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:

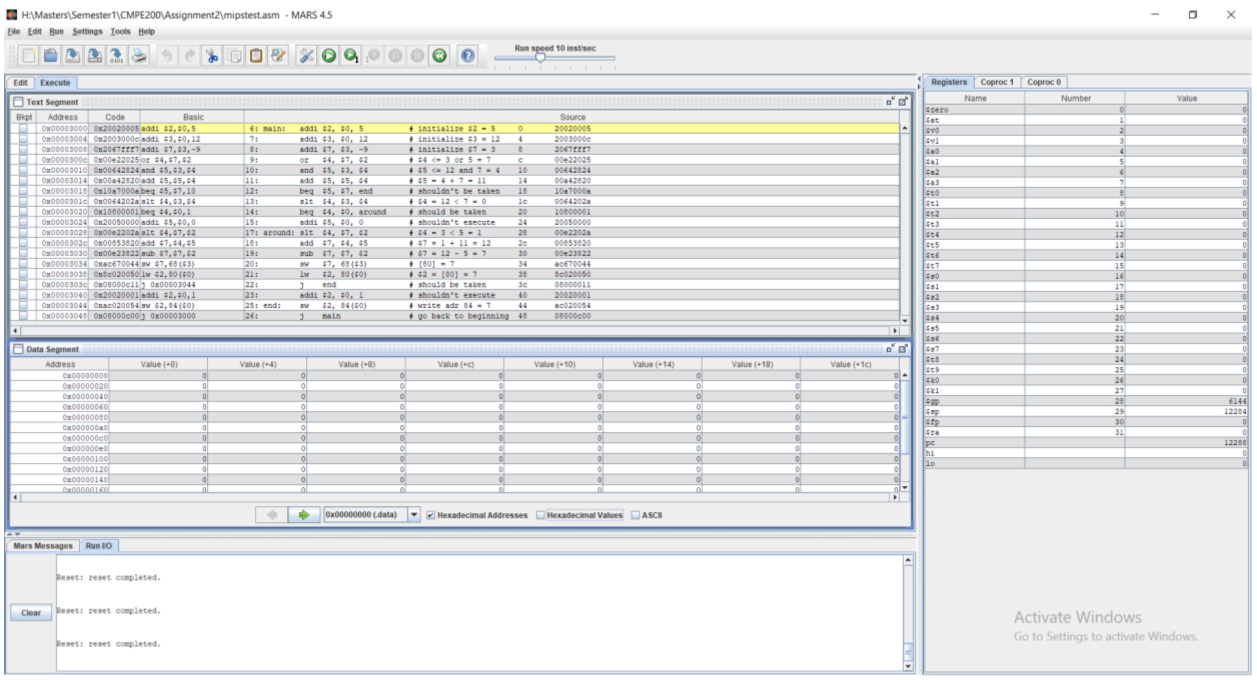
```
# mipstest.smd
# Test the following MIPS instructions.
# add, sub, and, or, slt, addi, lw, sw, beq, j

#       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.



H:\Masters\Semester1\CMPE200\Assignment2\mipstest.asm - MARS 4.5

Run speed 10 inst/sec

Text Segment

```
Bkpt Address    Code       Basic                                                           Source
     0x00003000 0x20020005 addi $2,$0,5     6: main:   addi $2, $0, 5      # initialize $2 = 5      0  20020005
     0x00003004 0x2003000c addi $3,$0,12    7:         addi $3, $0, 12     # initialize $3 = 12     4  2003000c
     0x00003008 0x2067fff7 addi $7,$3,-9    8:         addi $7, $3, -9     # initialize $7 = 3      8  2067fff7
     0x0000300c 0x00e22025 or $4,$7,$2      9:         or   $4, $7, $2     # $4 <= 3 or 5 = 7       c  00e22025
     0x00003010 0x00642824 and $5,$3,$4     10:        and  $5, $3, $4     # $5 <= 12 and 7 = 4     10 00642824
     0x00003014 0x00a42820 add $5,$5,$4     11:        add  $5, $5, $4     # $5 = 4 + 7 = 11        14 00a42820
     0x00003018 0x10a7000a beq $5,$7,10     12:        beq  $5, $7, end    # shouldn't be taken     18 10a7000a
     0x0000301c 0x0064202a slt $4,$3,$4     13:        slt  $4, $3, $4     # $4 = 12 < 7 = 0        1c 0064202a
     0x00003020 0x10800001 beq $4,$0,1      14:        beq  $4, $0, around # should be taken        20 10800001
     0x00003024 0x20050000 addi $5,$0,0     15:        addi $5, $0, 0      # shouldn't execute      24 20050000
     0x00003028 0x00e2202a slt $4,$7,$2     17: around: slt $4, $7, $2     # $4 = 3 < 5 = 1         28 00e2202a
     0x0000302c 0x00853820 add $7,$4,$5     18:        add  $7, $4, $5     # $7 = 1 + 11 = 12       2c 00853820
     0x00003030 0x00e23822 sub $7,$7,$2     19:        sub  $7, $7, $2     # $7 = 12 - 5 = 7        30 00e23822
     0x00003034 0xac670044 sw $7,68($3)     20:        sw   $7, 68($3)     # [80] = 7               34 ac670044
     0x00003038 0x8c020050 lw $2,80($0)     21:        lw   $2, 80($0)     # $2 = [80] = 7          38 8c020050
     0x0000303c 0x08000c11 j 0x00003044     22:        j    end            # should be taken        3c 08000011
     0x00003040 0x20020001 addi $2,$0,1     23:        addi $2, $0, 1      # shouldn't execute      40 20020001
     0x00003044 0xac020054 sw $2,84($0)     25: end:   sw   $2, 84($0)     # write adr 84 = 7       44 ac020054
     0x00003048 0x08000c00 j 0x00003000     26:        j    main           # go back to beginning   48 08000c00
```

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x00000000	0	0	0	0	0	0	0	0
0x00000020	0	0	0	0	0	0	0	0
0x00000040	0	0	0	0	0	0	0	0
0x00000060	0	0	0	0	0	0	0	0
0x00000080	0	0	0	0	0	0	0	0
0x000000a0	0	0	0	0	0	0	0	0
0x000000c0	0	0	0	0	0	0	0	0
0x000000e0	0	0	0	0	0	0	0	0
0x00000100	0	0	0	0	0	0	0	0
0x00000120	0	0	0	0	0	0	0	0
0x00000140	0	0	0	0	0	0	0	0
0x00000160	0	0	0	0	0	0	0	0

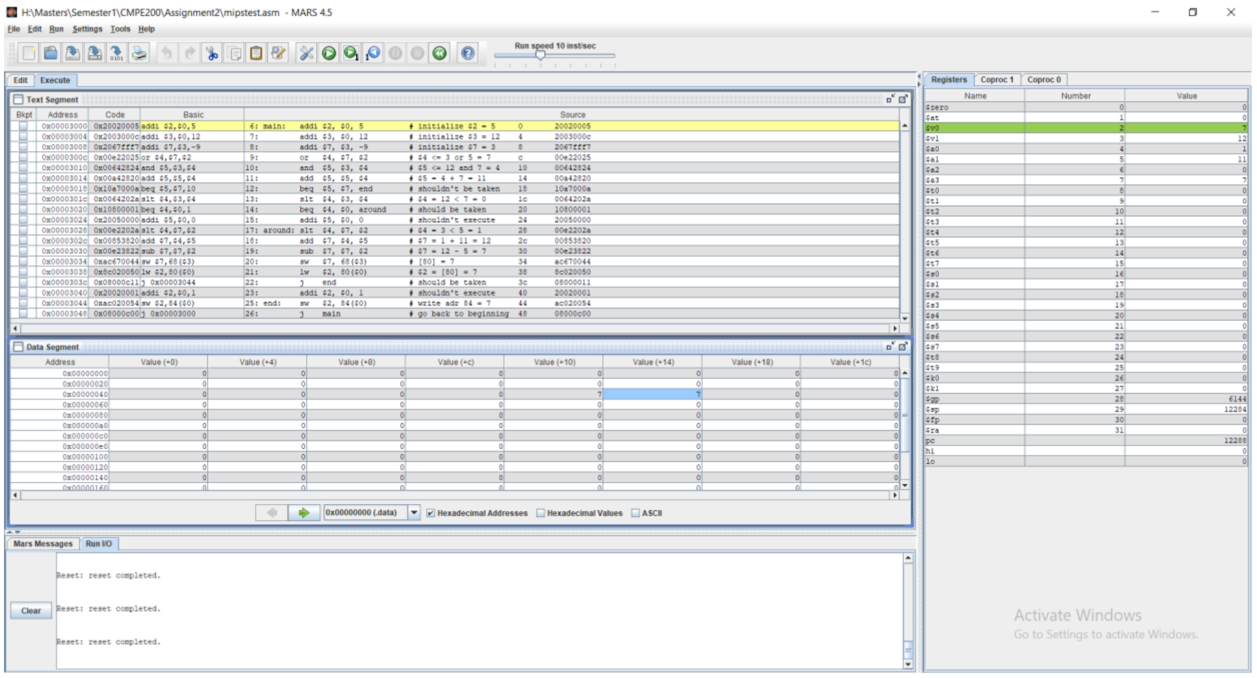
0x00000000 (.data) ☑ Hexadecimal Addresses ☐ Hexadecimal Values ☐ ASCII

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	0
$t1	9	0
$t2	10	0
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	6144
$sp	29	12284
$fp	30	0
$ra	31	0
pc		12288
hi		0
lo		0

```
Reset: reset completed.

Reset: reset completed.

Reset: reset completed.
```

Activate Windows
Go to Settings to activate Windows.

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



H:\Masters\Semester1\CMPE200\Assignment2\mipstest.asm - MARS 4.5

Run speed 10 inst/sec

Text Segment

Bkpt	Address	Code	Basic		Source
☐	0x00003000	0x20020005	addi $2,$0,5	6: main: addi $2, $0, 5 # initialize $2 = 5	0 20020005
☐	0x00003004	0x2003000c	addi $3,$0,12	7: addi $3, $0, 12 # initialize $3 = 12	4 2003000c
☐	0x00003008	0x2067fff7	addi $7,$3,-9	8: addi $7, $3, -9 # initialize $7 = 3	8 2067fff7
☐	0x0000300c	0x00e22025	or $4,$7,$2	9: or $4, $7, $2 # $4 <= 3 or 5 = 7	c 00e22025
☐	0x00003010	0x00642824	and $5,$3,$4	10: and $5, $3, $4 # $5 <= 12 and 7 = 4	10 00642824
☐	0x00003014	0x00a42820	add $5,$5,$4	11: add $5, $5, $4 # $5 = 4 + 7 = 11	14 00a42820
☐	0x00003018	0x10a7000a	beq $5,$7,10	12: beq $5, $7, end # shouldn't be taken	18 10a7000a
☐	0x0000301c	0x0064202a	slt $4,$3,$4	13: slt $4, $3, $4 # $4 = 12 < 7 = 0	1c 0064202a
☐	0x00003020	0x10800001	beq $4,$0,1	14: beq $4, $0, around # should be taken	20 10800001
☐	0x00003024	0x20050000	addi $5,$0,0	15: addi $5, $0, 0 # shouldn't execute	24 20050000
☐	0x00003028	0x00e2202a	slt $4,$7,$2	17: around: slt $4, $7, $2 # $4 = 3 < 5 = 1	28 00e2202a
☐	0x0000302c	0x00853820	add $7,$4,$5	18: add $7, $4, $5 # $7 = 1 + 11 = 12	2c 00853820
☐	0x00003030	0x00e23822	sub $7,$7,$2	19: sub $7, $7, $2 # $7 = 12 - 5 = 7	30 00e23822
☐	0x00003034	0xac670044	sw $7,68($3)	20: sw $7, 68($3) # [80] = 7	34 ac670044
☐	0x00003038	0x8c020050	lw $2,80($0)	21: lw $2, 80($0) # $2 = [80] = 7	38 8c020050
☐	0x0000303c	0x08000c11	j 0x00003044	22: j end # should be taken	3c 08000011
☐	0x00003040	0x20020001	addi $2,$0,1	23: addi $2, $0, 1 # shouldn't execute	40 20020001
☐	0x00003044	0xac020054	sw $2,84($0)	25: end: sw $2, 84($0) # write adr 84 = 7	44 ac020054
☐	0x00003048	0x08000c00	j 0x00003000	26: j main # go back to beginning	48 08000c00

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x00000000	0	0	0	0	0	0	0	0
0x00000020	0	0	0	0	0	0	0	0
0x00000040	0	0	0	0	7	7	0	0
0x00000060	0	0	0	0	0	0	0	0
0x00000080	0	0	0	0	0	0	0	0
0x000000a0	0	0	0	0	0	0	0	0
0x000000c0	0	0	0	0	0	0	0	0
0x000000e0	0	0	0	0	0	0	0	0
0x00000100	0	0	0	0	0	0	0	0
0x00000120	0	0	0	0	0	0	0	0
0x00000140	0	0	0	0	0	0	0	0
0x00000160	0	0	0	0	0	0	0	0

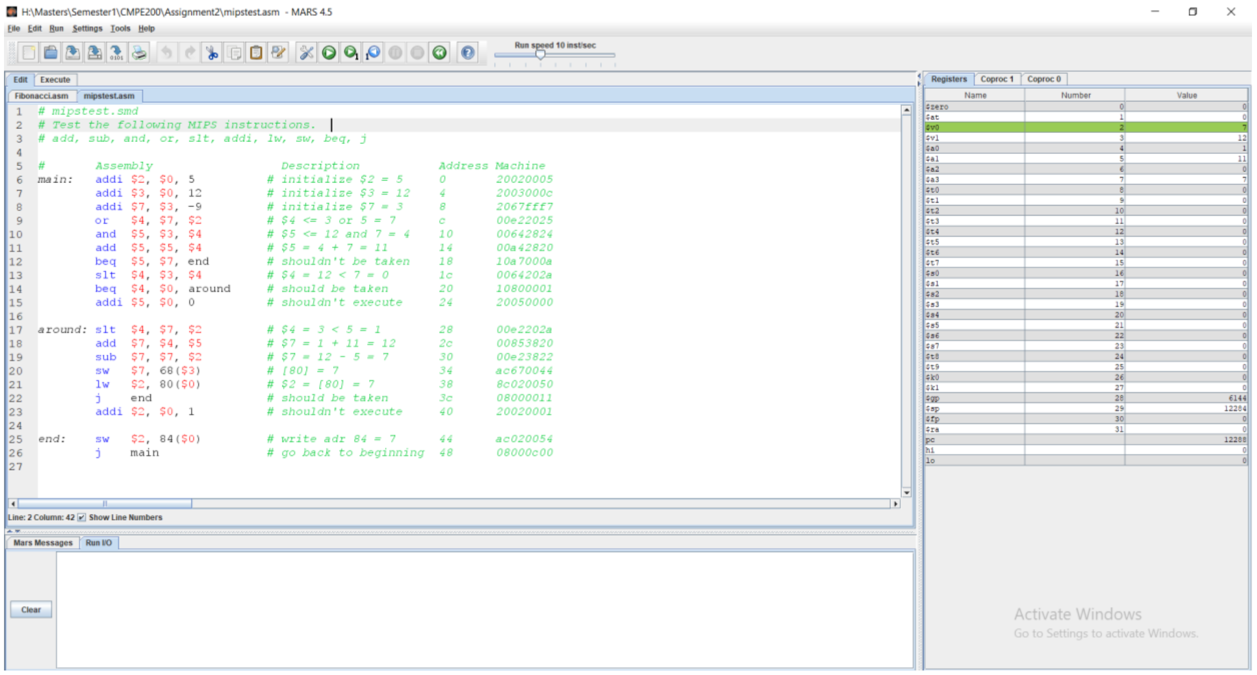
0x00000000 (.data) ☑ Hexadecimal Addresses ☐ Hexadecimal Values ☐ ASCII

Mars Messages

```
Reset: reset completed.

Reset: reset completed.

Reset: reset completed.
```

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	7
$v1	3	12
$a0	4	1
$a1	5	11
$a2	6	0
$a3	7	7
$t0	8	0
$t1	9	0
$t2	10	0
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	6144
$sp	29	12284
$fp	30	0
$ra	31	0
pc		12288
hi		0
lo		0

3. Snippet of source code and register values in MARS



H:\Masters\Semester1\CMPE200\Assignment2\mipstest.asm - MARS 4.5

Run speed 10 inst/sec

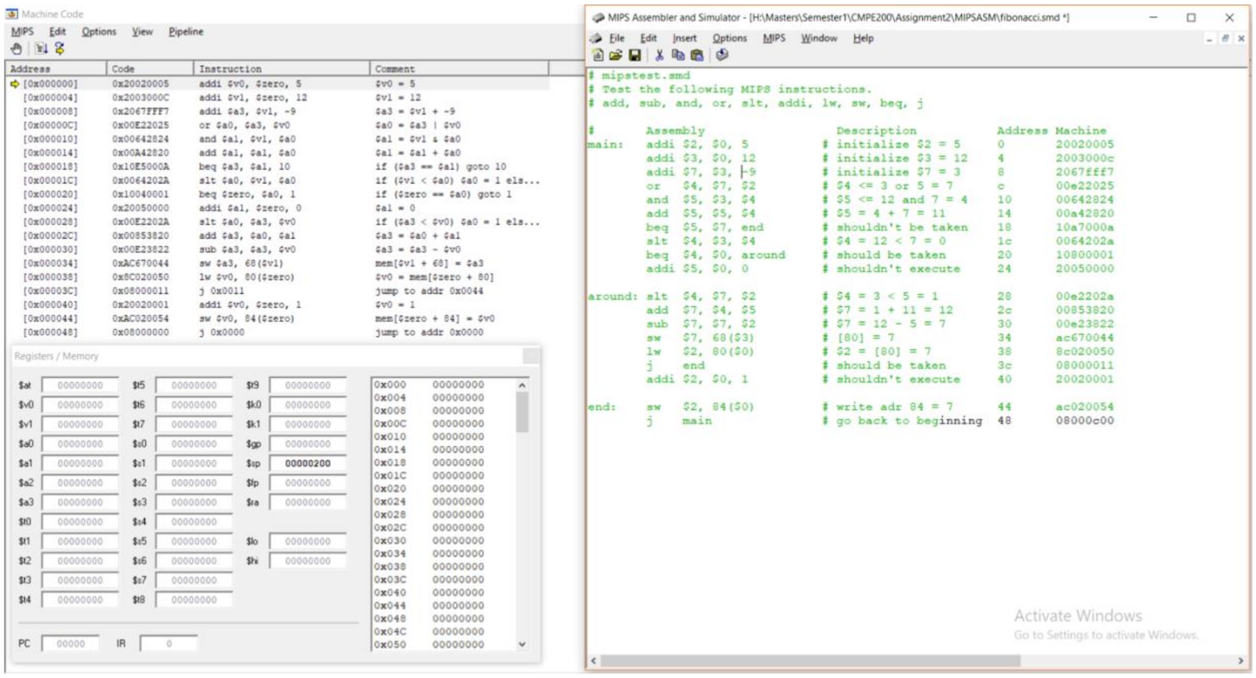
Fibonacci.asm mipstest.asm

```
# 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
```

Line: 2 Column: 42 ☑ Show Line Numbers

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	7
$v1	3	12
$a0	4	1
$a1	5	11
$a2	6	0
$a3	7	7
$t0	8	0
$t1	9	0
$t2	10	0
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	6144
$sp	29	12284
$fp	30	0
$ra	31	0
pc		12288
hi		0
lo		0

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



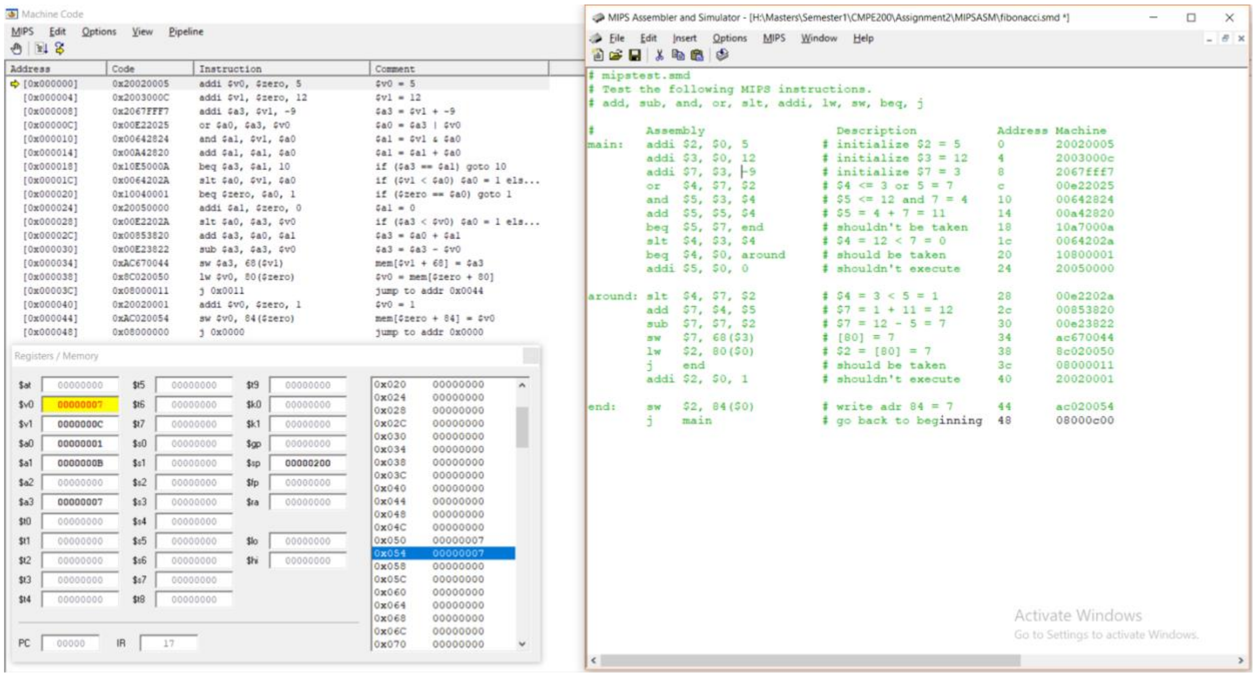
```
Address       Code        Instruction              Comment
[0x000000]    0x20020005  addi $v0, $zero, 5       $v0 = 5
[0x000004]    0x2003000C  addi $v1, $zero, 12      $v1 = 12
[0x000008]    0x2067FFF7  addi $a3, $v1, -9        $a3 = $v1 + -9
[0x00000C]    0x00E22025  or $a0, $a3, $v0         $a0 = $a3 | $v0
[0x000010]    0x00642824  and $a1, $v1, $a0        $a1 = $v1 & $a0
[0x000014]    0x00A42820  add $a1, $a1, $a0        $a1 = $a1 + $a0
[0x000018]    0x10E5000A  beq $a3, $a1, 10         if ($a3 == $a1) goto 10
[0x00001C]    0x0064202A  slt $a0, $v1, $a0        if ($v1 < $a0) $a0 = 1 els...
[0x000020]    0x10040001  beq $zero, $a0, 1        if ($zero == $a0) goto 1
[0x000024]    0x20050000  addi $a1, $zero, 0       $a1 = 0
[0x000028]    0x00E2202A  slt $a0, $a3, $v0        if ($a3 < $v0) $a0 = 1 els...
[0x00002C]    0x00853820  add $a3, $a0, $a1        $a3 = $a0 + $a1
[0x000030]    0x00E23822  sub $a3, $a3, $v0        $a3 = $a3 - $v0
[0x000034]    0xAC670044  sw $a3, 68($v1)          mem[$v1 + 68] = $a3
[0x000038]    0x8C020050  lw $v0, 80($zero)        $v0 = mem[$zero + 80]
[0x00003C]    0x08000011  j 0x0011                 jump to addr 0x0044
[0x000040]    0x20020001  addi $v0, $zero, 1       $v0 = 1
[0x000044]    0xAC020054  sw $v0, 84($zero)        mem[$zero + 84] = $v0
[0x000048]    0x08000000  j 0x0000                 jump to addr 0x0000
```

```
$at  00000000   $t5  00000000   $t9  00000000     0x000  00000000
$v0  00000000   $t6  00000000   $k0  00000000     0x004  00000000
$v1  00000000   $t7  00000000   $k1  00000000     0x008  00000000
$a0  00000000   $s0  00000000   $gp  00000000     0x00C  00000000
$a1  00000000   $s1  00000000   $sp  00000200     0x010  00000000
$a2  00000000   $s2  00000000   $fp  00000000     0x014  00000000
$a3  00000000   $s3  00000000   $ra  00000000     0x018  00000000
$t0  00000000   $s4  00000000                     0x01C  00000000
$t1  00000000   $s5  00000000   $lo  00000000     0x020  00000000
$t2  00000000   $s6  00000000   $hi  00000000     0x024  00000000
$t3  00000000   $s7  00000000                     0x028  00000000
$t4  00000000   $t8  00000000                     0x02C  00000000
                                                  0x030  00000000
                                                  0x034  00000000
                                                  0x038  00000000
                                                  0x03C  00000000
                                                  0x040  00000000
                                                  0x044  00000000
                                                  0x048  00000000
                                                  0x04C  00000000
                                                  0x050  00000000
PC  00000    IR  0
```

MIPS Assembler and Simulator - [H:\Masters\Semester1\CMPE200\Assignment2\MIPSASM\fibonacci.smd *]

```
# 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
```

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



```
Address       Code        Instruction              Comment
[0x000000]    0x20020005  addi $v0, $zero, 5       $v0 = 5
[0x000004]    0x2003000C  addi $v1, $zero, 12      $v1 = 12
[0x000008]    0x2067FFF7  addi $a3, $v1, -9        $a3 = $v1 + -9
[0x00000C]    0x00E22025  or $a0, $a3, $v0         $a0 = $a3 | $v0
[0x000010]    0x00642824  and $a1, $v1, $a0        $a1 = $v1 & $a0
[0x000014]    0x00A42820  add $a1, $a1, $a0        $a1 = $a1 + $a0
[0x000018]    0x10E5000A  beq $a3, $a1, 10         if ($a3 == $a1) goto 10
[0x00001C]    0x0064202A  slt $a0, $v1, $a0        if ($v1 < $a0) $a0 = 1 els...
[0x000020]    0x10040001  beq $zero, $a0, 1        if ($zero == $a0) goto 1
[0x000024]    0x20050000  addi $a1, $zero, 0       $a1 = 0
[0x000028]    0x00E2202A  slt $a0, $a3, $v0        if ($a3 < $v0) $a0 = 1 els...
[0x00002C]    0x00853820  add $a3, $a0, $a1        $a3 = $a0 + $a1
[0x000030]    0x00E23822  sub $a3, $a3, $v0        $a3 = $a3 - $v0
[0x000034]    0xAC670044  sw $a3, 68($v1)          mem[$v1 + 68] = $a3
[0x000038]    0x8C020050  lw $v0, 80($zero)        $v0 = mem[$zero + 80]
[0x00003C]    0x08000011  j 0x0011                 jump to addr 0x0044
[0x000040]    0x20020001  addi $v0, $zero, 1       $v0 = 1
[0x000044]    0xAC020054  sw $v0, 84($zero)        mem[$zero + 84] = $v0
[0x000048]    0x08000000  j 0x0000                 jump to addr 0x0000
```

```
$at  00000000   $t5  00000000   $t9  00000000     0x020  00000000
$v0  00000007   $t6  00000000   $k0  00000000     0x024  00000000
$v1  0000000C   $t7  00000000   $k1  00000000     0x028  00000000
$a0  00000001   $s0  00000000   $gp  00000000     0x02C  00000000
$a1  0000000B   $s1  00000000   $sp  00000200     0x030  00000000
$a2  00000000   $s2  00000000   $fp  00000000     0x034  00000000
$a3  00000007   $s3  00000000   $ra  00000000     0x038  00000000
$t0  00000000   $s4  00000000                     0x03C  00000000
$t1  00000000   $s5  00000000   $lo  00000000     0x040  00000000
$t2  00000000   $s6  00000000   $hi  00000000     0x044  00000000
$t3  00000000   $s7  00000000                     0x048  00000000
$t4  00000000   $t8  00000000                     0x04C  00000000
                                                  0x050  00000007
                                                  0x054  00000007
                                                  0x058  00000000
                                                  0x05C  00000000
                                                  0x060  00000000
                                                  0x064  00000000
                                                  0x068  00000000
                                                  0x06C  00000000
                                                  0x070  00000000
PC  00000    IR  17
```

MIPS Assembler and Simulator - [H:\Masters\Semester1\CMPE200\Assignment2\MIPSASM\fibonacci.smd *]

```
# 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
```

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.

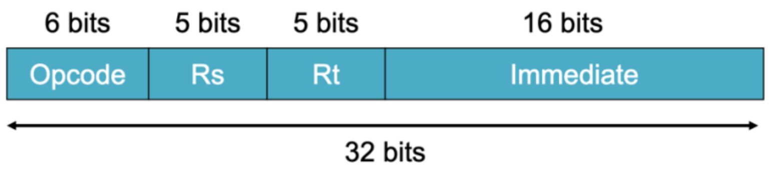
10. *j addr => PC = addr;*
    Jumps to the given target address by assigning this value to the program counter (PC).

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

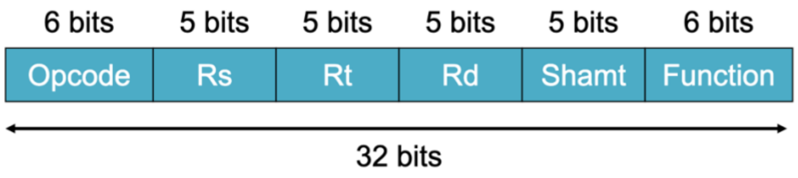


   For addi, the opcode will be 001000
   For Rs=$0, the bits will be 00000
   For Rt=$2, the bits will be 00010
   For immediate = 5, the bits will be 0000000000000101
   So, the machine code in binary is 0010 0000 0000 0010 0000 0000 0000 0101
   And in hexadecimal => 0x20020005

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.