Lab 2 - Implementing Decision Instructions in RISC V Assembly Language

Name: Shaheer Qureshi, Areeba Izhar	Student ID: sq09647, ai09625

Task 1:
Code:

Listing 3:

Task1a.s

```
1   # beq x5, x6, label -> opcode, then binary codes for registers, bits for the distance (contain a statement)
2   .text
3   .globl main
4   main:
5       li x22, 10          # i = 0
6       li x23, 10         # j = 10
7       li x20, 5          # g = 5
8       li x21, 3          # h = 3
9       li x19, 0          # f = 0
10  Loop:
11      bne x22, x23, Else # if (i==j): f= g+h  | Else f= g-h
12      add x19, x20, x21
13      beq x0, x0, Exit
14
15  Else:
16      sub x19, x20, x21
17  Exit:
18      j end
19
20  end:
21      j end
22
23  # exit and end are outside the loop bz they are the steps that define
24  #what the count is to cause the jump.
25  #main has the f else condition only
26
27  # ctrl+shift + P: assemble and run
28  # in assember take the middle hex value, convert to binary, use the table in RISC=V greencard reference to distrubite the bits.
29  # cross check values for the imm[12|10:5] and imm[4:1|11] with the binary value to get the value for the distance.
```

Results:

Task1a.s | assembly

```
1   0x00000000   0x00A00B13   addi x22 x0 10
2   0x00000004   0x00A00B93   addi x23 x0 10
3   0x00000008   0x00500A13   addi x20 x0 5
4   0x0000000C   0x00300A93   addi x21 x0 3
5   0x00000010   0x00000993   addi x19 x0 0
6   0x00000014   0x017B1663   bne x22 x23 12
7   0x00000018   0x015A09B3   add x19 x20 x21
8   0x0000001C   0x00000463   beq x0 x0 8
9   0x00000020   0x415A09B3   sub x19 x20 x21
10  0x00000024   0x0040006F   jal x0 4
11  0x00000028   0x0000006F   jal x0 0
```

The hex value for **bne** we obtained was: 0x017B1663
Converting it to binary we obtained:

Field	**imm[12]**	**imm[10:5]**	**rs2**	**rs1**	**funct3**	**imm[4:1]**	**imm[11]**	**opcode**
Value	0	000000	11000	01001	001	0110	0	1100011

Splitting it based on the RISC-V table we obtained the value for our offset:

- 000000001100 = 12 in Decimal.
- Step count = 3

The hex value for **beq** we obtained was: 0x00000463
converting it to binary we obtained:

Field	**imm[12]**	**imm[10:5]**	**rs2**	**rs1**	**funct3**	**imm[4:1]**	**imm[11]**	**opcode**
Value	0	000000	00000	00000	000	0100	0	1100011

Splitting it based on the RISC-V table we obtained the value for our offset:

- 000000001000 = 8 in decimal
- Step count = 2

Task 1:

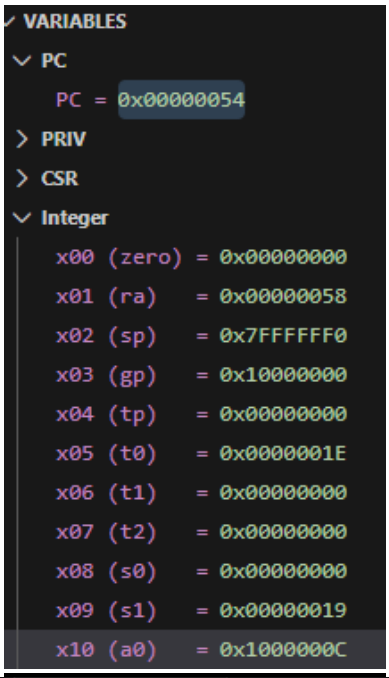
Code:

Listing 4:

Task1b.s

```
1   .data
2   my_array: .space 20
3   .text
4   .globl main
5   main:
6
7       # 1. Load the base address of the array into x25
8       la x25, my_array
9
10      li t0, 10           # Load 10 into temporary register t0
11      sw t0, 0(x25)       # Store t0 at array[0] (offset 0)
12
13      li t0, 10           # Load 10 into t0
14      sw t0, 4(x25)       # Store t0 at array[1] (offset 4)
15
16      li t0, 10           # Load 10 into t0
17      sw t0, 8(x25)       # Store t0 at array[2] (offset 8)
18
19      li t0, 25           # Load 25 into t0
20      sw t0, 12(x25)      # Store t0 at array[3] (offset 12)
21
22      li t0, 30           # Load 30 into t0
23      sw t0, 16(x25)      # Store t0 at array[4] (offset 16)
24
25      li x22, 0           # x22 (i) = 0 (Start index)
26      li x24, 10          # x24 (k) = 10 (The value to skip)
27
28  Loop:
29      slli x10, x22, 2
30      add x10, x10, x25
31      lw x9, 0(x10)
32      bne x9, x24, Exit
33      addi x22, x22, 1
34      beq x0, x0, Loop
35  Exit:
36      j end
37  end:
38      j end
```

Results:



Task1a.s | assembly | Task1b.s

```
1   0x00000000      0x10000C97      auipc x25 65536
2   0x00000004      0x000C8C93      addi x25 x25 0
3   0x00000008      0x00A00293      addi x5 x0 10
4   0x0000000C      0x005CA023      sw x5 0(x25)
5   0x00000010      0x00A00293      addi x5 x0 10
6   0x00000014      0x005CA223      sw x5 4(x25)
7   0x00000018      0x00A00293      addi x5 x0 10
8   0x0000001C      0x005CA423      sw x5 8(x25)
9   0x00000020      0x01900293      addi x5 x0 25
10  0x00000024      0x005CA623      sw x5 12(x25)
11  0x00000028      0x01E00293      addi x5 x0 30
12  0x0000002C      0x005CA823      sw x5 16(x25)
13  0x00000030      0x00000B13      addi x22 x0 0
14  0x00000034      0x00A00C13      addi x24 x0 10
15  0x00000038      0x002B1513      slli x10 x22 2
16  0x0000003C      0x01950533      add x10 x10 x25
17  0x00000040      0x00052483      lw x9 0(x10)
18  0x00000044      0x01849663      bne x9 x24 12
19  0x00000048      0x001B0B13      addi x22 x22 1
20  0x0000004C      0xFE0006E3      beq x0 x0 -20
21  0x00000050      0x0040006F      jal x0 4
22  0x00000054      0x0000006F      jal x0 0
```

The hex value for **bne** operation obtained is: 0x01849663
Converting it to binary we obtained: 00000001100001001001011001100011

Field	**imm[12]**	**imm[10:5]**	**rs2**	**rs1**	**funct3**	**imm[4:1]**	**imm[11]**	**opcode**
Value	0	000000	11000	01001	001	0110	0	1100011

Splitting it based on the RISC-V table we obtained the value for our offset:

- 000000001100 = 12 in Decimal.
- Step count = 3

The hex value for **beq** operation obtained is: 0xFE0006E3
Converting it to binary we obtained: 11111110000000000000011011100011

Field	**imm[12]**	**imm[10:5]**	**rs2**	**rs1**	**funct3**	**imm[4:1]**	**imm[11]**	**opcode**
Value	1	111111	00000	00000	000	0110	1	1100011

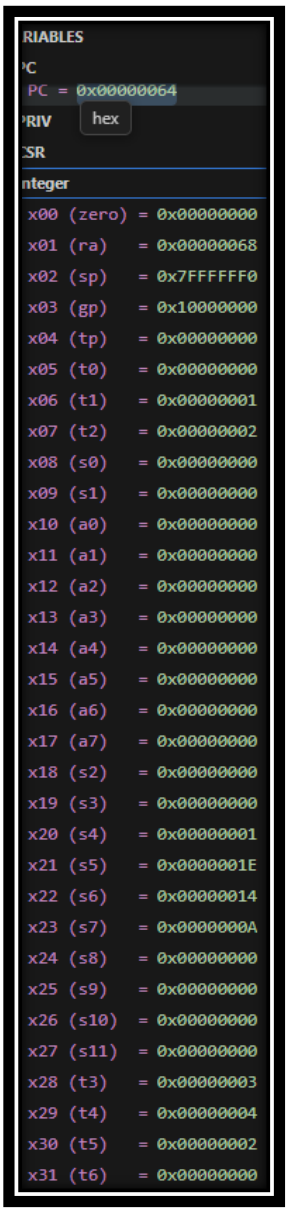
Splitting it based on the RISC-V table we obtained the value for our offset: 111111110110

- Shift Left 1 (adding an implicit 0 – Imm[0]) = 1111111101100
- Taking 2's complement = 0000000010100
- This is equivalent to -20 in Decimal.
- Step count = -20/4 = -5 (5 instructions Up)

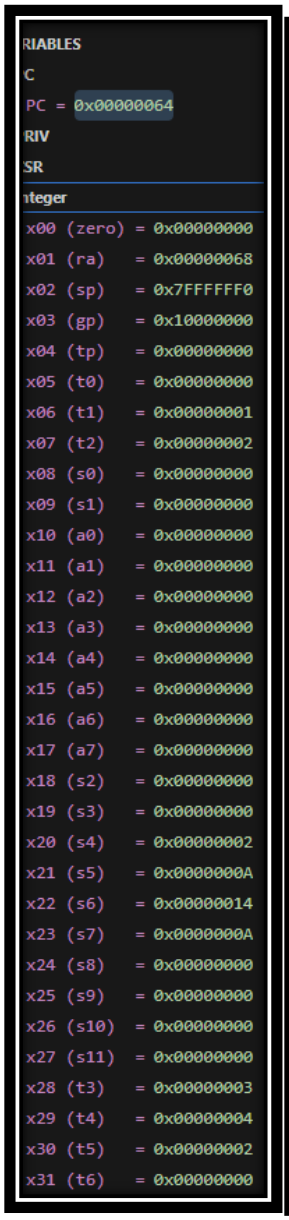
Task 2:
Code:

```
.text
.globl main
main:
    #valyes for a b c and x
    li x21, 0 #a
    li x22, 20          #b = 20
    li x23, 10          #c = 10
    li x20, 4           #x = 1 or 2 or 3 or 4

    li x6, 1 #if x ==1
    li x7, 2 #if x==2
    li x28, 3 #if x==3
    li x29, 4 #if x==4
    li x5, 0 #default case
    li x30, 2 # temp var for division and multi

    beq x20, x6, Case1
    beq x20, x7, Case2
    beq x20, x28, Case3
    beq x20, x29, Case4
    beq x0, x0, Default

Case1:
    add x21, x22, x23
    j end

Case2:
    sub x21, x22, x23
    j end

Case3:
    mul x21, x22, x30
    j end

Case4:
    div x21, x22, x30
    j end

Default:
    li x21, 0
    j end

end:
    j end
```

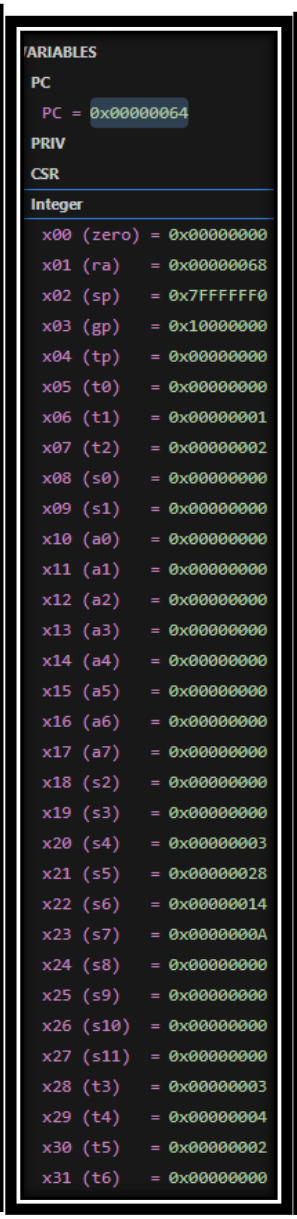
Results:



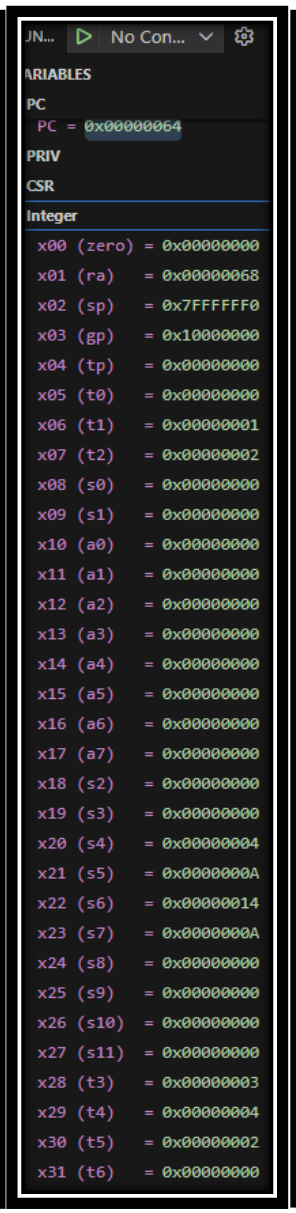
Case 1



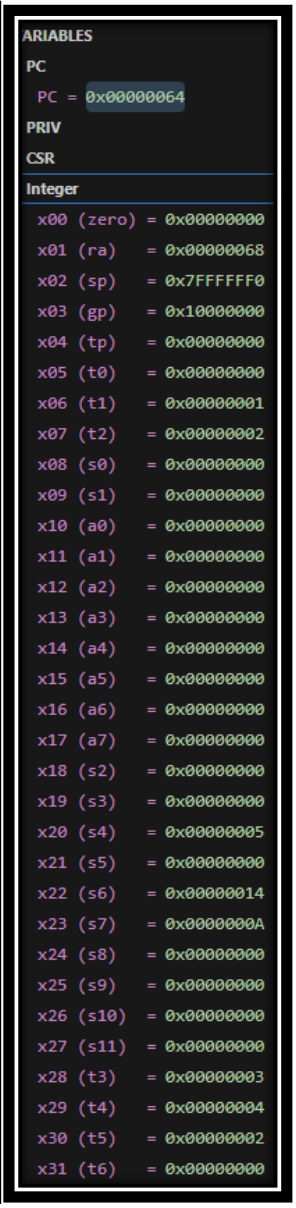
Case 2



Case 3



Case 4



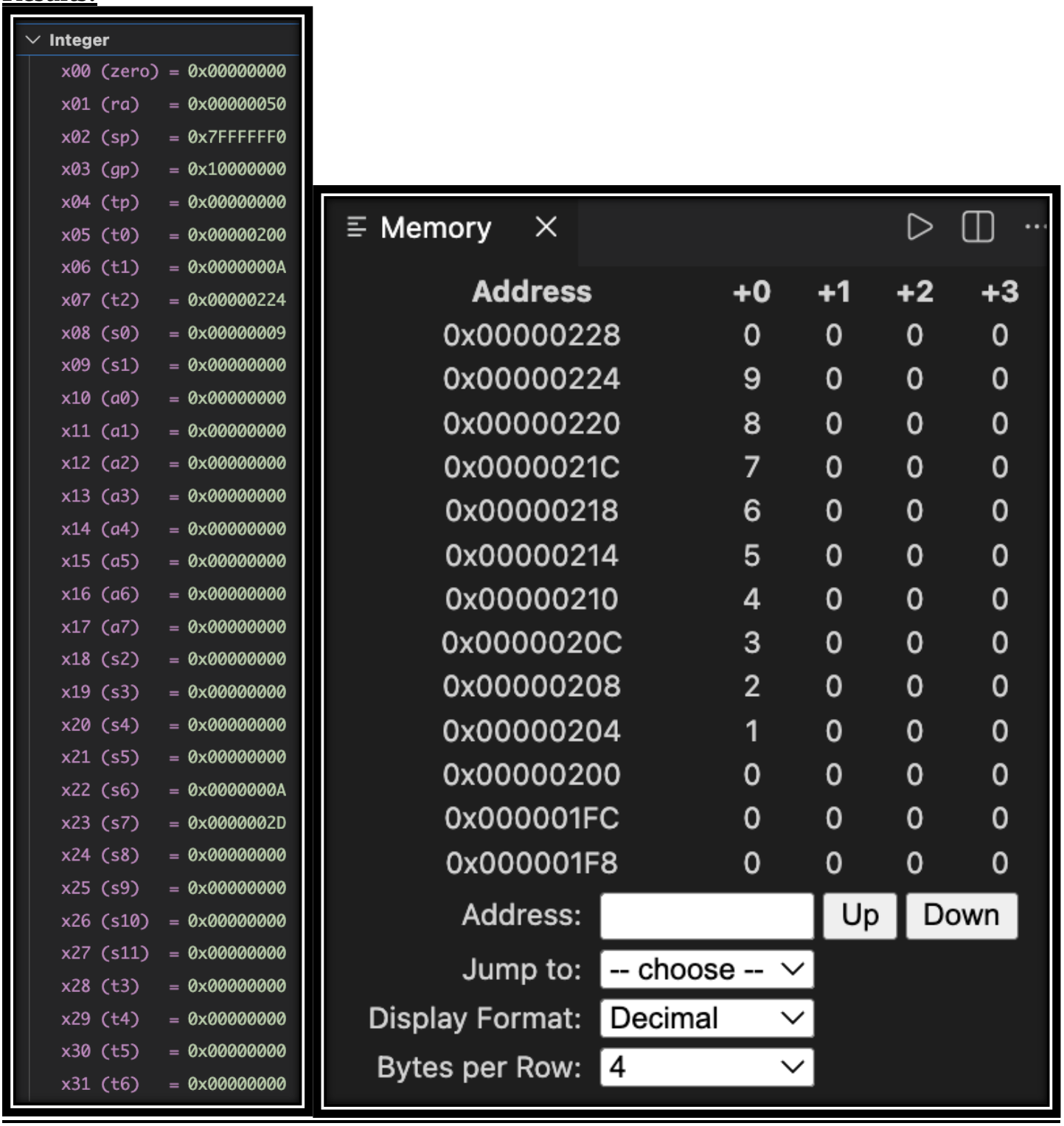
Default

Task 3:

Code:

```
.text
.globl main
main:
    li x22, 0 #i=0
    li x23, 0 #sum=0
    li x5, 0x200 #array address
    li x6, 10 #size of loop =10

loop1:
    bge x22, x6, loop1_done    # if i ⩾ 10, exit init loop
    slli x7, x22, 2           # x7 = i * 4 (byte offset)
    add x7, x7, x5            # x7 = address of a[i]
    sw x22, 0(x7)             # a[i] = i (store i into array)
    addi x22, x22, 1          # i++
    j loop1

loop1_done:
    li x22, 0 #reset i=0 for second loop

loop2:
    bge x22, x6, exit #if i⩾10 exit loop
    slli x7, x22, 2   #jump by 2^2 =4
    add x7, x7, x5 #address of array[i] in x7
    lw x8, 0(x7) #load array[i] in x8
    add x23, x23, x8 #sum = sum + array[i]
    addi x22, x22, 1 #i++
    j loop2

exit:
    j end

end:
    j end
```

Results:



Integer

x00 (zero)	0x00000000
x01 (ra)	0x00000050
x02 (sp)	0x7FFFFFF0
x03 (gp)	0x10000000
x04 (tp)	0x00000000
x05 (t0)	0x00000200
x06 (t1)	0x0000000A
x07 (t2)	0x00000224
x08 (s0)	0x00000009
x09 (s1)	0x00000000
x10 (a0)	0x00000000
x11 (a1)	0x00000000
x12 (a2)	0x00000000
x13 (a3)	0x00000000
x14 (a4)	0x00000000
x15 (a5)	0x00000000
x16 (a6)	0x00000000
x17 (a7)	0x00000000
x18 (s2)	0x00000000
x19 (s3)	0x00000000
x20 (s4)	0x00000000
x21 (s5)	0x00000000
x22 (s6)	0x0000000A
x23 (s7)	0x0000002D
x24 (s8)	0x00000000
x25 (s9)	0x00000000
x26 (s10)	0x00000000
x27 (s11)	0x00000000
x28 (t3)	0x00000000
x29 (t4)	0x00000000
x30 (t5)	0x00000000
x31 (t6)	0x00000000

Memory

Address	+0	+1	+2	+3
0x00000228	0	0	0	0
0x00000224	9	0	0	0
0x00000220	8	0	0	0
0x0000021C	7	0	0	0
0x00000218	6	0	0	0
0x00000214	5	0	0	0
0x00000210	4	0	0	0
0x0000020C	3	0	0	0
0x00000208	2	0	0	0
0x00000204	1	0	0	0
0x00000200	0	0	0	0
0x000001FC	0	0	0	0
0x000001F8	0	0	0	0

Address: Up Down

Jump to: -- choose --

Display Format: Decimal

Bytes per Row: 4

Task 4:

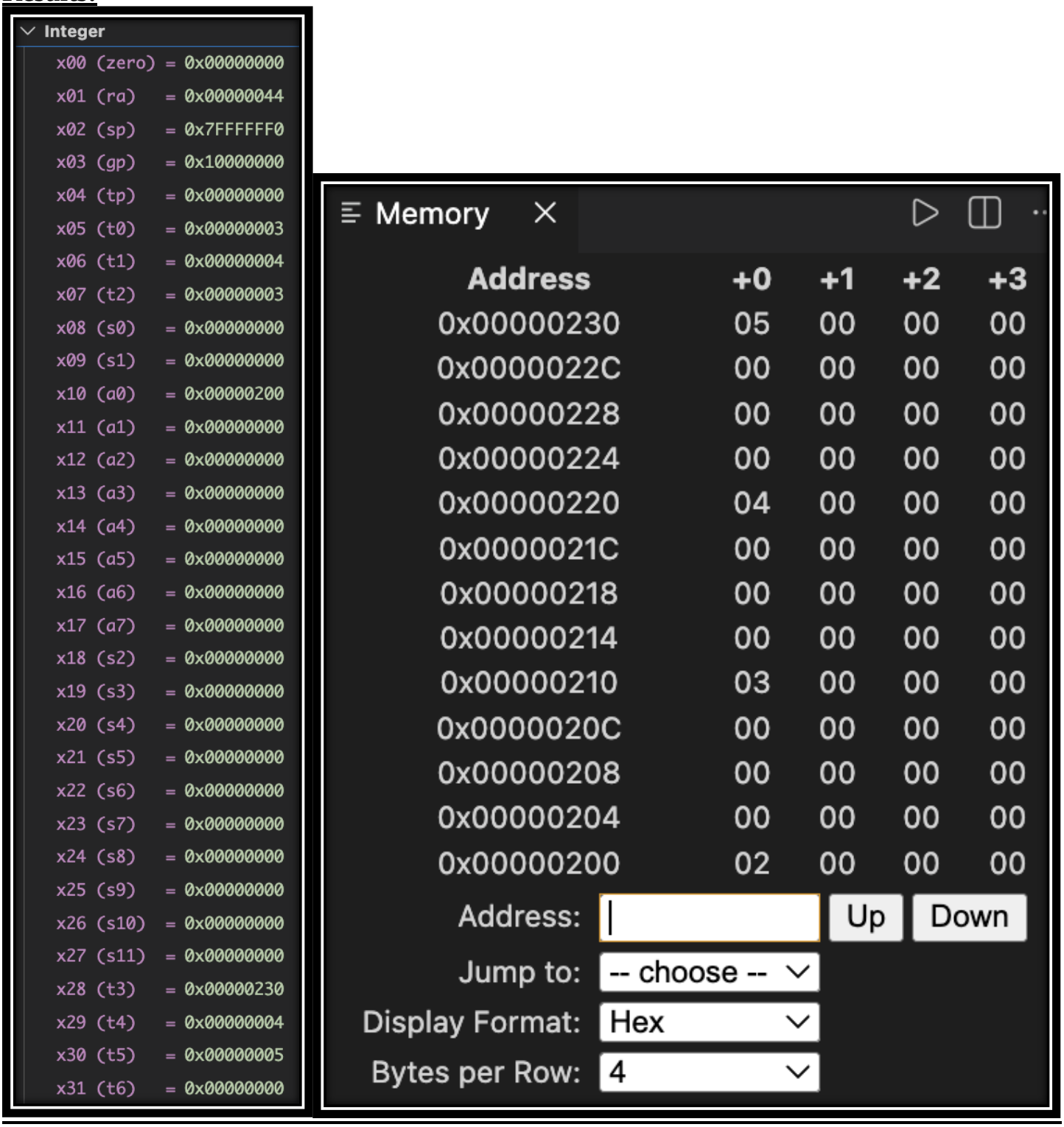
Code:

assembly Task4.s

LabsCA-Git-Shaheer_Areeba > Lab02 > Task4.s

```
.text
.globl main
main:
    li x5, 3 #a -> test val
    li x6, 4 #b -> assumed base vals for testing
    li x7, 0 #i
    li x10, 0x200 #array base address (random)

loopouter:
    bge x7, x5, exit #end program
    li x29, 0 # j will reset everytime i loop 1 iteration end

loopinner:
    bge x29, x6, endinner #if j>=b end inner loop

    slli x28, x29, 4 #x28 = j * 16 (byte offset for D[4*j])
    add x28, x28, x10 #x28 = address of D[4*j]

    add x30, x7, x29 #x30 = i + j
    sw x30, 0(x28) #D[4*j] = i + j

    addi x29, x29, 1 #j++
    j loopinner

endinner:
    addi x7, x7, 1 #i++
    j loopouter

exit:
    j end

end:
    j end
```

Results:


Integer
x00 (zero) = 0x00000000
x01 (ra) = 0x00000044
x02 (sp) = 0x7FFFFFF0
x03 (gp) = 0x10000000
x04 (tp) = 0x00000000
x05 (t0) = 0x00000003
x06 (t1) = 0x00000004
x07 (t2) = 0x00000003
x08 (s0) = 0x00000000
x09 (s1) = 0x00000000
x10 (a0) = 0x00000200
x11 (a1) = 0x00000000
x12 (a2) = 0x00000000
x13 (a3) = 0x00000000
x14 (a4) = 0x00000000
x15 (a5) = 0x00000000
x16 (a6) = 0x00000000
x17 (a7) = 0x00000000
x18 (s2) = 0x00000000
x19 (s3) = 0x00000000
x20 (s4) = 0x00000000
x21 (s5) = 0x00000000
x22 (s6) = 0x00000000
x23 (s7) = 0x00000000
x24 (s8) = 0x00000000
x25 (s9) = 0x00000000
x26 (s10) = 0x00000000
x27 (s11) = 0x00000000
x28 (t3) = 0x00000230
x29 (t4) = 0x00000004
x30 (t5) = 0x00000005
x31 (t6) = 0x00000000
Memory
Address +0 +1 +2 +3
0x00000230 05 00 00 00
0x0000022C 00 00 00 00
0x00000228 00 00 00 00
0x00000224 00 00 00 00
0x00000220 04 00 00 00
0x0000021C 00 00 00 00
0x00000218 00 00 00 00
0x00000214 00 00 00 00
0x00000210 03 00 00 00
0x0000020C 00 00 00 00
0x00000208 00 00 00 00
0x00000204 00 00 00 00
0x00000200 02 00 00 00
Address:
Up
Down
Jump to: -- choose --
Display Format: Hex
Bytes per Row: 4

Assessment Rubric
Lab 2 - Implementing Decision Instructions in RISC V Assembly Language

Name	Student ID:	Section:

Points Distribution:

	Task No.	LR 2 (Code)	LR 5 (Results)
In-Lab	**Task 1**	-	/5
	Task 2	/10	/10
	Task 3	/15	/10
	Task 4	/20	/10
Total Points: 100		**/45**	**/35**
CLO Mapped		**CLO 2**	

Affective Domain Rubric		Points	CLO Mapped
AR7	**Report Submission & Git Upload**	/10 & /10	**CLO 2**

CLO	Total Points	Points Obtained
2	100	
Total	**100**	