



Name: Hamna Munib, Jaweria Amer	Student ID: hm09929, jc09631	section*: T3
--	-------------------------------------	---------------------

Lab 1: Getting Started with RISC-V (Assembly Language) in VS Code

Task 1

Git Repository for Lab 01

Lab01-Git-HamnaJaweria (Public)

main 1 Branch 0 Tags

hamnamunib added_bye.txt cb3f8ff · 2 minutes ago 1 Commit

bye.txt added_bye.txt 2 minutes ago

README

Add a README

Help people interested in this repository understand your project.

About

No description, website, or topics provided.

Activity

0 stars

0 watching

0 forks

Releases

No releases published

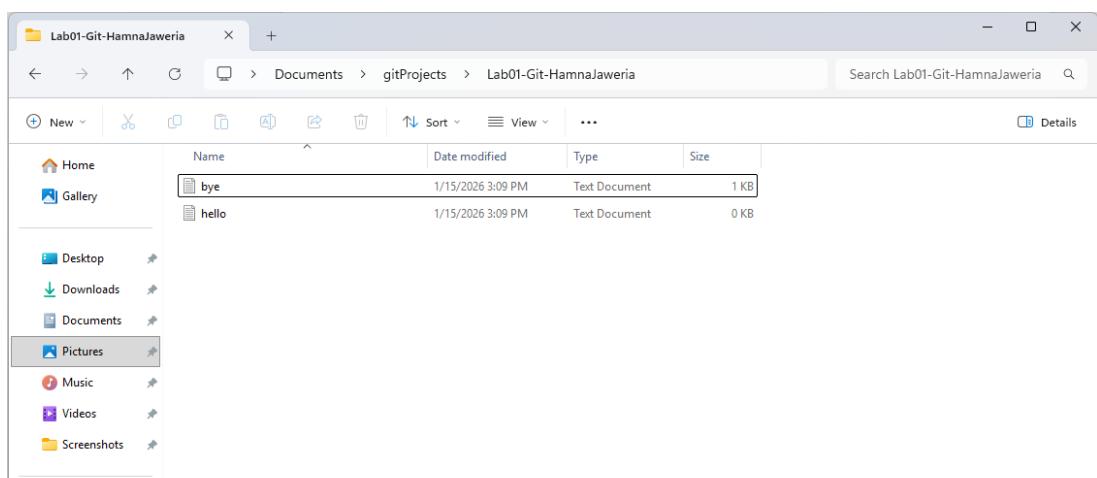
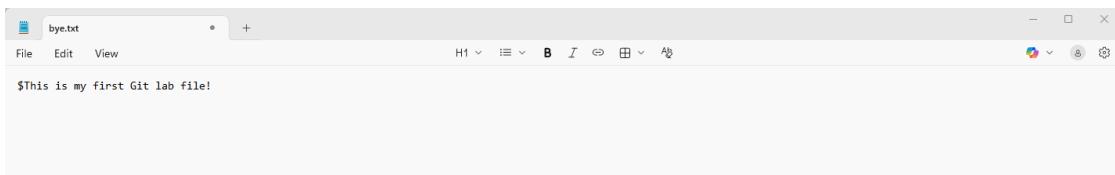
Create a new release

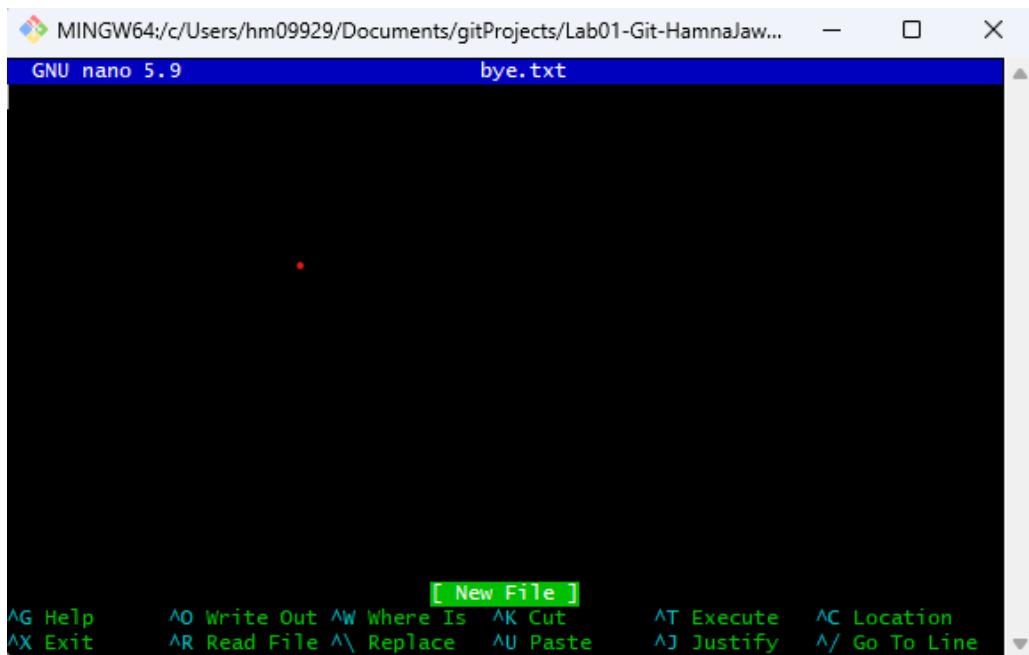
Packages

No packages published

Publish your first package

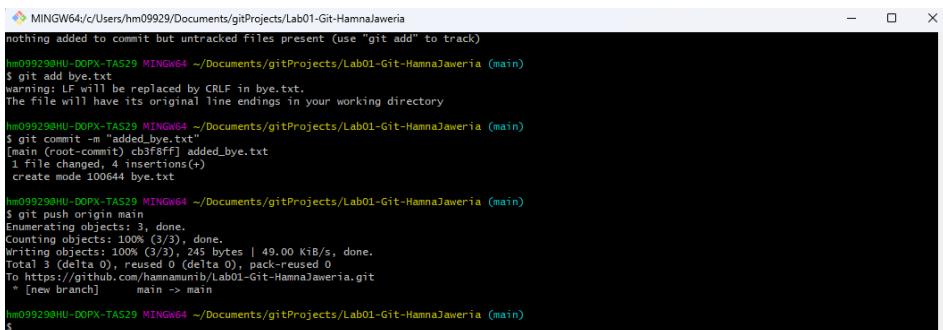
Creating and editing txt file





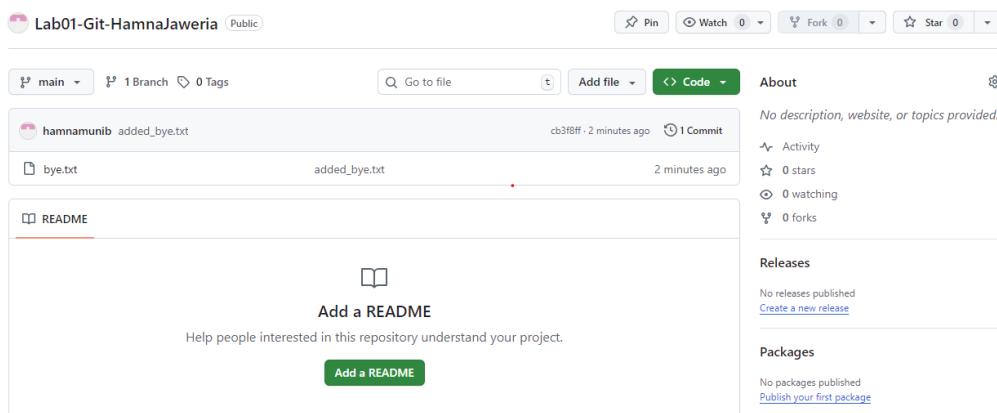
```
MINGW64:/c/Users/hm09929/Documents/gitProjects/Lab01-Git-HamnaJaw... GNU nano 5.9 bye.txt [ New File ] ^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location ^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line
```

Pushing to Git



```
nothing added to commit but untracked files present (use "git add" to track)
hm09929@HABIB-DESKTOP-98H:~/Documents/gitProjects/Lab01-Git-HamnaJaweria (main)
$ git add bye.txt
warning: LF will be replaced by CRLF in bye.txt.
The file will have its original line endings in your working directory
hm09929@HABIB-DESKTOP-98H:~/Documents/gitProjects/Lab01-Git-HamnaJaweria (main)
$ git commit -m "added_bye.txt"
[main (root-commit) cb3f8ff] added_bye.txt
 1 file changed, 4 insertions(+)
 create mode 100644 bye.txt
hm09929@HABIB-DESKTOP-98H:~/Documents/gitProjects/Lab01-Git-HamnaJaweria (main)
$ git push origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 245 bytes | 49.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/hamnamunib/Lab01-Git-HamnaJaweria.git
 * [new branch]      main > main

```



Lab01-Git-HamnaJaweria Public

main 1 Branch 0 Tags

hamnamunib added_bye.txt cb3f8ff · 2 minutes ago 1 Commit

bye.txt added_bye.txt 2 minutes ago

README

Add a README

Help people interested in this repository understand your project.

Add a README

About

No description, website, or topics provided.

Activity

0 stars

0 watching

0 forks

Releases

No releases published

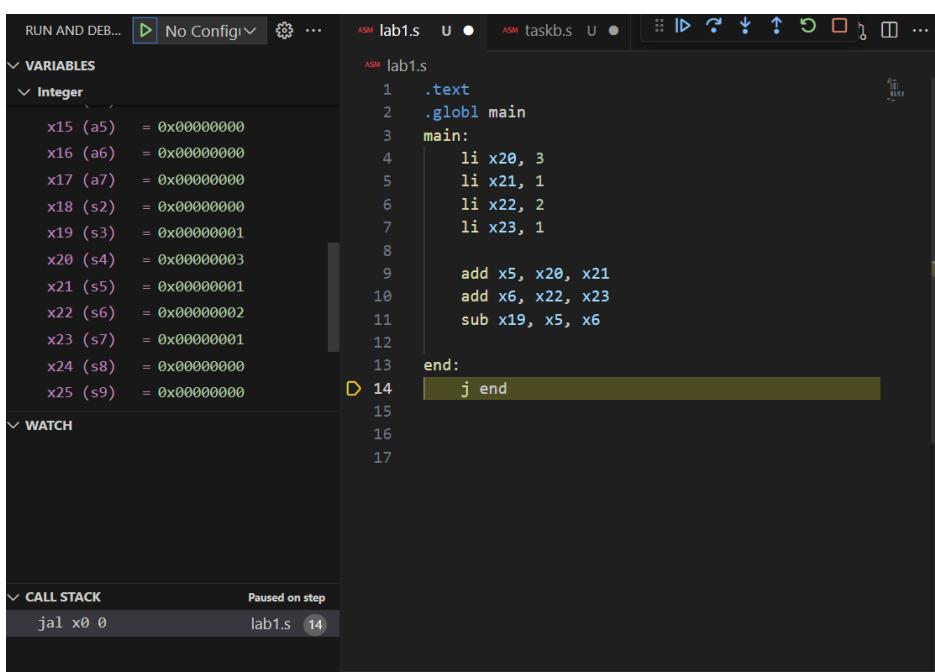
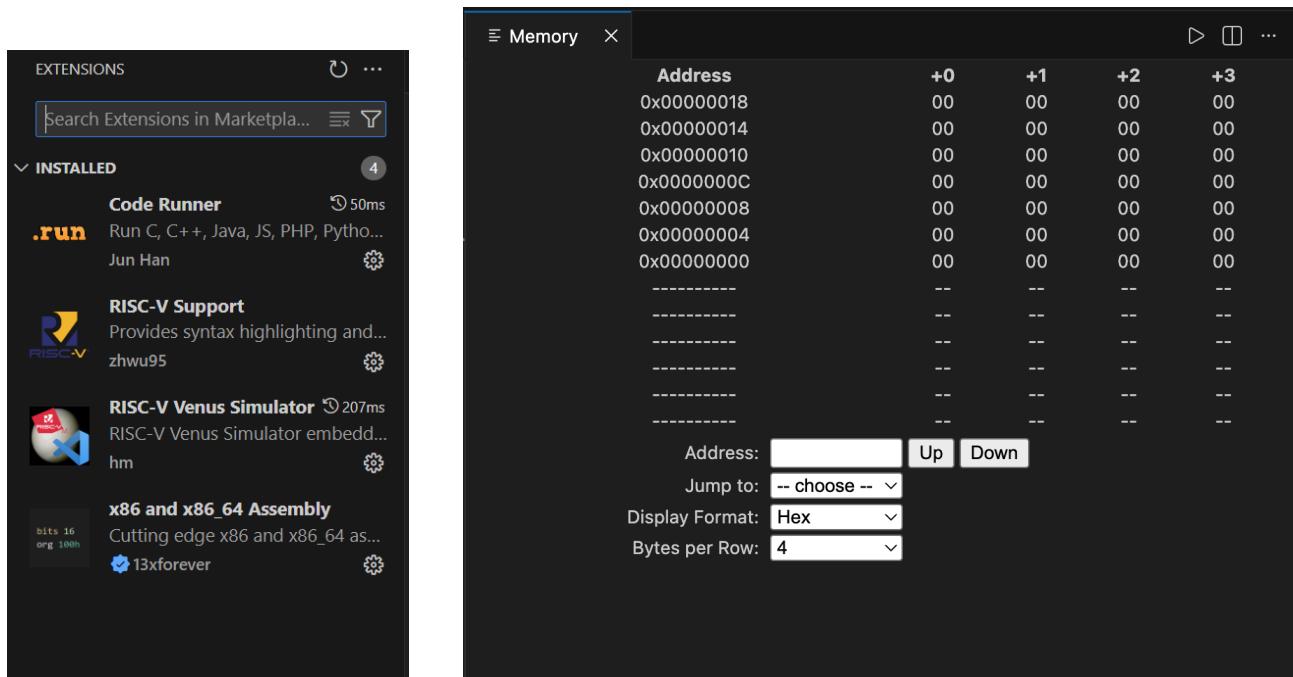
Create a new release

Packages

No packages published

Publish your first package

Task 02: Setting Up VS Code (RISC-V Simulation Environment)



Task 3

Convert the following statement to RISC V. You can use the same registers as given

```

1 int a = 5;
2 int b = 0 + 0; a
3 = b + 32;
4 int d = (a + b) - 5;
5 int e = (((a - d) + (b - a)) + d);
e = a + b + d + e;

```

Code

```

asm task3.s
1 .text
2 .globl main
3 main:
4     li x20, 5 # int a=5
5     li x21, 0 #b
6     # li x22, 0 #c
7     # li x23, 0 #d
8     # li x24, 0 #e
9
10    #x01= a +b
11    #x02= a -d
12
13    addi x21, x22, 0 # int b=0+0
14    addi x20, x21, 32 #int a = b+32
15    add x30, x20, x21 #int random = (a+b)
16    addi x23, x30, -5 # int d = random -5
17    sub x2, x20, x23 #a-d
18    sub x3, x21, x20 #b-a
19    add x24, x2, x3 # e = (a-d) + (b-a)
20    add x24, x24,x23 #e = (a-d) + (b-a) -5
21    add x4, x20, x21 #random =a+b
22    add x5, x23, x24 #random= d+e
23    add x24, x4, x5 #e= a+b+d+e
24
25 end:
26     j end
27

```

Output

VARIABLES	
Integer	
x13 (a3)	= 0x00000000
x14 (a4)	= 0x00000000
x15 (a5)	= 0x00000000
x16 (a6)	= 0x00000000
x17 (a7)	= 0x00000000
x18 (s2)	= 0x00000000
x19 (s3)	= 0x00000000
x20 (s4)	= 0x00000020
x21 (s5)	= 0x00000000
x22 (s6)	= 0x00000000
x23 (s7)	= 0x00000018
x24 (s8)	= 0x00000038
x25 (s9)	= 0x00000000
x26 (s10)	= 0x00000000
x27 (s11)	= 0x00000000
x28 (t3)	= 0x00000000
x29 (t4)	= 0x00000000
x30 (t5)	= 0x00000020
x31 (t6)	= 0x00000000

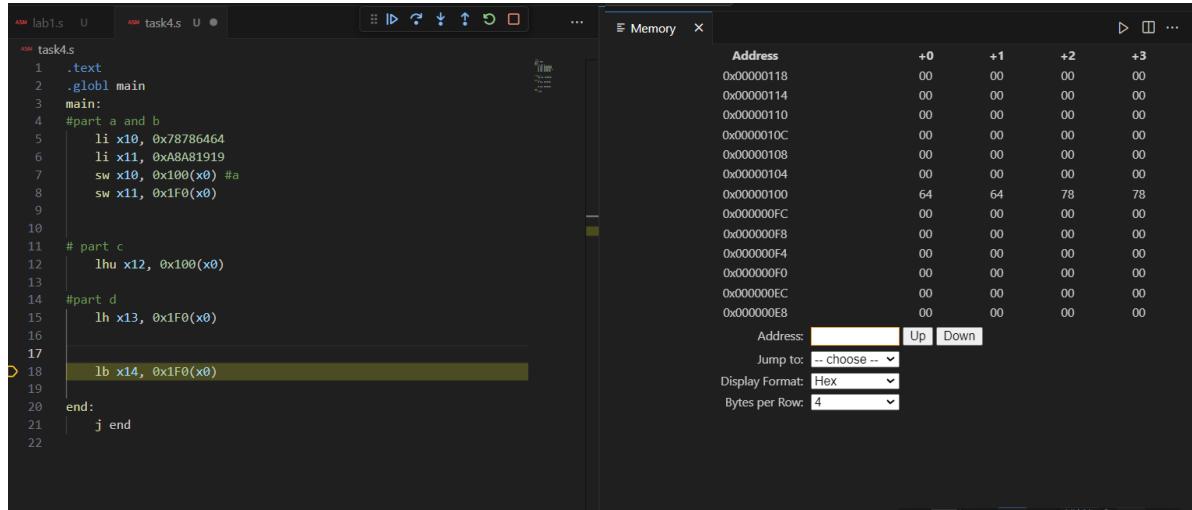
Final value of e = 3B (hex) and 59 (in decimal)
e value 3B stored in register x24

Task 4a

Initialize the register x10 and x11 with values 0x78786464, 0xA8A81919, respectively manually.

Write the RISC-V assembly code for each item below. Try guessing the result in each destination before executing the instruction and corroborate it after execution:

a) Store x10 as unsigned integer at address 0x100.



```

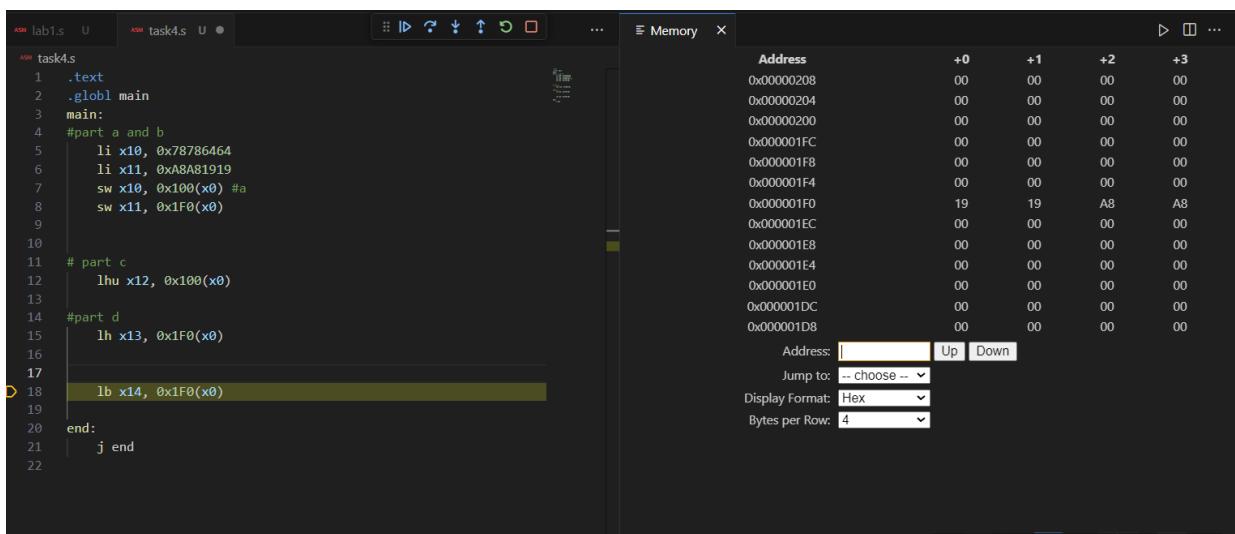
asm lab1.s U   asm task4.s U •
task4.s
1 .text
2 .globl main
3 main:
4 #part a and b
5 li x10, 0x78786464
6 li x11, 0xA8A81919
7 sw x10, 0x100(x0) #a
8 sw x11, 0x1F0(x0)
9
10
11 # part c
12 lhu x12, 0x100(x0)
13
14 #part d
15 lh x13, 0x1F0(x0)
16
17
18 lb x14, 0x1F0(x0)
19
20 end:
21 j end
22

```

Memory dump (Address 0x00000100 to 0x000001E8):

Address	+0	+1	+2	+3
0x00000118	00	00	00	00
0x00000114	00	00	00	00
0x00000110	00	00	00	00
0x0000010C	00	00	00	00
0x00000108	00	00	00	00
0x00000104	00	00	00	00
0x00000100	64	64	78	78
0x000000FC	00	00	00	00
0x000000F8	00	00	00	00
0x000000F4	00	00	00	00
0x000000F0	00	00	00	00
0x000000EC	00	00	00	00
0x000000E8	00	00	00	00

b) Store x11 as unsigned integer at address 0x1F0.



```

asm lab1.s U   asm task4.s U •
task4.s
1 .text
2 .globl main
3 main:
4 #part a and b
5 li x10, 0x78786464
6 li x11, 0xA8A81919
7 sw x10, 0x100(x0) #a
8 sw x11, 0x1F0(x0)
9
10
11 # part c
12 lhu x12, 0x100(x0)
13
14 #part d
15 lh x13, 0x1F0(x0)
16
17
18 lb x14, 0x1F0(x0)
19
20 end:
21 j end
22

```

Memory dump (Address 0x00000100 to 0x000001E8):

Address	+0	+1	+2	+3
0x00000208	00	00	00	00
0x00000204	00	00	00	00
0x00000200	00	00	00	00
0x000001FC	00	00	00	00
0x000001F8	00	00	00	00
0x000001F4	00	00	00	00
0x000001F0	19	19	A8	A8
0x000001EC	00	00	00	00
0x000001E8	00	00	00	00
0x000001E4	00	00	00	00
0x000001E0	00	00	00	00
0x000001DC	00	00	00	00
0x000001D8	00	00	00	00

c) Load an unsigned short integer (two bytes) from address 0x100 in x12.

d) Load a short integer from address 0x1F0 in register x13.

e) Load a singed character from address 0x1F0 in register x14.

Task 4b -- Loop unrolling

1 Assume there are three character arrays a, b, and c located at addresses 0x100, 0x200, 0x300 respectively.

3
4 **for** (**int** i=0 ; i<4; i++)
5 c [i]=a [i]+b [i] ; # c [0]=a [0]+b [0] ;
6

7 Write equivalent RISC-V code **for** the piece of code given. You have not studied loops yet, but the above code is manageable without loop instructions. Also assume that A is a character array, B is a **short** array, and C is an **unsigned** integer array.

Code

```
asm task4bs
3 .globl main
4 main:
5 #a
6 li x1, 1
7 li x2, 2
8 li x3, 3
9 li x4, 4
10
11 #b
12 li x5, 5
13 li x6, 6
14 li x7, 7
15 li x8, 8
16
17 #saving a
18 sw x1, 0x100(x0)
19 sw x2, 0x101(x0)
20 sw x3, 0x102(x0)
21 sw x4, 0x103(x0)
22
23
24 #saving b
25 sw x5, 0x200(x0)
26 sw x6, 0x201(x0)
27 sw x7, 0x202(x0)
28 sw x8, 0x203(x0)
29
30 #loading a
31 lw x11, 0x100(x0)
32 lw x12, 0x101(x0)
33 lw x13, 0x102(x0)
34 lw x14, 0x103(x0)
35
36 #loading b
37 lw x15, 0x200(x0)
38 lw x16, 0x201(x0)
39 lw x17, 0x202(x0)
40 lw x18, 0x203(x0)
41
42 #adding both
#adding both
add x21, x11, x15
add x22, x12, x16
add x22, x13, x17
add x24, x14, x18

#saving in c
sw x21, 0x300(x0)
sw x22, 0x301(x0)
sw x22, 0x302(x0)
sw x23, 0x303(x0)

end:
| j end
```

Output

Array a and b loaded into registers

The screenshot shows a debugger interface with the following details:

- Assembly View:** The code is as follows:

```

3 .globl main
4 main:
5 #a
6 li x1, 1
7 li x2, 2
8 li x3, 3
9 li x4, 4
10
11 #b
12 li x5, 5
13 li x6, 6
14 li x7, 7
15 li x8, 8
16

```
- Registers View:** Shows the following register values:

Register	Value
x00 (zero)	0x00000000
x01 (ra)	0x00000001
x02 (sp)	0x00000002
x03 (gp)	0x00000003
x04 (tp)	0x00000004
x05 (t0)	0x00000005
x06 (t1)	0x00000006
x07 (t2)	0x00000007
x08 (s0)	0x00000008
x09 (s1)	0x00000000

Saved a into memory (x100).

The screenshot shows a memory dump with the following data:

Address	+0	+1	+2	+3
0x00000118	00	00	00	00
0x00000114	00	00	00	00
0x00000110	00	00	00	00
0x0000010C	00	00	00	00
0x00000108	00	00	00	00
0x00000104	00	00	00	00
0x00000100	01	02	03	04
0x000000FC	00	00	00	00
0x000000F8	00	00	00	00
0x000000F4	00	00	00	00
0x000000F0	00	00	00	00
0x000000EC	00	00	00	00
0x000000E8	00	00	00	00

Configuration controls at the bottom:

- Address:
- Up | Down buttons
- Jump to: -- choose --
- Display Format: Hex
- Bytes per Row: 4

Saved b into memory (x200).

The screenshot shows a memory dump with the following data:

Address	+0	+1	+2	+3
0x00000218	00	00	00	00
0x00000214	00	00	00	00
0x00000210	00	00	00	00
0x0000020C	00	00	00	00
0x00000208	00	00	00	00
0x00000204	00	00	00	00
0x00000200	05	06	07	08
0x000001FC	00	00	00	00
0x000001F8	00	00	00	00
0x000001F4	00	00	00	00
0x000001F0	00	00	00	00
0x000001EC	00	00	00	00
0x000001E8	00	00	00	00

Configuration controls at the bottom:

- Address:
- Up | Down buttons
- Jump to: -- choose --
- Display Format: Hex
- Bytes per Row: 4

Result of $a + b$ loaded in registers

```
x21 (s5) = 0x0C0A0806          42  #adding both
x22 (s6) = 0x000C0A08          43  add x21, x11, x15
x23 (s7) = 0x00000C0A          44  add x22, x12, x16
x24 (s8) = 0x0000000C          45  add x23, x13, x17
x25 (s9) = 0x00000000          46  add x24, x14, x18
```

Final Output

Result of a+ b stored in array c (x300)

Memory

Address	+0	+1	+2	+3
0x00000318	00	00	00	00
0x00000314	00	00	00	00
0x00000310	00	00	00	00
0x0000030C	00	00	00	00
0x00000308	00	00	00	00
0x00000304	00	00	00	00
0x00000300	06	08	0A	0C
0x000002FC	00	00	00	00
0x000002F8	00	00	00	00
0x000002F4	00	00	00	00
0x000002F0	00	00	00	00
0x000002EC	00	00	00	00
0x000002E8	00	00	00	00

Address: Up Down

Jump to:

Display Format:

Bytes per Row:



Assessment Rubric

1: Getting Started with RISC-V (Assembly Language) in VS Code

Name:	Student ID:	section*:
-------	-------------	-----------

Points Distribution

	Task No.	LR 2 Code	LR 5 Results
In - Lab	Task 1	/0	/15
	Task 2	/0	/15
	Task 3	/10	/5
	Task 4a	/10	/5
	Task 4b	/10	/10
Total Points: 100		/30	/50
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	

For description of different levels of the mapped rubrics, please refer to the Lab Evaluation Assessment Rubrics and Affective Domain Assessment Rubrics provided here.