

lw $ra, 0($sp)

lw $a1, 4($sp)

addi $sp, $sp, 8

mult $v0, $a1

mflo $v0

jr $ra

end: sw $a1, 0($0)

sw $s0, 16($0)

***Screenshot***

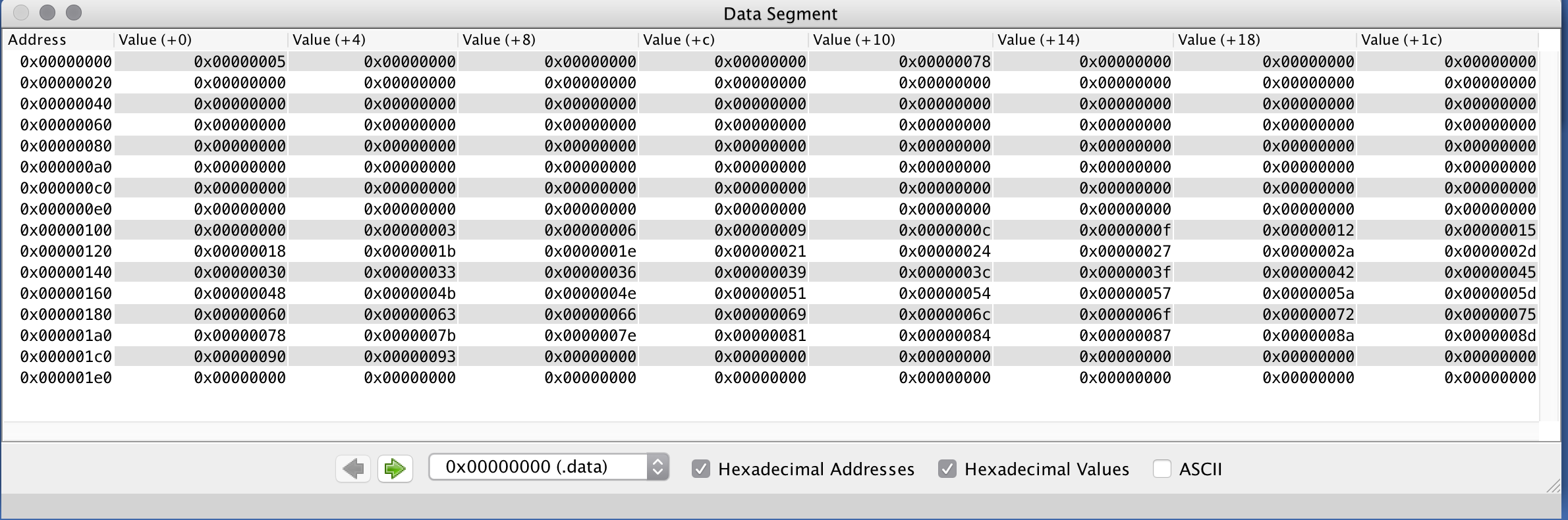


Figure 1: Data Screenshot

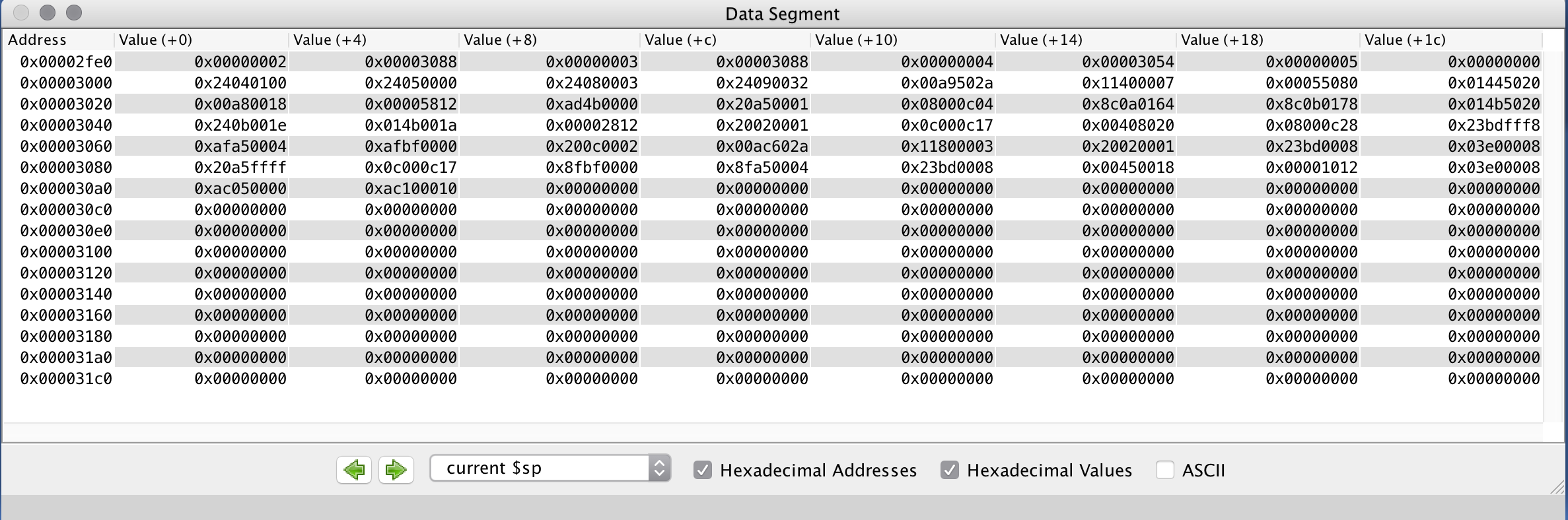


Figure 2: Sp Screenshot