

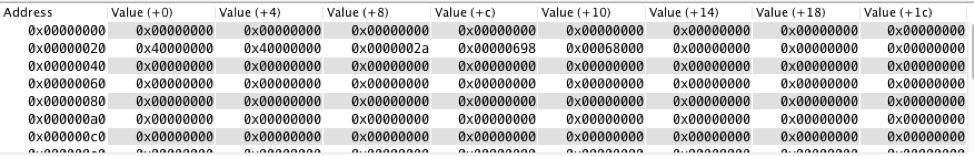
***Design Discussion***

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.

***Task1***

The first task of this assignment was to convert the given C++ pseudo code into MIPS Assembly. The pseudo code performed a series of seemingly trivial operations that are increasingly more complicated in MIPS. Once the MIPS program is written, students were to assemble it and single step through the program and record register values. We originally had an issue where our results did not match the expected values. This was a result of the way the assembler processed a certain pseudo command. Once this was changed manually to represent the originally intended functionality, the results matched their expected values.

Figure 1: Output of C++ Pseudo Code Memory Segment

***Task2***

The second task was to convert a C++ factorial calculator algorithm into MiPS assembly. As with the first task, the code was to be assembled and stepped through to ensure registers contained their expected values. This task was completed on the first attempt without any issues.

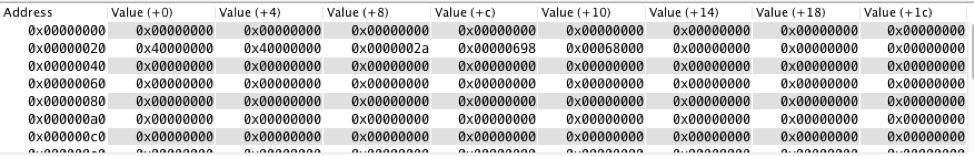


Figure 2: Output of Factorial Memory Segment

***Conclusion:***

This Lab was a good introduction to the branching features of MIPS. It also served as a good learning instance as to the differences between real MIPS commands and pseudo MIPS commands. Although there were some initial issues with the results of the first task, they were easily resolved. All other aspects of this assignment went without issues.

***Appendix:***

| ADR | MIPS Instruction | Machine Code | Registers | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| $a0 | $a1 | $s0 | $s1 | $s2 |
| 00 | lui $1, 0x000 | 0x3c010000 | 0 | 0 | 0 | 0 | 0 |
| 04 | ori $1, $1, 0x00008000 | 0x34218000 | 0 | 0 | 0 | 0 | 0 |
| 08 | addu $4, $0, $1 | 0x00012021 | 0x8000 | 0 | 0 | 0 | 0 |
| 0C | addiu $5, $0, 0xa9 | 0x240500a9 | 0x8000 | 0xa9 | 0 | 0 | 0 |
| 10 | addiu $16, $0, 0x000007b6 | 0241007b6 | 0x8000 | 0xa9 | 0x7b6 | 0 | 0 |
| 14 | mult $4, $4 | 0x00840018 | 0x8000 | 0xa9 | 0x7b6 | 0 | 0 |
| 18 | mflo $17 | 0x00008812 | 0x8000 | 0xa9 | 0x7b6 | 0x4000000 | 0 |
| 1c | sw $17, 0x20($0) | 0xac110020 | 0x8000 | 0xa9 | 0x7b6 | 0x40 | 0 |
| 20 | mult $17, $5 | 0x02250018 | 0x8000 | 0xa9 | 0x7b6 | 0x40… | 0 |
| 24 | mfhi $8 | 0x00004010 | 0x8000 | 0xa9 | 0x7b6 | 0x40… | 0 |
| 28 | sw $8, 0x24($0) | 0xac080024 | 0x8000 | 0xa9 | 0x7b6 | 0x40… | 0 |
| 2C | mflo $9 | 0x00004812 | 0x8000 | 0xa9 | 0x7b6 | 0x40… | 0 |
| 30 | sw $9, 0x28($0) | 0xac090028 | 0x8000 | 0xa9 | 0x7b6 | 0x40… | 0 |
| 34 | sll $8,$8, 0x10 | 0x00084400 | 0x8000 | 0xa9 | 0x7b6 | 0x40… | 0 |
| 38 | srl $9, $9, 0x10 | 0x00094ac02 | 0x8000 | 0xa9 | 0x7b6 | 0x40… | 0 |
| 3c | or $18, $8, $9 | 0x01099025 | 0x8000 | 0xa9 | 0x7b6 | 0x40… | 0x002a4000 |
| 40 | div $18, $16 | 0x0240001a | 0x8000 | 0xa9 | 0x7b6 | 0x40… | 0x002a4000 |
| 44 | mflo $8 | 0x00004012 | 0x8000 | 0xa9 | 0x7b6 | 0x40… | 0x002a4000 |
| 48 | add $16, $8, $16 | 0x01108020 | 0x8000 | 0xa9 | 0x0d30 | 0x40… | 0x002a4000 |
| 4c | srl $16, $16, 0x01 | 0x00108042 | 0x8000 | 0xa9 | 0x0698 | 0x40… | 0x002a4000 |
| 50 | sw $16, 0x2c($0) | 0xac10002c | 0x8000 | 0xa9 | 0x0698 | 0x40… | 0x002a4000 |
| 54 | stli $8, $16, 0x0681 | 0x2a080681 | 0x8000 | 0xa9 | 0x0698 | 0x40… | 0x002a4000 |
| 58 | bne $8, $0, 0x05 | 0x15000005 | 0x8000 | 0xa9 | 0x0698 | 0x40… | 0x002a4000 |
| 5C | div $18, $16 | 0x0250001a | 0x8000 | 0xa9 | 0x0698 | 0x40… | 0x002a4000 |
| 60 | mflo $8 | 0x00004012 | 0x8000 | 0xa9 | 0x0698 | 0x40… | 0x002a4000 |
| 64 | add $16, $8, $16 | 0x01108020 | 0x8000 | 0xa9 | 0x0d00 | 0x40 | 0x002a4000 |
| 68 | srl $16, $16, 0x01 | 0x00108042 | 0x8000 | 0xa9 | 0x0d00 | 0x40 | 0x002a4000 |
| 6C | j 0x3054 | 0x08000c15 | 0x8000 | 0xa9 | 0x0d00 | 0x40 | 0x002a4000 |
| 70 | Sll $16, $16, 0x08 | 0x00108200 | 0x8000 | 0xa9 | 0x68000 | 0x40 | 0x002a4000 |
| 74 | Sw $16, 0x30($0) | 0xac100030 | 0x8000 | 0xa9 | 0x68000 | 0x40 | 0x002a4000 |

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

| Adr | MIPS Instruction | Machine Code |  | Registers | |  | Memory Content | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **$a0** | **$s0** | **$t0** | **$** | Word @ 0x00 | Word @ 0x10 |
| 00 | addi $a0, $0, 5 | 20040005 | 0 | 0 | 0 | 0 | 0 | 0 |
| 04 | sw $a0, 0x00 | ac040000 | 5 | 0 | 0 | 0 | 0 | 0 |
| 08 | addi $s0, $0, 1 | 20100001 | 5 | 0 | 0 | 0 | 5 | 0 |
| 0c | addi $t0, $0, 1 | 20080001 | 5 | 1 | 0 | 0 | 5 | 0 |
| 10 | beq $a0, $t0, 0x3024 | 10880004 | 5, 4, 3, 2, 1 | 1, 5, 20, 60, 120 | 1, 1, 1, 1, 1 | 0, 0, 0, 0, 0 | 5, 5, 5. 5, 5 | 0, 0, 0, 0, 0 |
| 14 | mult $a0, $s0 | 00900018 | 5, 4, 3, 2 | 1, 5, 20, 60 | 1, 1, 1, 1 | 0, 0, 0, 0 | 5, 5, 5, 5 | 0, 0, 0, 0 |
| 18 | mflo $s0 | 00008012 | 5, 4, 3, 2 | 1, 5, 20, 60 | 1, 1, 1, 1 | 0, 0, 0, 0 | 5, 5, 5, 5 | 0, 0, 0, 0 |
| 1c | addi $a0, $a0, -1 | 2084ffff | 5, 4, 3, 2 | 5, 20, 60, 120 | 1, 1, 1, 1 | 0, 0, 0, 0 | 5, 5, 5, 5 | 0, 0, 0, 0 |
| 20 | j 0x3010 | 08000c04 | 4, 3, 2, 1 | 5, 20, 60, 120 | 1, 1, 1, 1 | 0, 0, 0, 0 | 5, 5, 5, 5 | 0, 0, 0, 0 |
| 24 | Sw $s0, 0x10 | ac100010 | 1 | 120 | 1 | 0 | 5 | 120 |

| ***Task A Pseudo Code*** |
| --- |
| a = 0x8000; // MIPS instruction: addiu $a0, $0, 0x8000  b = 0x00A9;  c = 1974;  // store the value of x to memory location at address 0x20;  x = a \* a;    // store the value of y to memory location at address 0x24;  y = x \* b;  y = y >> 16;  // store the value of c to memory location at address 0x2C;  c = (c + y / c) / 2;  while(c >= 1665){  c = (c + y / c) / 2;  }  // store the value of c to memory location at 0x30;  c = c << 8; |

| ***Task A MIPS Code*** |
| --- |
| ########################  # Variable declaration #  ########################  # $a0 = a  # $a1 = b  # $s0 = c  # $s1 = x  # $s2 = y  #################  # Begin Program #  #################  #################  # Begin Program #  #################  addiu $4, $0, 0x8000 #a=0x8000  addiu $5, $0, 0x00A9 #b=0x00A9  addiu $16, $0, 1974 #c=1974  mult $4, $4 #a\*a  mflo $17 #store lo in x  sw $17, 0x20($0) #store x into 0x20  mult $17, $5 #multiply x\*b  mfhi $8 #store hi in temp  sw $8, 0x24($0) #store temp hi in 0x24  mflo $8 #store lo in temp  sw $8, 0x28($0) #store temp lo in 0x28  lui $18, 0x2A  srl $8, $8, 16  add $18, $18, $8 #y = 0x002A\_4000  div $18, $16 #y/c  mflo $8 #store quotient in c  add $16, $8, $16 #c = quotient + c  srl $16, $16, 1 #shift right one bit to divide c by 2  sw $16, 0x2C($0)  loop:  slti $8, $16, 1665  bne $8, $0, done  div $18, $16 #y/c  mflo $8 #store quotient in c  add $16, $8, $16 #c = quotient + c  srl $16, $16, 1 #shift right one bit to divide c by 2  j loop  done:  sll $16, $16, 8  sw $16, 0x30($0) |

| ***Task B Pseudo Code*** |
| --- |
| INPUT n = 5; //given number n  f = 1;  while (n > 1)  {  f = f \* n;  n = n - 1;  }  OUTPUT f; //factorial f = n! |

| ***Task B MIPS Code*** |
| --- |
| ########################  # Variable declaration #  ########################  # $a0 = n  # $a1 = f  #################  # Begin Program #  #################  var\_init: addi $a0, $0, 5 # assign 5 to $a0  sw $a0, 0x00  addi $s0, $0, 1 # assign 1 to $a1  addi $t0, $0, 1 # assign 1 to $t0  loop\_begin: beq $a0, $t0, end # branch to end if $a0 == s01  mult $a0, $s0 # $a1 = $a1 \* $a0  mflo $s0  addi $a0, $a0, -1  j loop\_begin  end: sw $s0, 0x10 |