***Design Discussion***

In this lab my partner and I had to write two MIPS assembly program that performs arithmetic computations. One of the programs included a factorial program to calculate the factorial of the number *n* (*n!* = *n\**(*n*-1)\*.....\*1). While each instruction was executed we had to trace the output of each register being used in the program. The source code and output test results table are provided below. We learned more about writing MIPS assembly programs, specifically how to write while statements and execute arithmetic computations. Also, we learned more about how to step through and observe the contents of the different registers in a MIPS assembly program.

The list of tasks were fully completed in this lab:

* Write a MIPS Assembly program to perform the arithmetic computation provided in the source code
* Assemble the MIPS Assembly code and single-step execute through all of the instructions. Verify the contents of each relevant register.
* Write a MIPS assembly program to calculate the factorial of a given integer n
* Assemble the factorial MIPS Assembly code and single-step execute through all of the instructions. Verify the contents of each relevant register.

Source Code:

1.

main: addiu $4, $0, 0x8000 #a = 0x8000

addiu $5, $0, 0x00A9 #b = 0x00A9

addiu $16, $0, 1974 #c = 1974

multu $4, $4 #a \* a

mflo $17 #x = a \* a

sw $17,0x20($0) #store x in 0x20

multu $17, $5 #x \* b

mflo $18 #y = lo

mfhi $9 #t1 = hi

sw $18, 0x24($0) #store y lo in 0x24

sw $9, 0x28($0) #store t1 hi in 0x28

srl $18, $18, 16 #y = y >> 16

sll $9, $9, 16 #t1 << 16

xor $18, $18, $9 #y = y xor t1

addiu $9, $0, 1

jal addloop #jump down to the addloop

sw $16, 0x2c($0) #store c into mem 0x2c

loop: slti $9, $16, 1665 #t1 = (s0 < 1665) 1 : 0

beq $9, $0, addloop #if(t0 == 0)

j end

addloop: divu $18, $16 #y / c

mflo $8 #t0 = y / c

addu $8, $16, $8 #t0 = (c + y / c)

srl $16, $8, 1 #t0 / 2

beq $9, $0, loop

jr $ra

end: sll $16, $16, 8 #c = c << 8

sw $16, 48($0) #store c in mem 0x30

2.

main: addiu $4, $0, 5 #n = 5

sw $4, 0($0) #n into 0x00

addiu $16, $0, 1 #f = 1

whileloop: slti $8, $4, 1

bne $8, $0, end

multu $4, $16 #f \* n

mflo $16 #f = f \* n

addiu $9, $0, 1

subu $4, $4, $9 #n = n - 1

j whileloop

end: sw $16, 16($0) #store n! into 0x10

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ADR. | MIPS Instruction | Machine Code | Registers | | | | | |
| $a0 | $a1 | $s0 | $s1 | | $s2 |
| 00 | Addiu $a0, $zero, 0x8000 | 24048000 | 00008000 | 0 | 0 | 0 | 0 | |
| 04 | Addiu $a1, $zero, 0x00A9 | 240500A9 | 00008000 | 000000A9 | 0 | 0 | 0 | |
| 08 | Addiu $s0, $zero, 1974 | 241007B6 | 00008000 | 000000A9 | 000007B6 | 0 | 0 | |
| 0c | Multu $a0, $a0 | 00840019 | 00008000 | 000000A9 | 000007B6 | 40000000 | 0 | |
| 10 | Mflo $s1 | 00008812 | 00080000 | 000000A9 | 000007B6 | 40000000 | 0 | |
| 14 | sw $s1, 32($zero) | AC110020 | 00008000 | 000000A9 | 000007B6 | 40000000 | 0 | |
| 18 | Multu $s1, $a1 | 02250019 | 00008000 | 000000A9 | 000007B6 | 40000000 | 0 | |
| 1c | Mflo $s2 | 00009012 | 00008000 | 000000A9 | 000007B6 | 40000000 | 40000000 | |
| 20 | Mfhi $t1 | 00004810 | 00008000 | 000000A9 | 000007B6 | 40000000 | 40000000 | |
| 24 | Sw $s2, 36($zero) | AC120024 | 00008000 | 000000A9 | 000007B6 | 40000000 | 40000000 | |
| 28 | Sw $t1, 40($zero) | AC090028 | 00008000 | 000000A9 | 000007B6 | 40000000 | 40000000 | |
| 2c | Srl $s2, $s2, 16 | 00084402 | 00008000 | 000000A9 | 000007B6 | 40000000 | 00004000 | |
| 30 | Sll $t1, $t1, 16 | 00094C00 | 00008000 | 000000A9 | 000007B6 | 40000000 | 00004000 | |
| 34 | Xor $s2, $s2, $t1 | 02499026 | 00008000 | 000000A9 | 000007B6 | 40000000 | 002A4000 | |
| 38 | Addiu $t1, $zero, 1 | 24090001 | 00008000 | 000000A9 | 000007B6 | 40000000 | 002A4000 | |
| 3c | Jal addloop | 08000014 | 00008000 | 000000A9 | 000007B6 | 40000000 | 002A4000 | |
| 40 | Sw $s0, 44($zero) | AC100026 | 00008000 | 000000A9 | 00000698 | 40000000 | 002A4000 | |
| 44 | Slti $t1, $s0, 1665 | 2A090681 | 00008000 | 000000A9 | 00000698 | 40000000 | 002A4000 | |
| 48 | Beq $zero, $t1, addloop | 10090001 | 00008000 | 000000A9 | 00000698 | 40000000 | 002A4000 | |
| 4c | J end | 0800001A | 00008000 | 000000A9 | 00000680 | 40000000 | 002A4000 | |
| 50 | Divu $s2, $s0 | 02500018 | 00008000 | 000000A9 | 000007B6 | 40000000 | 002A4000 | |
| 54 | Mflo $t0 | 00004012 | 00008000 | 000000A9 | 000007B6 | 40000000 | 002A4000 | |
| 58 | Addu $t0, $s0, $t0 | 02084021 | 00008000 | 000000A9 | 000007B6 | 40000000 | 002A4000 | |
| 5c | Srl $s0, $t0, 1 | 00088042 | 00008000 | 000000A9 | 00000698 | 40000000 | 002A4000 | |
| 60 | Bew $zero, $t1, loop | 1009FFF8 | 00008000 | 000000A9 | 00000698 | 40000000 | 002A4000 | |
| 64 | Jr $ra | 03E00008 | 00008000 | 000000A9 | 00000698 | 40000000 | 002A4000 | |
| 68 | Sll $s0, $s0, 8 | 00108200 | 00008000 | 000000A9 | 00068000 | 40000000 | 002A4000 | |
| 6c | Sw $s0, 48($zero) | AC100030 | 00008000 | 000000A9 | 00068000 | 40000000 | 002A4000 | |
|  |  |  |  |  |  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Memory Content | | | |
| Word @0x20 | Word @0x24 | Word @0x2c | Word @0x30 |
| 000000A9 | 40000000 | 00000698 | 00068000 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Adr | MIPS Instruction | Machine Code | Registers | | Memory Content | |
| $a0 | $s0 | Word @0x00 | Word @ 0x10 |
| 00 | Addiu $a0, $zero, 5 | 24040005 | 5 | 0 | 00000000 | 00000000 |
| 04 | Sw $a0, 0($zero) | AC040000 | 5 | 0 | 00000005 | 00000000 |
| 08 | Addiu $s0, $zero, 1 | 24100001 | 5 | 1 | 00000005 | 00000000 |
| 0c | Slti $t0, $a0, 1 | 28880001 | 5 | 1 | 00000005 | 00000000 |
| 10 | Bne $t0, $zero, end | 14080005 | 5 | 1 | 00000005 | 00000000 |
| 14 | Multu $a0, $s0 | 00900019 | 5 | 1 | 00000005 | 00000000 |
| 18 | Mflo $0 | 00008012 | 5 | 5 | 00000005 | 00000000 |
| 1c | Addiu $t1, $zero, 1 | 24090001 | 5 | 5 | 00000005 | 00000000 |
| 20 | Subu $a0, $a0, $t1 | 00892023 | 4 | 5 | 00000005 | 00000000 |
| 24 | J whileloop | 08000003 | 4 | 5 | 00000005 | 00000000 |
| 28 | Sw $a0, 16($zero) | AC100010 | 0 | 00000078 | 00000005 | 00000078 |

Screen Captures:

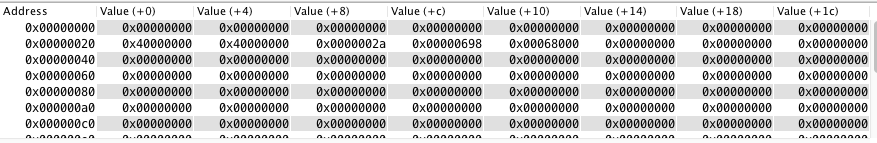


Figure 1. C++ Psuedo Code Data Segment

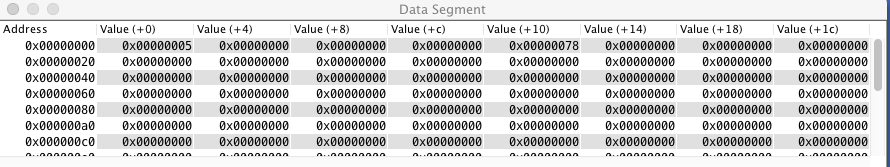


Figure 2. Factorial Data Segment