San Jose State University
Department of Computer Engineering

CMPE 200 Report

Assignment 2 Report

Title MIPS Instruction Set Architecture & Programming (2)

Semester: Fall 2023 **Date: 09/16/2023**

by

Name: Sai Kashyap Kurella **SID: 016018925**

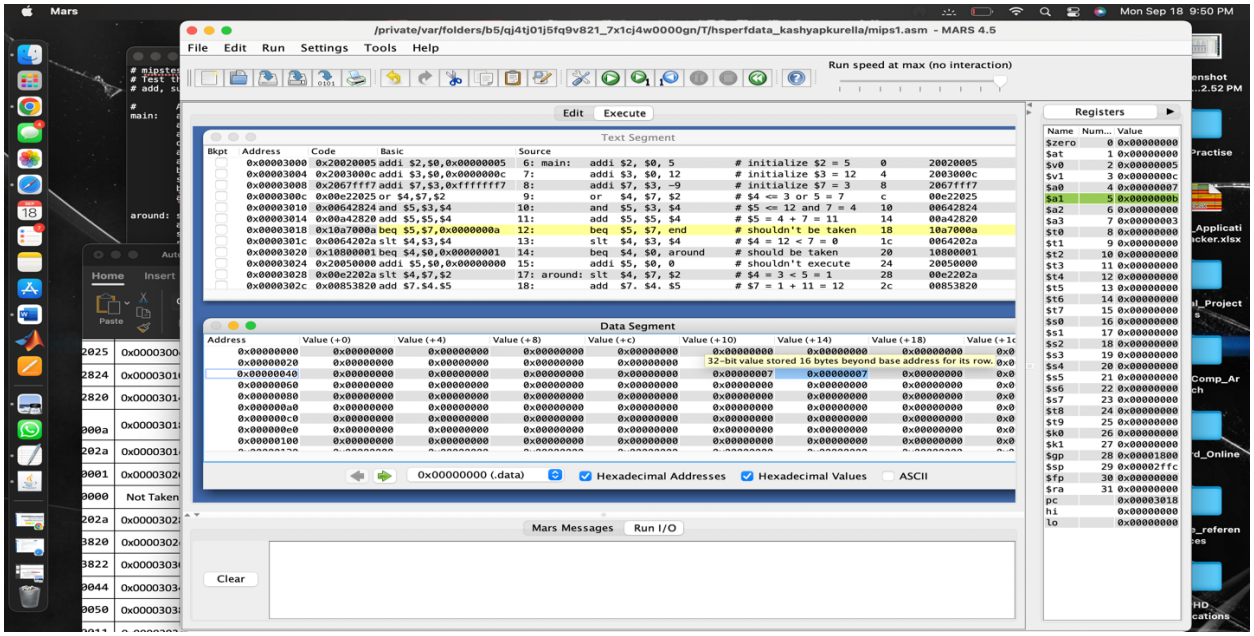
I. SOURCE CODE

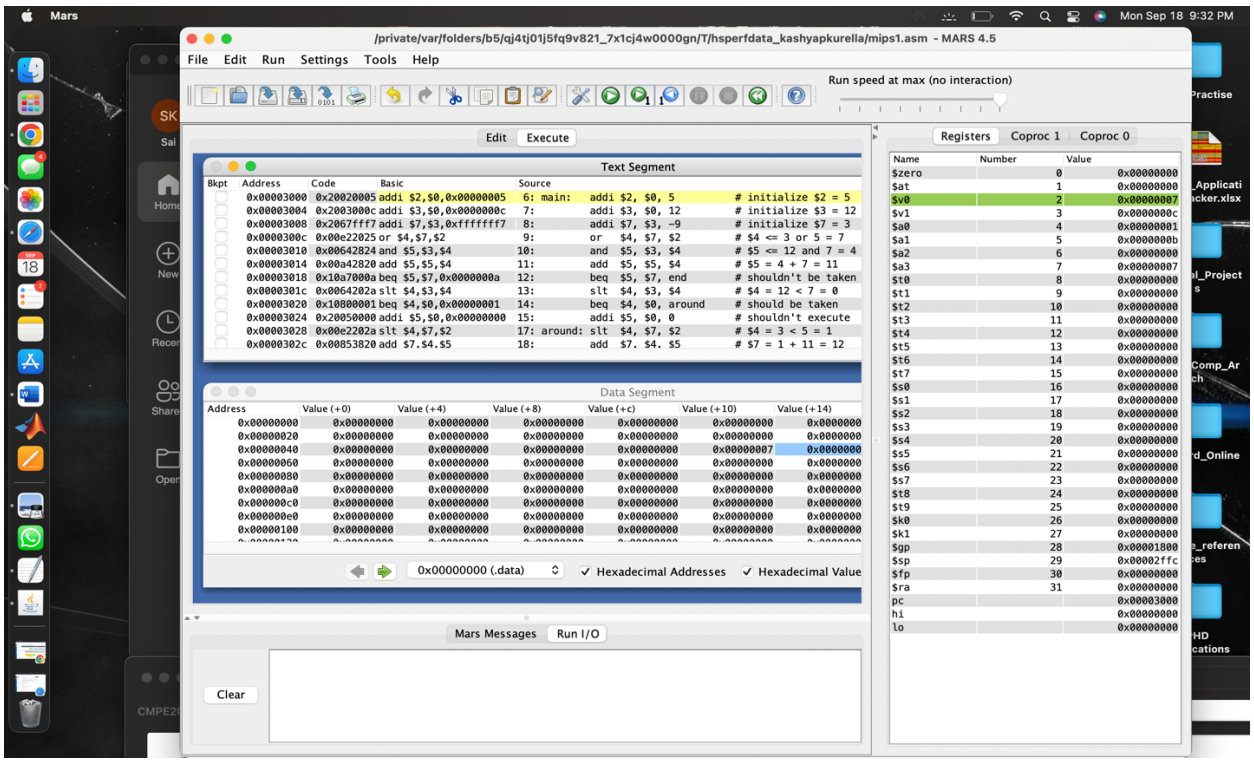
```
# mipstest.smd
# Test the following MIPS instructions.
# add, sub, and, or, slt, addi, lw, sw, beq, j

#       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
```

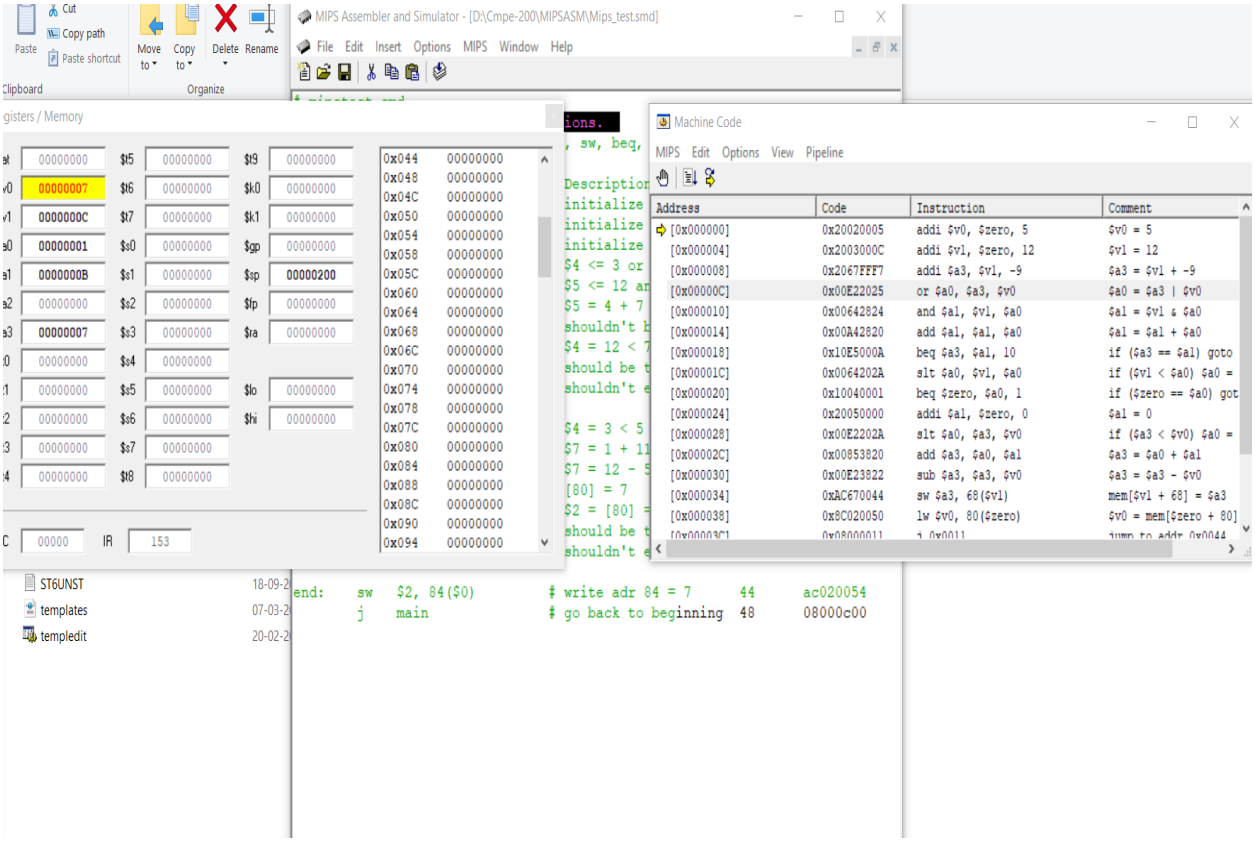
II. TEST LOG

Adr	Machine Code for MARS	Machine Code for MIPSASM	PC	Registers					Memory Content	
				$v0	$v1	$a0	$a1	$a3	[80]	[84]
3000	0x20020005	0x20020005	0x00003000	0x00000007	0x0000000c	0x00000001	0x0000000b	0x00000007	0x00000000	0x00000000
3004	0x2003000c	0x2003000c	0x00003004	0x00000005	0x0000000c	0x00000001	0x0000000b	0x00000007	0x00000000	0x00000000
3008	0x2067fff7	0x2067fff7	0x00003008	0x00000005	0x0000000c	0x00000001	0x0000000b	0x00000007	0x00000000	0x00000000
300c	0x00e22025	0x00e22025	0x0000300c	0x00000005	0x0000000c	0x00000001	0x0000000b	0x00000003	0x00000000	0x00000000
3010	0x00642824	0x00642824	0x00003010	0x00000005	0x0000000c	0x00000007	0x0000000b	0x00000003	0x00000000	0x00000000
3014	0x00a42820	0x00a42820	0x00003014	0x00000005	0x0000000c	0x00000007	0x00000004	0x00000003	0x00000000	0x00000000
3018	0x10a7000a	0x10e5000a	0x00003018	0x00000005	0x0000000c	0x00000007	0x0000000b	0x00000003	0x00000000	0x00000000
301c	0x0064202a	0x0064202a	0x0000301c	0x00000005	0x0000000c	0x00000007	0x0000000b	0x00000003	0x00000000	0x00000000
3020	0x10800001	0x10040001	0x00003020	0x00000005	0x0000000c	0x00000000	0x0000000b	0x00000003	0x00000000	0x00000000
3024	0x20050000	0x20050000	Not Taken	Not Taken	Not Taken	Not Taken	Not Taken	Not Taken	Not Taken	Not Taken
3028	0x00e2202a	0x00e2202a	0x00003028	0x00000005	0x0000000c	0x00000000	0x0000000b	0x00000003	0x00000000	0x00000000
302c	0x00853820	0x00853820	0x0000302c	0x00000005	0x0000000c	0x00000001	0x0000000b	0x00000003	0x00000000	0x00000000
3030	0x00e23822	0x00e23822	0x00003030	0x00000005	0x0000000c	0x00000001	0x0000000b	0x0000000c	0x00000000	0x00000000
3034	0xac670044	0xac670044	0x00003034	0x00000005	0x0000000c	0x00000001	0x0000000b	0x00000007	0x00000007	0x00000000
3038	0x8c020050	0x8c020050	0x00003038	0x00000005	0x0000000c	0x00000001	0x0000000b	0x00000007	0x00000007	0x00000000
303c	0x08000c11	0x08000011	0x0000303c	0x00000007	0x0000000c	0x00000001	0x0000000b	0x00000007	0x00000007	0x00000000
3040	0x20020001	0x20020001	Not Taken	Not Taken	Not Taken	Not Taken	Not Taken	Not Taken	Not Taken	Not Taken
3044	0xac020054	0xac020054	0x00003044	0x00000007	0x0000000c	0x00000001	0x0000000b	0x00000007	0x00000007	0x00000000
3048	0x08000c00	0x08000000	0x00003048	0x00000007	0x0000000c	0x00000001	0x0000000b	0x00000007	0x00000007	0x00000007

III. MARS & MIPSASM SNAPSHOTS






Cut
Copy path
Paste shortcut
Paste
Move to
Copy to
Delete
Rename
Clipboard
Organize
MIPS Assembler and Simulator - [D:\Cmpe-200\MIPSASM\Mips_test.smd]
File Edit Insert Options MIPS Window Help
gisters / Memory
$t5 $t6 $t7 $s0 $s1 $s2 $s3 $s4 $s5 $s6 $s7 $t8
$t9 $k0 $k1 $gp $sp $fp $ra $lo $hi
00000007
0000000C
00000001
0000000B
00000007
00000200
IR
153
0x044 0x048 0x04C 0x050 0x054 0x058 0x05C 0x060 0x064 0x068 0x06C 0x070 0x074 0x078 0x07C 0x080 0x084 0x088 0x08C 0x090 0x094
Machine Code
MIPS Edit Options View Pipeline
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
[0x00001C] 0x0064202A slt $a0, $v1, $a0 if ($v1 < $a0) $a0 =
[0x000020] 0x10040001 beq $zero, $a0, 1 if ($zero == $a0) got
[0x000024] 0x20050000 addi $a1, $zero, 0 $a1 = 0
[0x000028] 0x00E2202A slt $a0, $a3, $v0 if ($a3 < $v0) $a0 =
[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]
ST6UNST 18-09-2
templates 07-03-2
templedit 20-02-2
end: sw $2, 84($0) # write adr 84 = 7 44 ac020054
j main # go back to beginning 48 08000c00

IV. DISCUSSIONS

***1. Initialization**:*

At the beginning of the **main** section, there is variable initialization using **addi** instructions. Registers **$2**, **$3**, and **$7** are initialized with values 5, 12, and 3, respectively.

***2. Logical and Arithmetic Operations**:*

or: The **or** instruction performs a bitwise OR operation between registers **$7** and **$2**, storing the result in **$4**. In this case, it calculates 3 OR 5, resulting in 7.

and: The **and** instruction performs a bitwise AND operation between registers **$3** and **$4**, storing the result in **$5**. It calculates 12 AND 7, resulting in 4.

add: The **add** instruction adds registers **$5** and **$4**, storing the result back in **$5**. It calculates 4 + 7, resulting in 11.

slt: The **slt** instruction sets **$4** to 1 if **$3** is less than **$4** (12 < 7). In this case, it sets **$4** to 0.

***3. Branching**:*

beq: There are two **beq** (branch on equal) instructions. The first one checks if **$5** is equal to **$7** and branches to the **end** label if true. The second one checks if **$4** is equal to 0 and branches to the **around** label if true.

***4. Label Usage**:*

Labels like **around**, **end**, and **main** are used as targets for branching instructions, providing control flow within the program.

***5. Memory Operations**:*

sw: The **sw** (store word) instruction stores the value of register **$7** into memory at address offset 68 from the address in register **$3**.

lw: The **lw** (load word) instruction loads a word from memory at address offset 80 from the address in register **$0** (which is typically used as a null register) into register **$2**.

***6. Jump Instructions**:*

j: The **j** (jump) instruction is used to unconditionally jump to the specified label. It is used to control the program flow, both at the **end** and **main** labels.

7. *Commented Instruction Addresses*:

The comments indicate the memory addresses for each instruction in hexadecimal format.

8. *Execution Flow*:

The code has conditional branches (**beq**) and unconditional jumps (**j**), which determine the execution flow of the program. The program starts at the **main** label and loops between the **around** and **end** labels.

V. CONCLUSION

To sum up, this MIPS assembly code performs basic arithmetic and logic operations, utilizes branching for conditional execution, and interacts with memory through load and store operations. The specific behaviour and results of the code execution would depend on the initial values of registers and memory locations, as well as the branching conditions. The machine codes of the corresponding assembly codes and register values were observed on the MARS & MIPSASM assemblers which helped me in realizing how an assembly code is interpreted by the underlying CPU.