**San Jose State University**
**Department of Computer Engineering**

# CMPE 200 Report

# Assignment 2 Report

**Title** MIPS Instruction Set Architecture & Programming (2)

**Semester** Fall 2022 **Date** 09/18/2022

**by**

**Name** Tirumala Saiteja Goruganthu *(typed)* **SID** 016707210 *(typed)*

## INTRODUCTION:

This activity is used to run a sample assembly code in both MARS and MIPSASM and compare the machine code of both assemblers, logging few registers' values and memory content of selected addresses.

## MY GROUP:

**Name:** Student_Team 6
**Members:** Tirumala Saiteja Goruganthu (016707210), Harish Marepalli (016707314)

## SOURCE CODE:

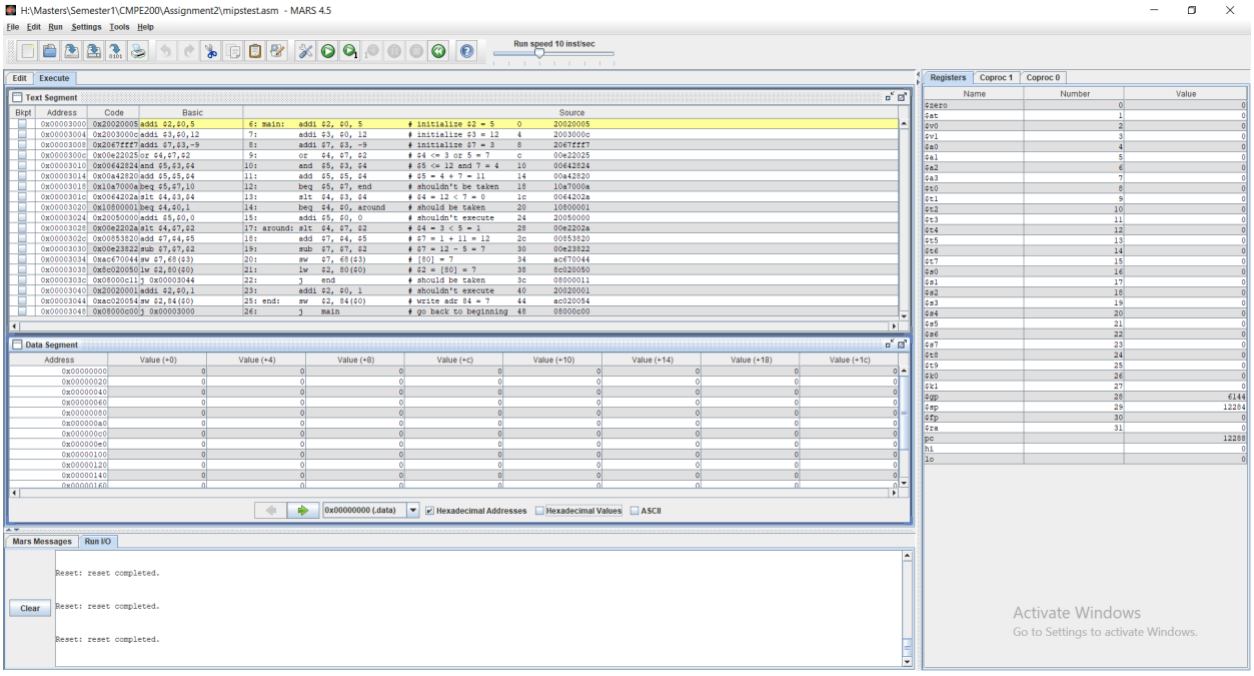
```
# mipstest.smd
# Test the following MIPS instructions.
# add, sub, and, or, slt, addi, lw, sw, beq, j

#       Assembly              Description              Address    Machine
main:   addi $2, $0, 5        # initialize $2 = 5        0        20020005
        addi $3, $0, 12       # initialize $3 = 12       4        2003000c
        addi $7, $3, -9       # initialize $7 = 3        8        2067fff7
        or   $4, $7, $2       # $4 <= 3 or 5 = 7         c        00e22025
        and  $5, $3, $4       # $5 <= 12 and 7 = 4       10       00642824
        add  $5, $5, $4       # $5 = 4 + 7 = 11          14       00a42820
        beq  $5, $7, end      # shouldn't be taken       18       10a7000a
        slt  $4, $3, $4       # $4 = 12 < 7 = 0          1c       0064202a
        beq  $4, $0, around   # should be taken          20       10800001
        addi $5, $0, 0        # shouldn't execute        24       20050000

around: slt  $4, $7, $2       # $4 = 3 < 5 = 1           28       00e2202a
        add  $7, $4, $5       # $7 = 1 + 11 = 12         2c       00853820
        sub  $7, $7, $2       # $7 = 12 - 5 = 7          30       00e23822
        sw   $7, 68($3)       # [80] = 7                 34       ac670044
        lw   $2, 80($0)       # $2 = [80] = 7            38       8c020050
        j    end              # should be taken          3c       08000011
        addi $2, $0, 1        # shouldn't execute        40       20020001

end:    sw   $2, 84($0)       # write adr 84 = 7         44       ac020054
        j    main             # go back to beginning     48       08000c00
```

## TEST LOG:

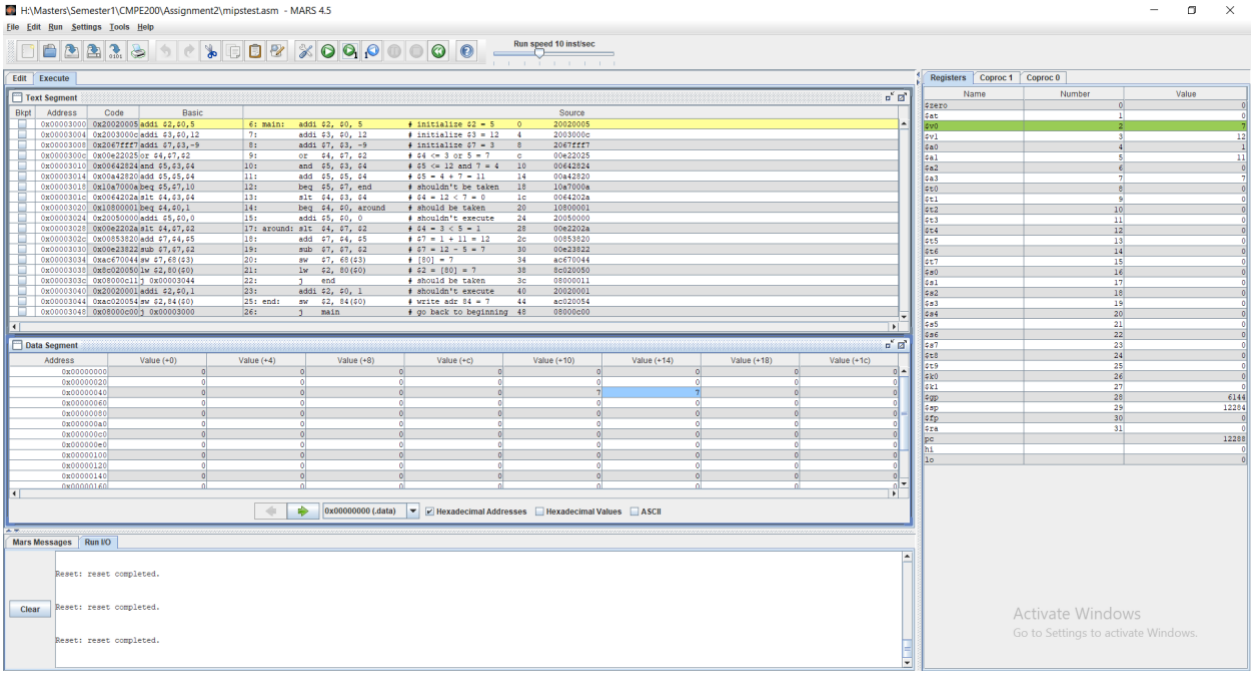
| Adr | Machine Code for MARS | Machine Code for MIPSASM | PC | Registers | | | | | Memory Content | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | $v0 | $v1 | $a0 | $a1 | $a3 | [80] | [84] |
| 3000 | 0x20020005 | 0x20020005 | 0x00003004 | 0x00000005 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 3004 | 0x2003000c | 0x2003000C | 0x00003008 | 0x00000005 | 0x0000000C | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 |
| 3008 | 0x2067fff7 | 0x2067FFF7 | 0x0000300c | 0x00000005 | 0x0000000C | 0x00000000 | 0x00000000 | 0x00000003 | 0x00000000 | 0x00000000 |
| 300c | 0x00e22025 | 0x00E22025 | 0x00003010 | 0x00000005 | 0x0000000C | 0x00000007 | 0x00000000 | 0x00000003 | 0x00000000 | 0x00000000 |
| 3010 | 0x00642824 | 0x00642824 | 0x00003014 | 0x00000005 | 0x0000000C | 0x00000007 | 0x00000004 | 0x00000003 | 0x00000000 | 0x00000000 |
| 3014 | 0x00a42820 | 0x00A42820 | 0x00003018 | 0x00000005 | 0x0000000C | 0x00000007 | 0x0000000B | 0x00000003 | 0x00000000 | 0x00000000 |
| 3018 | 0x10a7000a | 0x10E5000A | 0x0000301c | 0x00000005 | 0x0000000C | 0x00000007 | 0x0000000B | 0x00000003 | 0x00000000 | 0x00000000 |
| 301c | 0x0064202a | 0x0064202A | 0x00003020 | 0x00000005 | 0x0000000C | 0x00000000 | 0x0000000B | 0x00000003 | 0x00000000 | 0x00000000 |
| 3020 | 0x10800001 | 0x10040001 | 0x00003028 | 0x00000005 | 0x0000000C | 0x00000000 | 0x0000000B | 0x00000003 | 0x00000000 | 0x00000000 |
| 3024 | 0x20050000 | 0x20050000 | | | | | | | | |
| 3028 | 0x00e2202a | 0x00E2202A | 0x0000302c | 0x00000005 | 0x0000000C | 0x00000001 | 0x0000000B | 0x00000003 | 0x00000000 | 0x00000000 |
| 302c | 0x00853820 | 0x00853820 | 0x00003030 | 0x00000005 | 0x0000000C | 0x00000001 | 0x0000000B | 0x0000000C | 0x00000000 | 0x00000000 |
| 3030 | 0x00e23822 | 0x00E23822 | 0x00003034 | 0x00000005 | 0x0000000C | 0x00000001 | 0x0000000B | 0x00000007 | 0x00000000 | 0x00000000 |
| 3034 | 0xac670044 | 0xAC670044 | 0x00003038 | 0x00000005 | 0x0000000C | 0x00000001 | 0x0000000B | 0x00000007 | 0x00000007 | 0x00000000 |
| 3038 | 0x8c020050 | 0x8C020050 | 0x0000303c | 0x00000007 | 0x0000000C | 0x00000001 | 0x0000000B | 0x00000007 | 0x00000007 | 0x00000000 |
| 303c | 0x08000c11 | 0x08000011 | 0x00003044 | 0x00000007 | 0x0000000C | 0x00000001 | 0x0000000B | 0x00000007 | 0x00000007 | 0x00000000 |
| 3040 | 0x20020001 | 0x20020001 | | | | | | | | |
| 3044 | 0xac020054 | 0xAC020054 | 0x00003048 | 0x00000007 | 0x0000000C | 0x00000001 | 0x0000000B | 0x00000007 | 0x00000007 | 0x00000007 |
| 3048 | 0x08000c00 | 0x08000000 | 0x00003000 | 0x00000007 | 0x0000000C | 0x00000001 | 0x0000000B | 0x00000007 | 0x00000007 | 0x00000007 |

## SCREEN CAPTURES:

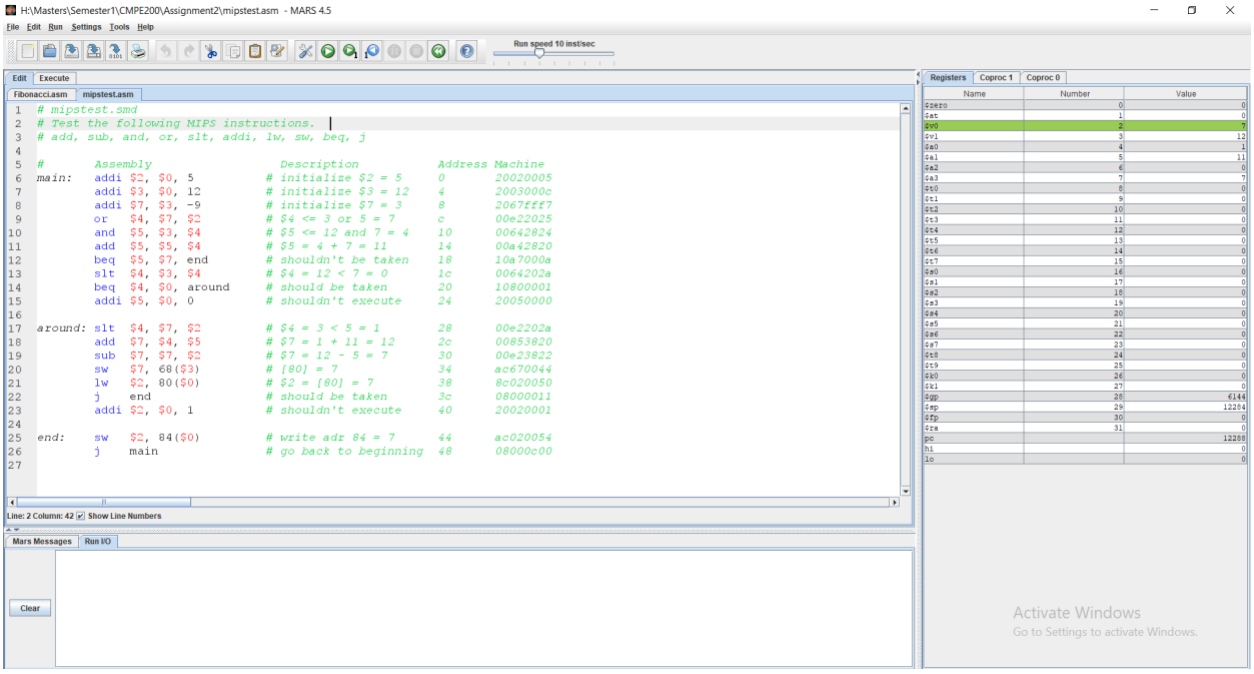
1. Snippet at initial execution of assembly program in MARS assembler.



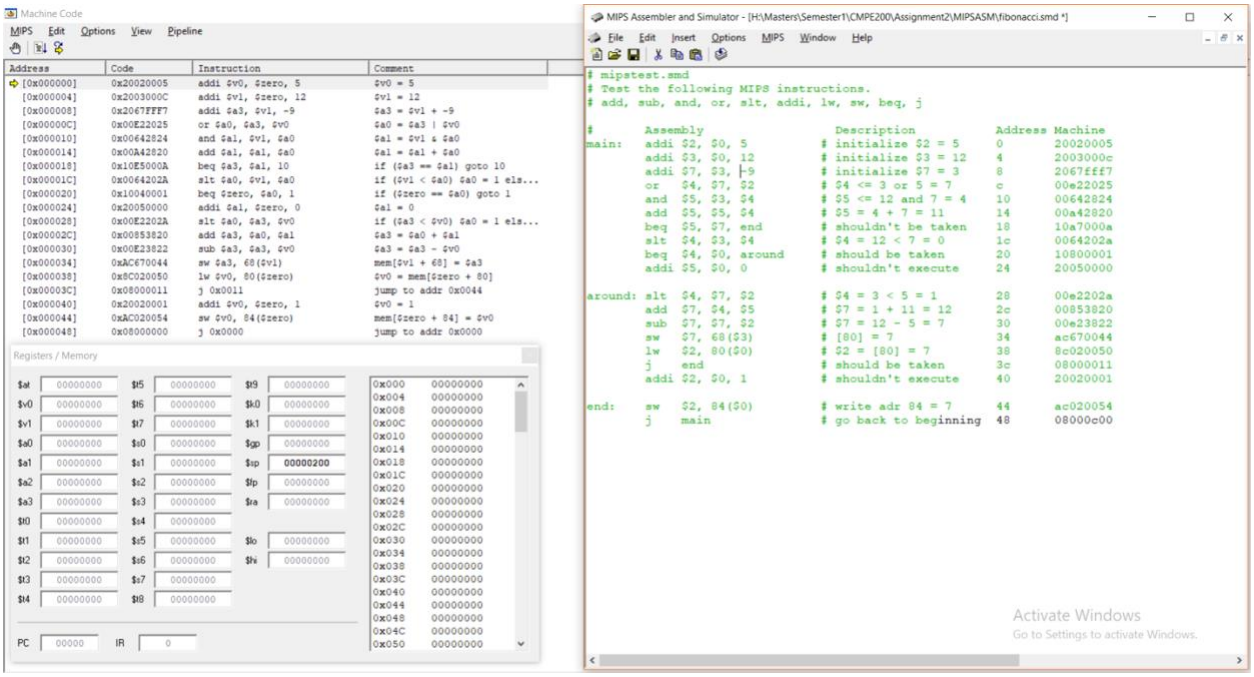
2. Snippet after the execution of the whole assembly program in MARS.



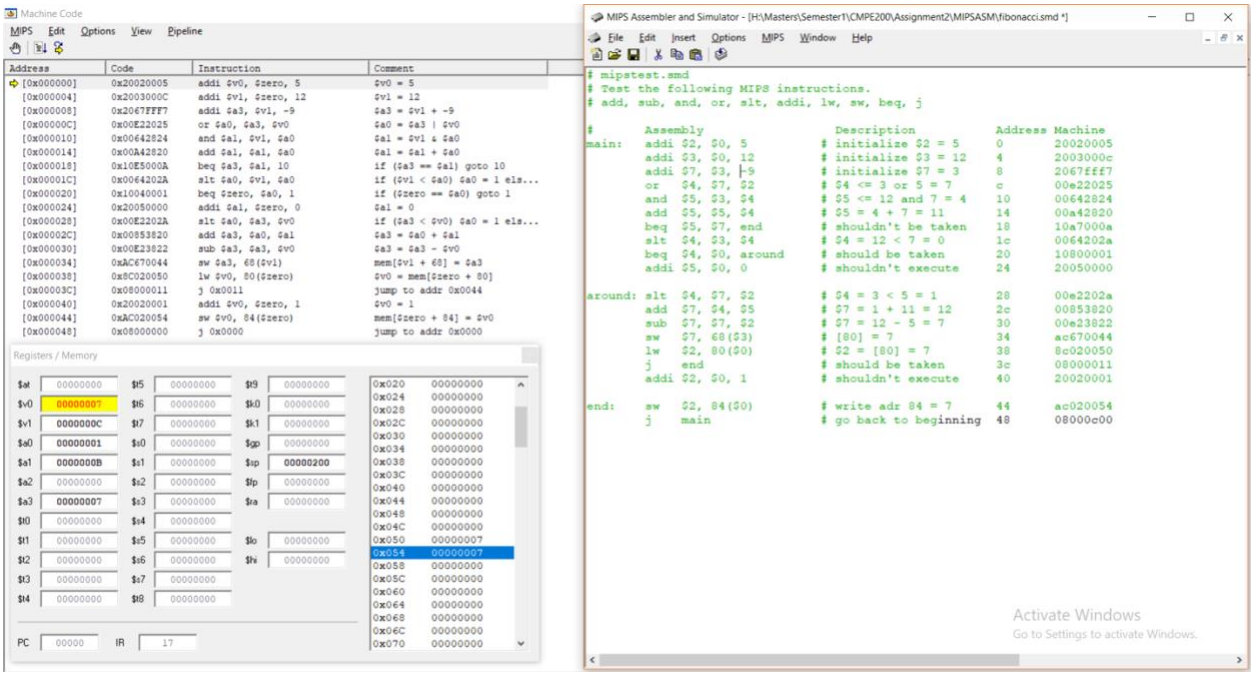

3. Snippet of source code and register values in MARS




4. Snippet at initial execution of the assembly code in MIPSASM assembler.




5. Snippet after the execution of the whole assembly program in MIPSASM.




## DISCUSSION SECTION:

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

1. *addi rt, rs, imm => rt = rs + imm;*
   Add Immediate value to the source register content.

2. *or rd, rs, rt => rd = rs | rt;*
   Perform OR operation between the contents of the register rs, rt and putting the result in register rd.

3. *and rd, rs, rt => rd = rs & rt;*
   Perform AND operation between the contents of the register rs, rt and putting the result in register rd.

4. *add rd, rs, rt => rd = rs + rt;*
   Perform ADD operation between the contents of the register rs, rt and putting the result in register rd.

5. *beq rt, rs, Target => if(rt == rs) branch to target;*
   If the contents of registers rt and rs are equal then the program counter will branch to the target.

6. *slt rd, rs, rt => if(rs < rt) rd = 1; else rd = 0;*
   If the content of register rs is less than rt then rd will become 1 else it becomes 0.

7. *sub rd, rs, rt => rd = rs – rt;*
   Perform SUB operation between the contents of the register rs, rt and putting the result in register rd.

8. *sw rt, imm(rs) => Memory[rs + imm] = rt;*
   Stores the content of rt in memory using the location provided by the combination of register rs and the offset value (imm).

9. *lw rt, imm(rs) => rt = Memory[rs + imm];*
   Loads the content of memory in the location provided by the combination of register rs and the offset value (imm) and puts the result in register rt.

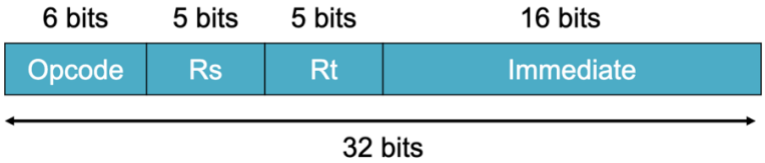
10. *j addr => PC = addr;*
    Jumps to the given target address by assigning this value to the program counter (PC).

Following are the observations with respect to the machine codes:

1. Let us take the first instruction in the above source code.

   addi $2, $0, 5

   This is an I-type instruction which means the machine code is written in the below format

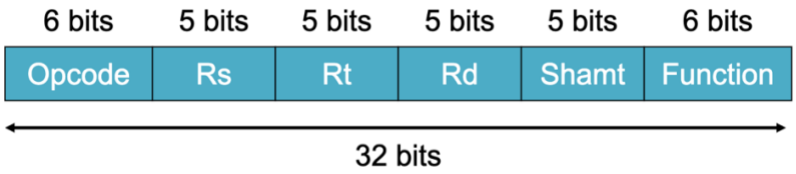



   For addi, the opcode will be 001000
   For Rs=$0, the bits will be 00000
   For Rt=$2, the bits will be 00010
   For immediate = 5, the bits will be 0000000000000101
   So, the machine code in binary is 0010 0000 0000 0010 0000 0000 0000 0101
   And in hexadecimal => 0x20020005

2. Let us take an R-type instruction in the above source code.

   add $5, $5, $4

The machine code for R-type instructions is written in below format




For all R-type instructions, the opcode will be 0.
For Rs=$5, the bits will be 00101
For Rt=$4, the bits will be 00100
For Rd=$5, the bits will be 00101
For non-shift type instructions, the shamt bits will be 0
For add, the bits for the function part will be 0x20 => 100000

So, the machine code for this instruction in binary is 0000 0000 1010 0100 0010 1000 0010 0000 and in hexadecimal => 0x00A42820

Following are the observations in the test log:

1. The machine codes for MARS and MIPSASM assemblers for the same assembly program is similar for the most part except for the lines with addresses 0x00003018, 0x00003020, 0x0000303c and 0x00003048. The different machine codes for these addresses are given in the table below.

| Address | Machine Code in MARS | Machine Code in MIPSASM |
|---|---|---|
| 0x00003018 | 0x10a7000a | 0x10E5000A |
| 0x00003020 | 0x10800001 | 0x10040001 |
| 0x0000303c | 0x08000c11 | 0x08000011 |
| 0x00003048 | 0x08000c00 | 0x08000000 |

2. The program counter value will never be 0x00003024 and 0x00003040 because of the ‘beq’ instruction, these lines will not get executed during run-time.

## COLLABORATION SECTION:

1. Installed and setup MARS and MIPSASM by collaborating with each other.
2. Assembled the given MIPS assembly code in the MARS by collaborating with each other.
3. Executed the code and observed all the operations and values.
4. Collaborated with each other to compare the machine code generated by two different assemblers.
5. In addition to the mipstest.asm, modified and run the Fibonacci code in MARS by collaborating and discussing the operations done.
6. By collaborating with each other, debugged each line to know the contents of the relevant registers and recorded the memory values at certain addresses.

## CONCLUSION:

In conclusion, the machine codes of MARS and MIPSASM for all the addresses cannot be the same. By assembling, simulating, and analyzing the given sample program gained familiarity with the MIPS instruction set.