

**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*

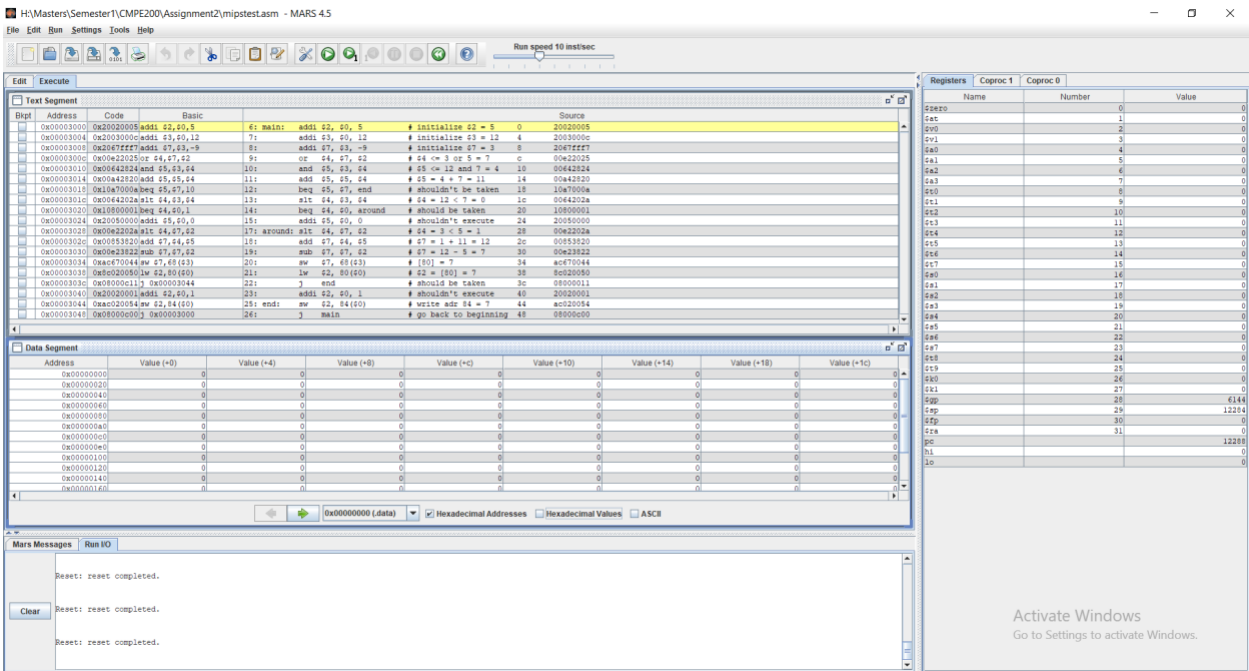
#	Assembly	Description	Address	Machine
main:	<i>addi \$2, \$0, 5</i>	<i># initialize \$2 = 5</i>	<i>0</i>	<i>20020005</i>
	<i>addi \$3, \$0, 12</i>	<i># initialize \$3 = 12</i>	<i>4</i>	<i>2003000c</i>
	<i>addi \$7, \$3, -9</i>	<i># initialize \$7 = 3</i>	<i>8</i>	<i>2067fff7</i>
	<i>or \$4, \$7, \$2</i>	<i># \$4 &lt;= 3 or 5 = 7</i>	<i>c</i>	<i>00e22025</i>
	<i>and \$5, \$3, \$4</i>	<i># \$5 &lt;= 12 and 7 = 4</i>	<i>10</i>	<i>00642824</i>
	<i>add \$5, \$5, \$4</i>	<i># \$5 = 4 + 7 = 11</i>	<i>14</i>	<i>00a42820</i>
	<i>beq \$5, \$7, end</i>	<i># shouldn't be taken</i>	<i>18</i>	<i>10a7000a</i>
	<i>slt \$4, \$3, \$4</i>	<i># \$4 = 12 &lt; 7 = 0</i>	<i>1c</i>	<i>0064202a</i>
	<i>beq \$4, \$0, around</i>	<i># should be taken</i>	<i>20</i>	<i>10800001</i>
	<i>addi \$5, \$0, 0</i>	<i># shouldn't execute</i>	<i>24</i>	<i>20050000</i>
around:	<i>slt \$4, \$7, \$2</i>	<i># \$4 = 3 &lt; 5 = 1</i>	<i>28</i>	<i>00e2202a</i>
	<i>add \$7, \$4, \$5</i>	<i># \$7 = 1 + 11 = 12</i>	<i>2c</i>	<i>00853820</i>
	<i>sub \$7, \$7, \$2</i>	<i># \$7 = 12 - 5 = 7</i>	<i>30</i>	<i>00e23822</i>
	<i>sw \$7, 68(\$3)</i>	<i># [80] = 7</i>	<i>34</i>	<i>ac670044</i>
	<i>lw \$2, 80(\$0)</i>	<i># \$2 = [80] = 7</i>	<i>38</i>	<i>8c020050</i>
	<i>j end</i>	<i># should be taken</i>	<i>3c</i>	<i>08000011</i>
	<i>addi \$2, \$0, 1</i>	<i># shouldn't execute</i>	<i>40</i>	<i>20020001</i>
end:	<i>sw \$2, 84(\$0)</i>	<i># write adr 84 = 7</i>	<i>44</i>	<i>ac020054</i>
	<i>j main</i>	<i># go back to beginning</i>	<i>48</i>	<i>08000c00</i>

## 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.

The screenshot shows the MARS MIPS assembler interface. The 'Text Segment' window displays the assembly code with comments. The 'Registers' window shows the state of MIPS registers after execution. The 'Data Segment' window shows memory addresses and their values. The 'Mars Messages' window shows status messages.

**Text Segment:**

```

1  # mipstest.asm
2  # Test the following MIPS instructions.
3  # add, sub, and, or, slt, addi, lw, sw, beq, j
4
5  # Assembly
6  main: addi $2, $0, 5      # initialize $2 = 5      0  20020005
7        addi $3, $0, 12    # initialize $3 = 12     4  2003000c
8        addi $7, $3, -9    # initialize $7 = 3      8  2067ffff
9        or $4, $7, $2      # $4 <= 3 or 5 = 7      c  00e22025
10       and $5, $3, $4      # $5 <= 12 and 7 = 4    10  00642824
11       add $5, $5, $4      # $5 = 4 + 7 = 11      14  00a42820
12       beq $5, $7, end     # shouldn't be taken    18  10a7000a
13       slt $4, $3, $4      # $4 = 12 < 7 = 0      1c  0064202a
14       beq $4, $0, around  # should be taken     20  10800001
15       addi $5, $0, 0      # shouldn't execute 24  20050000
16
17  around: slt $4, $7, $2    # $4 = 3 < 5 = 1      28  00e2202a
18         add $7, $4, $5    # $7 = 1 + 11 = 12     2c  00853820
19         sub $7, $7, $2    # $7 = 12 - 5 = 7      30  00e23822
20         sw $7, 68($3)     # [80] = 7          34  ac670044
21         lw $2, 80($0)     # $2 = [80] = 7    38  8c020050
22         j end            # should be taken      3c  08000011
23         addi $2, $0, 1    # shouldn't execute 40  20020001
24
25  end: sw $2, 84($0)       # write adr 84 = 7    44  ac020054
26       j main            # go back to beginning 48  08000c00
27

```

**Registers:**

Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	12
\$a0	4	11
\$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
\$s8	24	0
\$s9	25	0
\$a0	26	0
\$a1	27	0
\$a2	28	6144
\$a3	29	12214
\$t0	30	0
\$t1	31	12214
\$t2	32	0
\$t3	33	0
\$t4	34	0
\$t5	35	0
\$t6	36	0
\$t7	37	0
\$s0	38	0
\$s1	39	0
\$s2	40	0
\$s3	41	0
\$s4	42	0
\$s5	43	0
\$s6	44	0
\$s7	45	0
\$s8	46	0
\$s9	47	0
\$a0	48	0
\$a1	49	0
\$a2	50	0
\$a3	51	0
\$t0	52	0
\$t1	53	0
\$t2	54	0
\$t3	55	0
\$t4	56	0
\$t5	57	0
\$t6	58	0
\$t7	59	0
\$s0	60	0
\$s1	61	0
\$s2	62	0
\$s3	63	0
\$s4	64	0
\$s5	65	0
\$s6	66	0
\$s7	67	0
\$s8	68	0
\$s9	69	0
\$a0	70	0
\$a1	71	0
\$a2	72	0
\$a3	73	0
\$t0	74	0
\$t1	75	0
\$t2	76	0
\$t3	77	0
\$t4	78	0
\$t5	79	0
\$t6	80	0
\$t7	81	0
\$s0	82	0
\$s1	83	0
\$s2	84	0
\$s3	85	0
\$s4	86	0
\$s5	87	0
\$s6	88	0
\$s7	89	0
\$s8	90	0
\$s9	91	0
\$a0	92	0
\$a1	93	0
\$a2	94	0
\$a3	95	0
\$t0	96	0
\$t1	97	0
\$t2	98	0
\$t3	99	0
\$t4	100	0
\$t5	101	0
\$t6	102	0
\$t7	103	0
\$s0	104	0
\$s1	105	0
\$s2	106	0
\$s3	107	0
\$s4	108	0
\$s5	109	0
\$s6	110	0
\$s7	111	0
\$s8	112	0
\$s9	113	0
\$a0	114	0
\$a1	115	0
\$a2	116	0
\$a3	117	0
\$t0	118	0
\$t1	119	0
\$t2	120	0
\$t3	121	0
\$t4	122	0
\$t5	123	0
\$t6	124	0
\$t7	125	0
\$s0	126	0
\$s1	127	0
\$s2	128	0
\$s3	129	0
\$s4	130	0
\$s5	131	0
\$s6	132	0
\$s7	133	0
\$s8	134	0
\$s9	135	0
\$a0	136	0
\$a1	137	0
\$a2	138	0
\$a3	139	0
\$t0	140	0
\$t1	141	0
\$t2	142	0
\$t3	143	0
\$t4	144	0
\$t5	145	0
\$t6	146	0
\$t7	147	0
\$s0	148	0
\$s1	149	0
\$s2	150	0
\$s3	151	0
\$s4	152	0
\$s5	153	0
\$s6	154	0
\$s7	155	0
\$s8	156	0
\$s9	157	0
\$a0	158	0
\$a1	159	0
\$a2	160	0
\$a3	161	0
\$t0	162	0
\$t1	163	0
\$t2	164	0
\$t3	165	0
\$t4	166	0
\$t5	167	0
\$t6	168	0
\$t7	169	0
\$s0	170	0
\$s1	171	0
\$s2	172	0
\$s3	173	0
\$s4	174	0
\$s5	175	0
\$s6	176	0
\$s7	177	0
\$s8	178	0
\$s9	179	0
\$a0	180	0
\$a1	181	0
\$a2	182	0
\$a3	183	0
\$t0	184	0
\$t1	185	0
\$t2	186	0
\$t3	187	0
\$t4	188	0
\$t5	189	0
\$t6	190	0
\$t7	191	0
\$s0	192	0
\$s1	193	0
\$s2	194	0
\$s3	195	0
\$s4	196	0
\$s5	197	0
\$s6	198	0
\$s7	199	0
\$s8	200	0
\$s9	201	0
\$a0	202	0
\$a1	203	0
\$a2	204	0
\$a3	205	0
\$t0	206	0
\$t1	207	0
\$t2	208	0
\$t3	209	0
\$t4	210	0
\$t5	211	0
\$t6	212	0
\$t7	213	0
\$s0	214	0
\$s1	215	0
\$s2	216	0
\$s3	217	0
\$s4	218	0
\$s5	219	0
\$s6	220	0
\$s7	221	0
\$s8	222	0
\$s9	223	0
\$a0	224	0
\$a1	225	0
\$a2	226	0
\$a3	227	0
\$t0	228	0
\$t1	229	0
\$t2	230	0
\$t3	231	0
\$t4	232	0
\$t5	233	0
\$t6	234	0
\$t7	235	0
\$s0	236	0
\$s1	237	0
\$s2	238	0
\$s3	239	0
\$s4	240	0
\$s5	241	0
\$s6	242	0
\$s7	243	0
\$s8	244	0
\$s9	245	0
\$a0	246	0
\$a1	247	0
\$a2	248	0
\$a3	249	0
\$t0	250	0
\$t1	251	0
\$t2	252	0
\$t3	253	0
\$t4	254	0
\$t5	255	0

**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
0x00000004	0	0	0	0	0	0	0	0
0x00000008	0	0	0	0	0	0	0	0
0x0000000c	0	0	0	0	0	0	0	0
0x00000010	0	0	0	0	0	0	0	0
0x00000014	0	0	0	0	0	0	0	0
0x00000018	0	0	0	0	0	0	0	0
0x0000001c	0	0	0	0	0	0	0	0
0x00000020	0	0	0	0	0	0	0	0
0x00000024	0	0	0	0	0	0	0	0
0x00000028	0	0	0	0	0	0	0	0
0x0000002c	0	0	0	0	0	0	0	0
0x00000030	0	0	0	0	0	0	0	0
0x00000034	0	0	0	0	0	0	0	0
0x00000038	0	0	0	0	0	0	0	0
0x0000003c	0	0	0	0	0	0	0	0
0x00000040	0	0	0	0	0	0	0	0
0x00000044	0	0	0	0	0	0	0	0
0x00000048	0	0	0	0	0	0	0	0
0x0000004c	0	0	0	0	0	0	0	0
0x00000050	0	0	0	0	0	0	0	0
0x00000054	0	0	0	0	0	0	0	0
0x00000058	0	0	0	0	0	0	0	0
0x0000005c	0	0	0	0	0	0	0	0
0x00000060	0	0	0	0	0	0	0	0
0x00000064	0	0	0	0	0	0	0	0
0x00000068	0	0	0	0	0	0	0	0
0x0000006c	0	0	0	0	0	0	0	0
0x00000070	0	0	0	0	0	0	0	0
0x00000074	0	0	0	0	0	0	0	0
0x00000078	0	0	0	0	0	0	0	0
0x0000007c	0	0	0	0	0	0	0	0
0x00000080	0	0	0	0	0	0	0	0
0x00000084	0	0	0	0	0	0	0	0
0x00000088	0	0	0	0	0	0	0	0
0x0000008c	0	0	0	0	0	0	0	0
0x00000090	0	0	0	0	0	0	0	0
0x00000094	0	0	0	0	0	0	0	0
0x00000098	0	0	0	0	0	0	0	0
0x0000009c	0	0	0	0	0	0	0	0
0x000000a0	0	0	0	0	0	0	0	0
0x000000a4	0	0	0	0	0	0	0	0
0x000000a8	0	0	0	0	0	0	0	0
0x000000ac	0	0	0	0	0	0	0	0
0x000000b0	0	0	0	0	0	0	0	0
0x000000b4	0	0	0	0	0	0	0	0
0x000000b8	0	0	0	0	0	0	0	0
0x000000bc	0	0	0	0	0	0	0	0
0x000000c0	0	0	0	0	0	0	0	0
0x000000c4	0	0	0	0	0	0	0	0
0x000000c8	0	0	0	0	0	0	0	0
0x000000cc	0	0	0	0	0	0	0	0
0x000000d0	0	0	0	0	0	0	0	0
0x000000d4	0	0	0	0	0	0	0	0
0x000000d8	0	0	0	0	0	0	0	0
0x000000dc	0	0	0	0	0	0	0	0
0x000000e0	0	0	0	0	0	0	0	0
0x000000e4	0	0	0	0	0	0	0	0
0x000000e8	0	0	0	0	0	0	0	0
0x000000ec	0	0	0	0	0	0	0	0
0x000000f0	0	0	0	0	0	0	0	0
0x000000f4	0	0	0	0	0	0	0	0
0x000000f8	0	0	0	0	0	0	0	0
0x000000fc	0	0	0	0	0	0	0	0
0x00000100	0	0	0	0	0	0	0	0

**Mars Messages:**

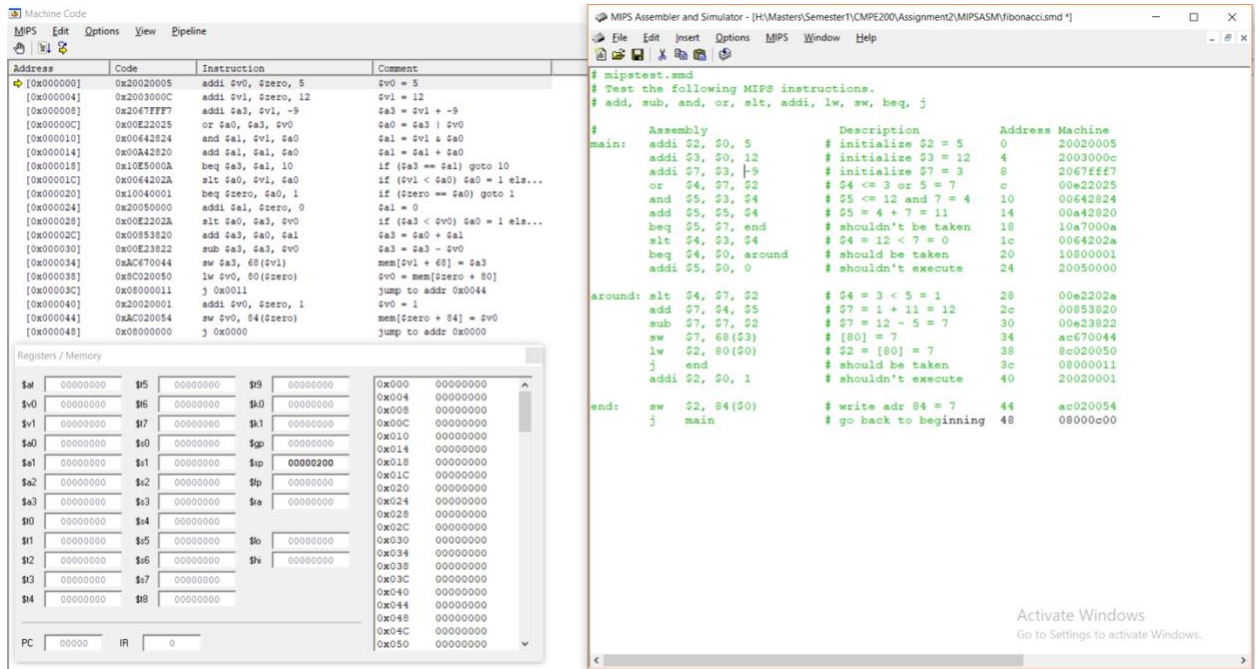
```

Reset: reset completed.
Reset: reset completed.
Reset: reset completed.

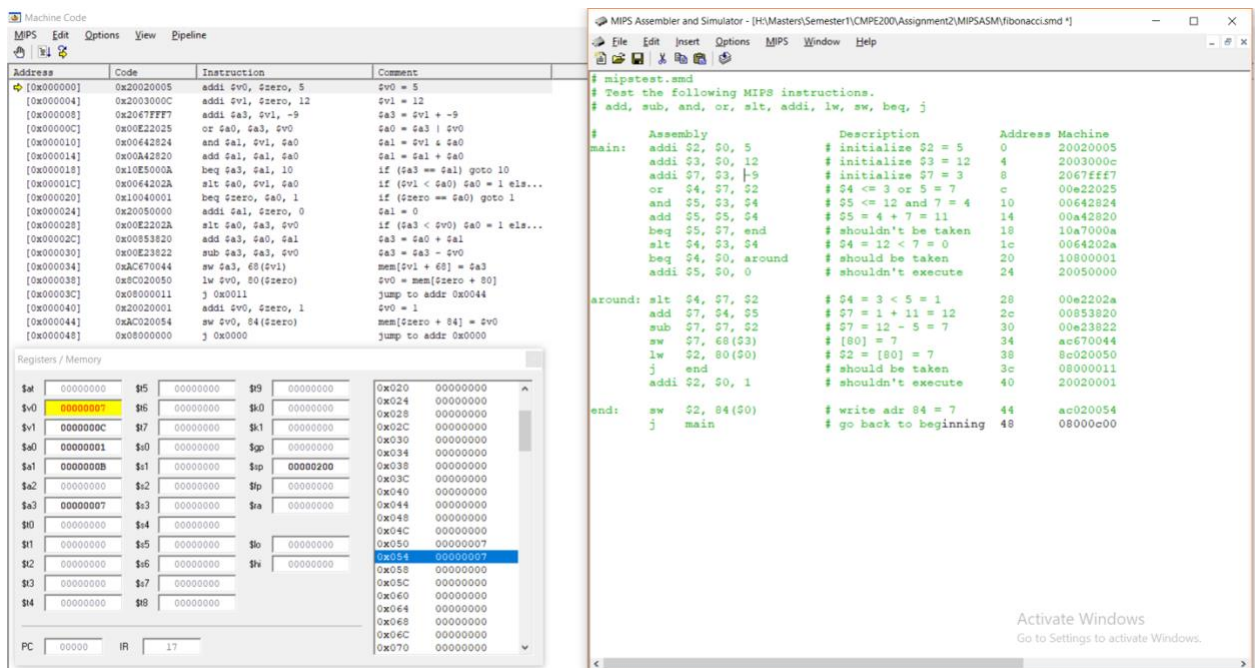
```

## 3. Snippet of source code and register values in MARS

The screenshot shows the MARS MIPS assembler interface. The 'Text Segment' window displays the assembly code with comments. The 'Registers' window shows the state of MIPS registers after execution. The



## 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.

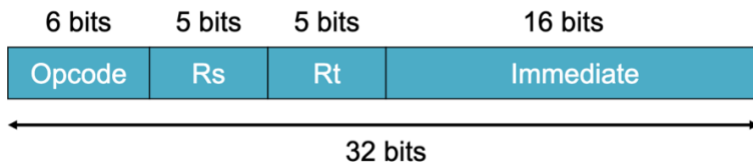
3. *and rd, rs, rt => rd = rs & rt;*  
Perform AND operation between the contents of the register rs, rt and putting the result in register rd.
4. *add rd, rs, rt => rd = rs + rt;*  
Perform ADD operation between the contents of the register rs, rt and putting the result in register rd.
5. *beq rt, rs, 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 00000000000000101

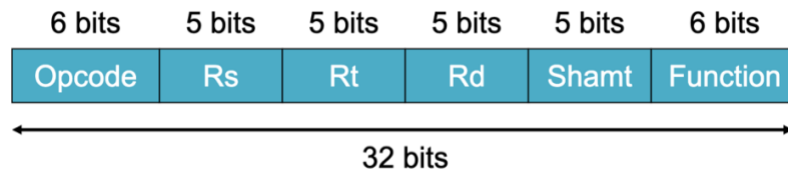
So, the machine code in binary is 0010 0000 0000 0010 0000 0000 0000 0101

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.