San Jose State University
Department of Computer Engineering

CMPE 200 Report

Assignment 3 Report

Title MIPS Instruction Set Architecture & Programming (2)

Semester Fall 2022 **Date** 09/25/2022

by

Name Harish Marepalli *(typed)* **SID** 016707314 *(typed)*

INTRODUCTION:

This activity is used to write a MIPS assembly program to perform the arithmetic computation for a given C++ pseudo code and calculate the factorial of a given number. After writing the MIPS program, it has to be executed step by step to record the values of Machine code, registers and memory.

MY GROUP:

Name: Student_Team 6
Harish Marepalli – 016707314
Tirumala Saiteja Goruganthu – 016707210

SOURCE CODE (Task 1):

Task1: The total number of lines are 26

```
                ori $a0, $0, 0x8000     #line 1: store the value 0x8000 in a variable a
                ori $a1, $0, 0x00A9     #line 2: store the value 0x00A9 in a variable b
                ori $s0, $0, 1974       #line 3: store the value 1974 in a variable c
                multu $a0, $a0          #line 4: multiply a with a
                mflo $s1                #line 5: move li content to x. Store in s1 register (no
                                        overflow condition)
                addi $t0, $0, 0x20      #line 6: store the base address in a temporary register t0
                sw $s1, 0($t0)          #line 7: store x in a location 0x20
                multu $s1, $a1          #line 8: multiply x with b
                mfhi $s2                #line 9: move hi content to y
                sw $s1, 4($t0)          #line 10: store lo content in a location 0x24[y]
                sw $s2, 8($t0)          #line 11: store hi content in a location 0x28[y+4]
                srl $s2, $s1, 16        #line 12: right shift y.lo value
                jal compute             #line 13: jump to compute label to calculate the given
                                        formula
                sw $s0, 12($t0)         #line 14: store c value in a location 0x2c
                addi $t3, $0, 1         #line 15: store the value '1' in t2 register for future beq
                                        comparison
while:          slti $t1, $s0, 1665     #line 16: check if c<1665 => $t1=1 else $t1=0
                beq $t1, $t3, done      #line 17: branch to done if $t1==$t3 since we should come
                                        out of the while loop
                jal compute             #line 18: jump to compute label to calculate the given
                                        formula
                j while                 #line 19: loop back to while
compute:        divu $s2, $s0           #line 20: divide y with c. The quotient gets stored in lo and
                                        the remainder gets stored in hi
                mflo $t1                #line 21: move the quotient of previous result into t1 register
                add $t1, $s0, $t1       #line 22: add c to the previous result and store the result in a
                                        temporary t1 register
                srl $s0, $t1, 1         #line 23: right shift by '1' to essentially divide by 2
                jr $ra                  #line 24: jump to return address given by ra register
done:           sll $s0, $s0, 8         #line 25: logic left shift c value by 8 and store it back in c
                                        ($s0)
                sw $s0, 16($t0)         #line 26: store c value in a location 0x30
```

CMPE200 Assignment 3 Task 1 Test Log
Algorithm 1

Programmer's Name: Harish Marepalli

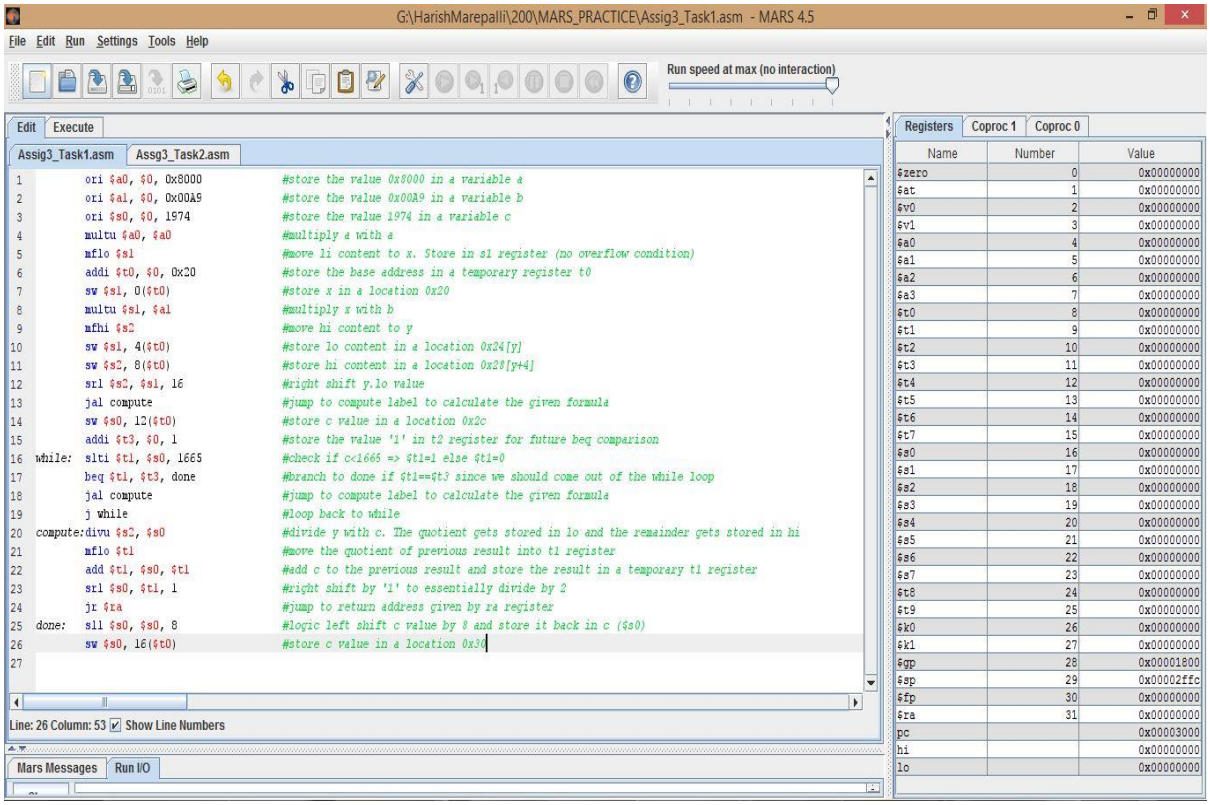
Date: 09/24/2022

Adr	MIPS Instruction	Machine Code	Registers				
			$a0	$a1	$s0	$s1	$s2
3000	ori $a0, $0, 0x8000	0x34048000	0x00008000	0x00000000	0x00000000	0x00000000	0x00000000
3004	ori $a1, $0, 0x00A9	0x340500a9	0x00008000	0x000000a9	0x00000000	0x00000000	0x00000000
3008	ori $s0, $0, 1974	0x341007b6	0x00008000	0x000000a9	0x000007b6	0x00000000	0x00000000
300c	multu $a0, $a0	0x00840019	0x00008000	0x000000a9	0x000007b6	0x00000000	0x00000000
3010	mflo $s1	0x00008812	0x00008000	0x000000a9	0x000007b6	0x40000000	0x00000000
3014	addi $t0, $0, 0x20	0x20080020	0x00008000	0x000000a9	0x000007b6	0x40000000	0x00000000
3018	sw $s1, 0($t0)	0xad110000	0x00008000	0x000000a9	0x000007b6	0x40000000	0x00000000
301c	multu $s1, $a1	0x02250019	0x00008000	0x000000a9	0x000007b6	0x40000000	0x00000000
3020	mfhi $s2	0x00009010	0x00008000	0x000000a9	0x000007b6	0x40000000	0x0000002a
3024	sw $s1, 4($t0)	0xad110004	0x00008000	0x000000a9	0x000007b6	0x40000000	0x0000002a
3028	sw $s2, 8($t0)	0xad120008	0x00008000	0x000000a9	0x000007b6	0x40000000	0x0000002a
302c	srl $s2, $s1, 16	0x00119402	0x00008000	0x000000a9	0x000007b6	0x40000000	0x00004000
3030	jal compute	0x0c000c13	0x00008000	0x000000a9	0x000007b6	0x40000000	0x00004000
3034	sw $s0, 12($t0)	0xad10000c	0x00008000	0x000000a9	0x000003df	0x40000000	0x00004000
3038	addi $t3, $0, 1	0x200b0001	0x00008000	0x000000a9	0x000003df	0x40000000	0x00004000
303c	while: slti $t1, $s0, 1665	0x2a090681	0x00008000	0x000000a9	0x000003df	0x40000000	0x00004000
3040	beq $t1, $t3, done	0x112b0007	0x00008000	0x000000a9	0x000003df	0x40000000	0x00004000
3044	jal compute	0x0c000c13					
3048	j while	0x08000c0f					
304c	compute: divu $s2, $s0	0x0250001b	0x00008000	0x000000a9	0x000007b6	0x40000000	0x00004000
3050	mflo $t1	0x00004812	0x00008000	0x000000a9	0x000007b6	0x40000000	0x00004000
3054	add $t1, $s0, $t1	0x02094820	0x00008000	0x000000a9	0x000007b6	0x40000000	0x00004000
3058	srl $s0, $t1, 1	0x00098042	0x00008000	0x000000a9	0x000003df	0x40000000	0x00004000
305C	jr $ra	0x03e00008	0x00008000	0x000000a9	0x000003df	0x40000000	0x00004000
3060	done: sll $s0, $s0, 8	0x00108200	0x00008000	0x000000a9	0x0003df00	0x40000000	0x00004000
3064	sw $s0, 16($t0)	0xad100010	0x00008000	0x000000a9	0x0003df00	0x40000000	0x00004000
3068							
306C							

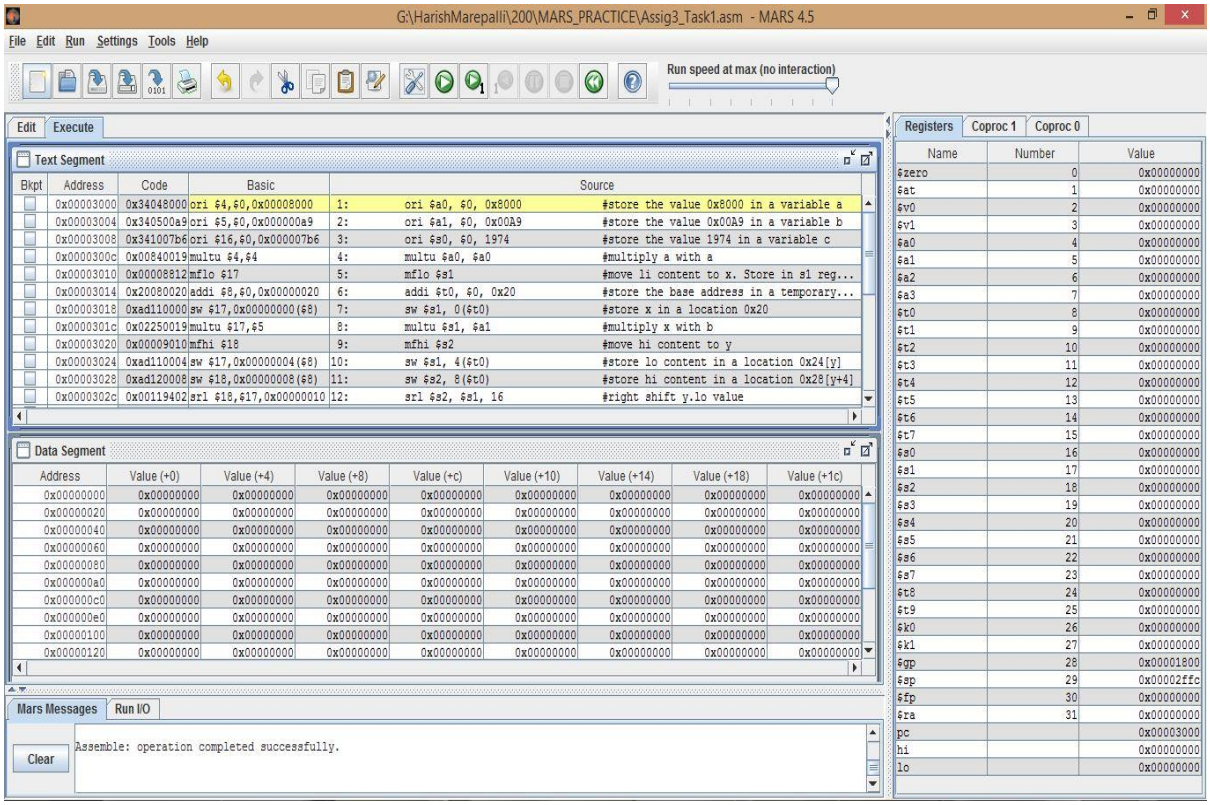
Memory contents				
Word @ 0x20	Word @ 0x24	Word @ 0x28	Word @ 0x2C	Word @ 0x30
0x40000000	0x40000000	0x0000002a	0x000003df	0x0003df00

SCREEN CAPTURES:

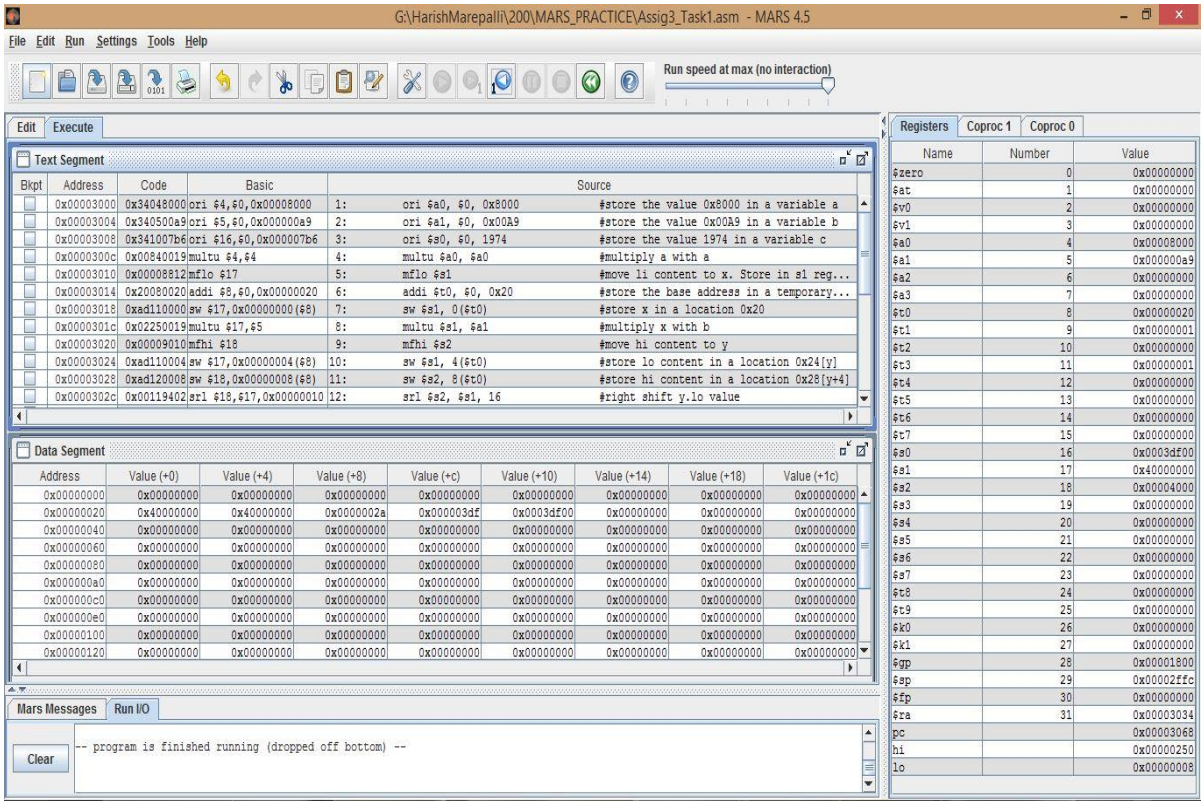
1. Screenshot of the Task1 code in the MARS editor.



2. Screenshot of the execution window after assembling and before executing the code in MARS.



3. Screenshot of the execution window after executing the code in MARS.



SOURCE CODE (Task 2):

Task2: Total number of lines are 10

```
        ori $a0, $0, 5          #line 1: store the value 5 in a variable n($a0)
        sw $a0, 0($t0)          #line 2: store the value of n in memory location at address 0x00. $t0 by
                                default is 0x00
        ori $t1, $0, 1          #line 3: store the value 1 in a variable f($t1)
while:  beq $a0, $0, done       #line 4: branch to done if n = 0 as we have to come out of the while condition
        mult $t1, $a0           #line 5: multiply f with n
        mflo $t1                #line 6: move the content from lo to f
        addi $a0, $a0, -1       #line 7: decrement the value of n by adding it with -1
        j while                 #line 8: loop back to while
done:   sw $t1, 16($t0)         #line 9: store the value of f in memory location at address 0x10
        lw $s0, 16($t0)         #line 10: load the value at address 0x10 to the register s0
```

CMPE200 Assignment 3 Task 2 Test Log
Algorithm 2

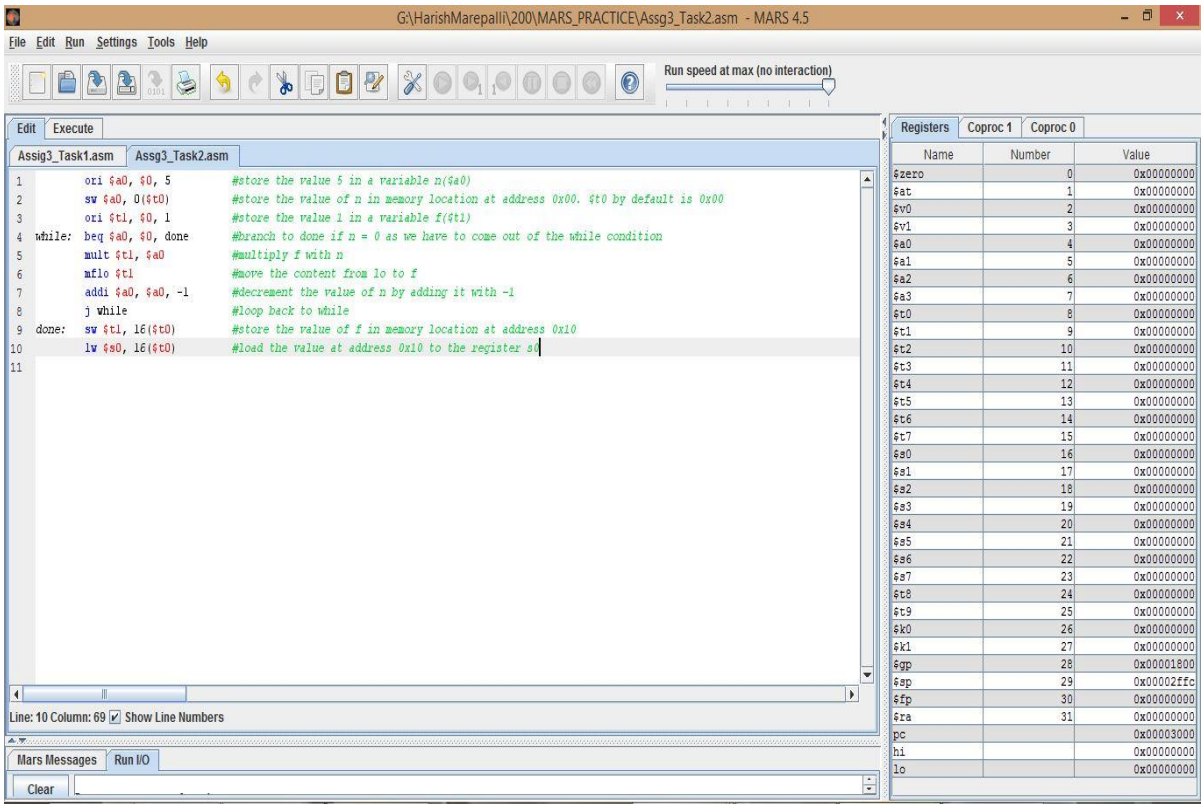
Programmer's Name: Harish Marepalli

Date: 09/24/2022

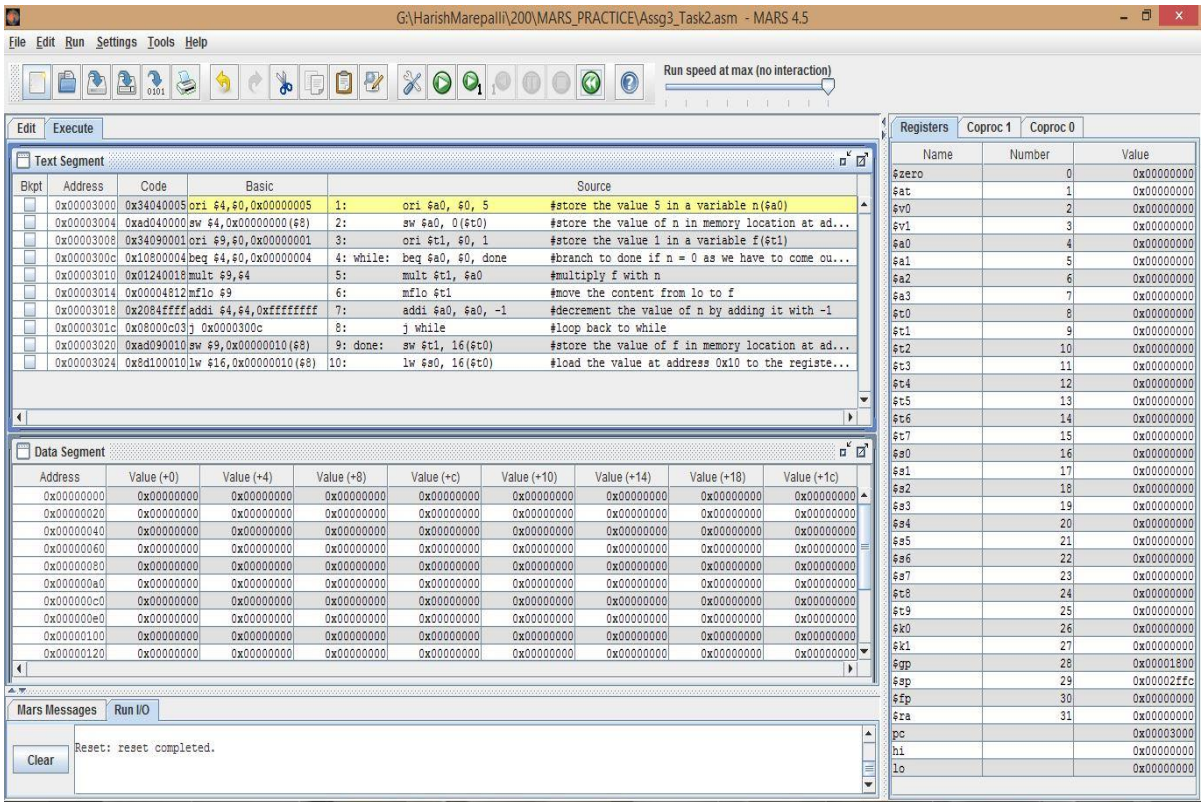
Adr	MIPS Instruction	Machine Code	Registers				Memory Content	
			$a0	**$s0**	**$t0**	**$t1**	Word @ 0x00	Word @ 0x10
3000	ori $a0, $0, 5	0x34040005	0x00000005	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
3004	sw $a0, 0($t0)	0xad040000	0x00000005	0x00000000	0x00000000	0x00000000	0x00000005	0x00000000
3008	ori $t1, $0, 1	0x34090001	0x00000005	0x00000000	0x0000000	0x00000001	0x00000005	0x00000000
300c	while: beq $a0, $0, done	0x10800004	0x00000000	0x00000000	0x00000000	0x00000078	0x00000005	0x00000000
3010	mult $t1, $a0	0x01240018	0x00000001	0x00000000	0x00000000	0x00000078	0x00000005	0x00000000
3014	mflo $t1	0x00004812	0x00000001	0x00000000	0x00000000	0x00000078	0x00000005	0x00000000
3018	addi $a0, $a0, -1	0x2084ffff	0x00000000	0x00000000	0x00000000	0x00000078	0x00000005	0x00000000
301c	j while	0x08000c03	0x00000000	0x00000000	0x00000000	0x00000078	0x00000005	0x00000000
3020	done: sw $t1, 16($t0)	0xad090010	0x00000000	0x00000000	0x00000000	0x00000078	0x00000005	0x00000078
3024	lw $s0, 16($t0)	0x8d100010	0x00000000	0x00000078	0x00000000	0x00000078	0x00000005	0x00000078
3028								

SCREEN CAPTURES:

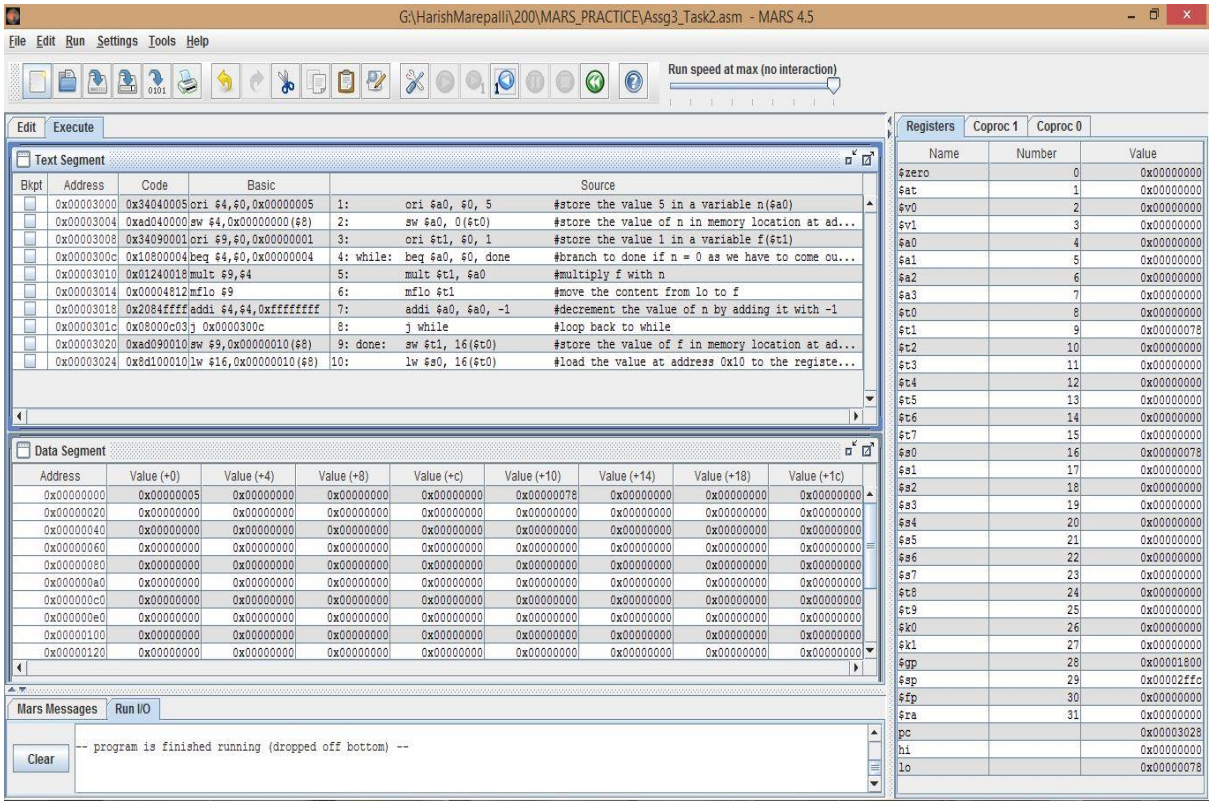
1. Screenshot of the Task2 (Factorial) code in the MARS editor.



2. Screenshot of the execution window after assembling and before executing the code in MARS.



3. Screenshot of the execution window after executing the code in MARS.



DISCUSSION SECTION

The explanation of a few instructions in the sample code with the help of the MIPS reference data card.

1. *multu rs, rt => rs*rt*
   Perform multiplication of the contents present in the two registers. A part of the result is stored in lo register and the other part is stored in hi register if the result is greater than 32 bits.

2. *divu rs, rt => rs/rt*
   Perform division of the contents present in the two registers. The quotient is stored in lo register and the remainder is stored in ho register.

3. *mflo rd => rd = lo*
   Moves the content from lo register to the rd register.

4. *mfhi rd => rd = hi*
   Moves the content from hi register to the rd register.

5. *sll rd, rt, sh => rd = rt<<sh*
   It left shifts the content in the register rt by sh times.

6. *srl rd, rt, sh => rd = rt>>sh*
   It right shifts the content in the register rt by sh times.

7. *j addr => PC = addr*
   Jumps to the specified address by placing the value in PC.

8. *jal addr => $ra = next PC value, PC = addr*
   Jump and link to the given address by putting its value in the PC as well as storing the next PC in $ra.

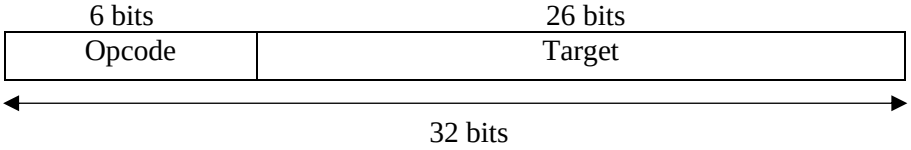
9. *jr $ra => PC = ra*
   After the completion of a subroutine, PC must go back to the next instruction after the caller method. jr is used for this.

Following are the observations related to some of the machine codes:

1. Explanation with respect to the Task1 code 19th line (j while)
   i.e., j while

   0x0000303c while: slti $t1, $s0, 1665
   …
   0x00003048 j while

   Machine code of it is written below:
   It is a j-type instruction



   Opcode for j instruction is 000010
   The address is 32 bit, but only 26 bits are available. To solve this problem, 'Jump Target Addressing' method is followed.

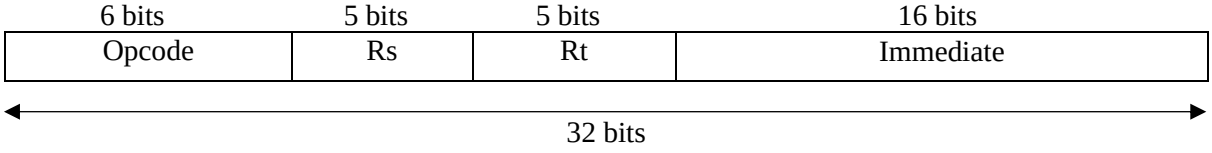
   Target address in hexadecimal => 0x0000303c
   Target address in binary format => 0000 0000 0000 0000 0011 0000 0011 1100
   Here, remove the first 4 bits and last 2 bits that are added from next PC and the shift left operation respectively to get the required 26 bits immediate.
   Immediate => 0000 0000 0000 0011 0000 0011 11

   Finally, the machine code comes as opcode + immediate
   i.e., 000010 0000 0000 0000 0011 0000 0011 11
   => 0000 1000 0000 0000 0000 1100 0000 1111
   So, the machine code in hexa decimal format is 0x08000c0f

2. Explanation with respect to the Task1 code 17th line (beq)
   i.e., beq $t1, $t3, done

   0x00003040 beq $t1, $t3, done
   …
   0x00003060 done

   Machine code of it is written below:
   It is a I-type instruction



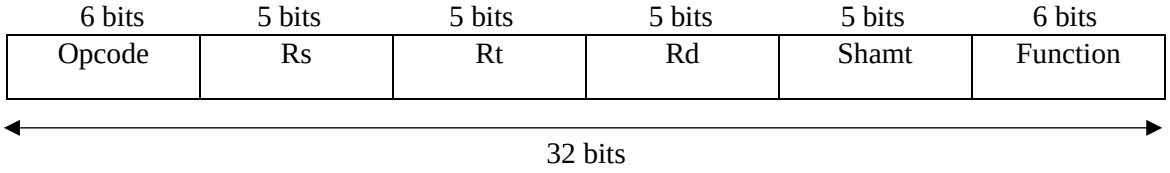
   The machine code for the above beq instruction can be found using the 16-bit immediate, which will be positive value since the branching happens downwards direction.

It is known that the branch target addressing is done using the below equation
$PC_{Target} = (PC_{beq} + 4) + 4N$ where N = immediate
Here, PC_{Target} = 0x00003060 and PC_{beq} = 0x00003040

So, 0x00003060 = (0x00003040 +4) + 4N
4N = 0x0000001c
N = 0x0x00000007 (Take only 16 bits) = 0007
The machine code is:
000100 01001 01011 0000 0000 0000 0111
0001 0001 0010 1011 0000 0000 0000 0111
So, the machine code in hexa decimal format is 0x112b0007

3. Explanation with respect to the fourth line
   i.e., multu $a0, $a0

   This is an Mult R-type instruction, and the machine code is written as:



   Opcode: 000000-011001
   Rs = $a0 = $4 = 00100
   Rt = $a0 = $4 = 00100
   Rd = 00000
   Shamt = 00000
   So, the machine code for the above is: 000000 00100 00100 00000 00000 011001
   It can be written as: 0000 0000 1000 0100 0000 0000 0001 1001
   Finally, it can be written as: 0x00840019

COLLABORATION SECTION:

1. Written MIPS assembly program for the C++ algorithm of the given two tasks in the MARS by collaborating with each other.
2. Assembled and executed the codes and observed all the operations and values.
3. Collaborated with each other to understand how the value gets stored in lo and hi register upon doing multiplication and division operations.
4. By collaborating with each other, debugged each line to know the contents of the relevant registers and recorded the memory values at certain addresses.
5. Understanding of how the machine code comes for j-type instruction was tough, but by collaborating with each other, it was understandable.

CONCLUSION:

In conclusion, by assembling, simulating, and analyzing the converted MIPS programs gained familiarity with the ISA control structures and the hi and lo registers.