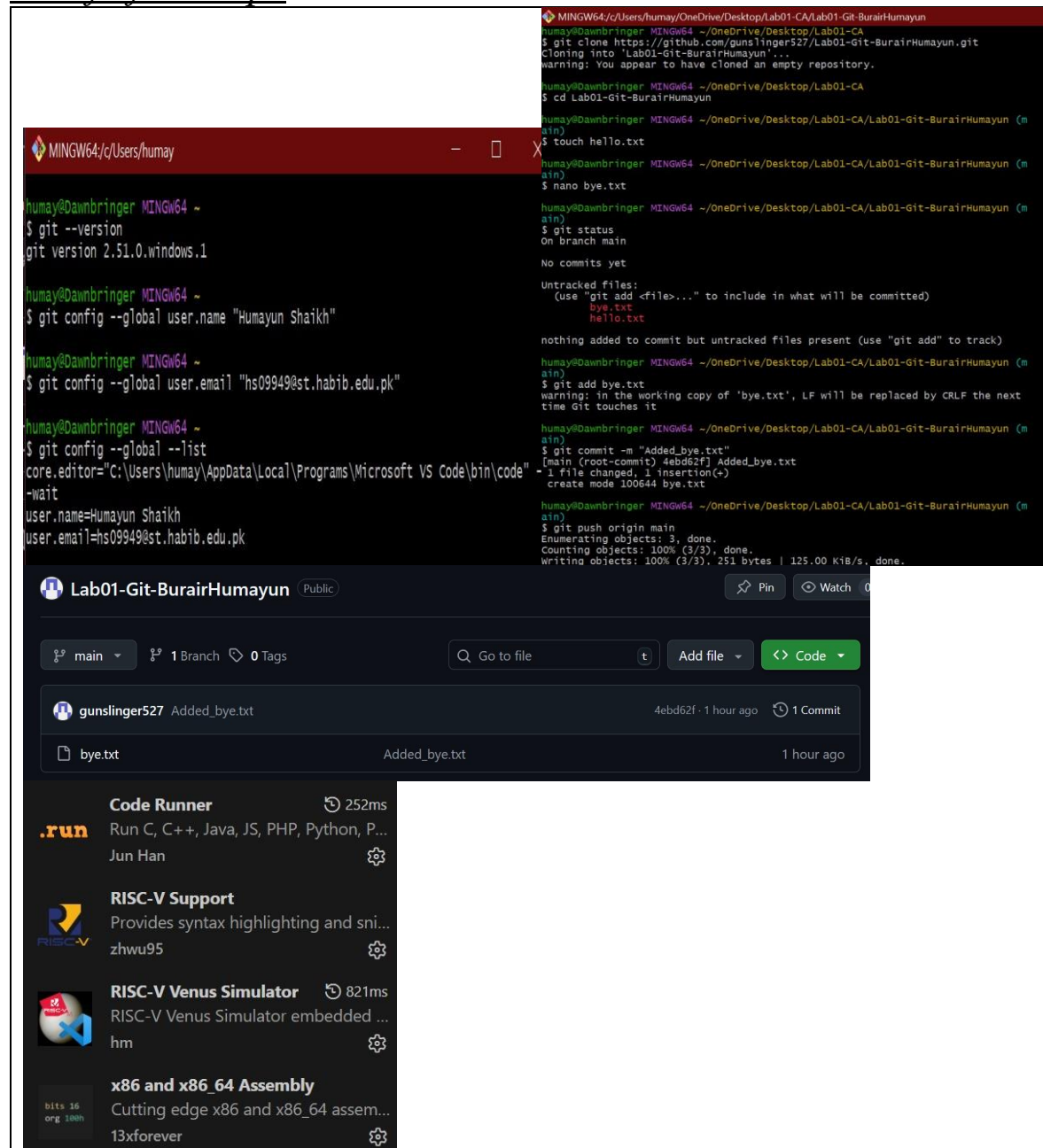


By: Humayun Shaikh and Burair Haidar

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

### Task 1:

#### Proof of all steps



The screenshot displays a Windows terminal window and a GitHub repository interface. The terminal window, titled "MINGW64/c/Users/humay/OneDrive/Desktop/Lab01-CA/Lab01-Git-BurairHumayun", shows the following commands and output:

```
humay@Dawnbringer MINGW64 ~
$ git --version
git version 2.51.0.windows.1

humay@Dawnbringer MINGW64 ~
$ git config --global user.name "Humayun Shaikh"

humay@Dawnbringer MINGW64 ~
$ git config --global user.email "hs09949@st.habib.edu.pk"

humay@Dawnbringer MINGW64 ~
$ git config --global --list
core.editor="C:\Users\humay\AppData\Local\Programs\Microsoft VS Code\bin\code"
user.name=Humayun Shaikh
user.email=hs09949@st.habib.edu.pk
```

The GitHub repository, titled "Lab01-Git-BurairHumayun", shows the following commit history:

- gunslinger527 Added\_bye.txt 4ebd62f · 1 hour ago 1 Commit

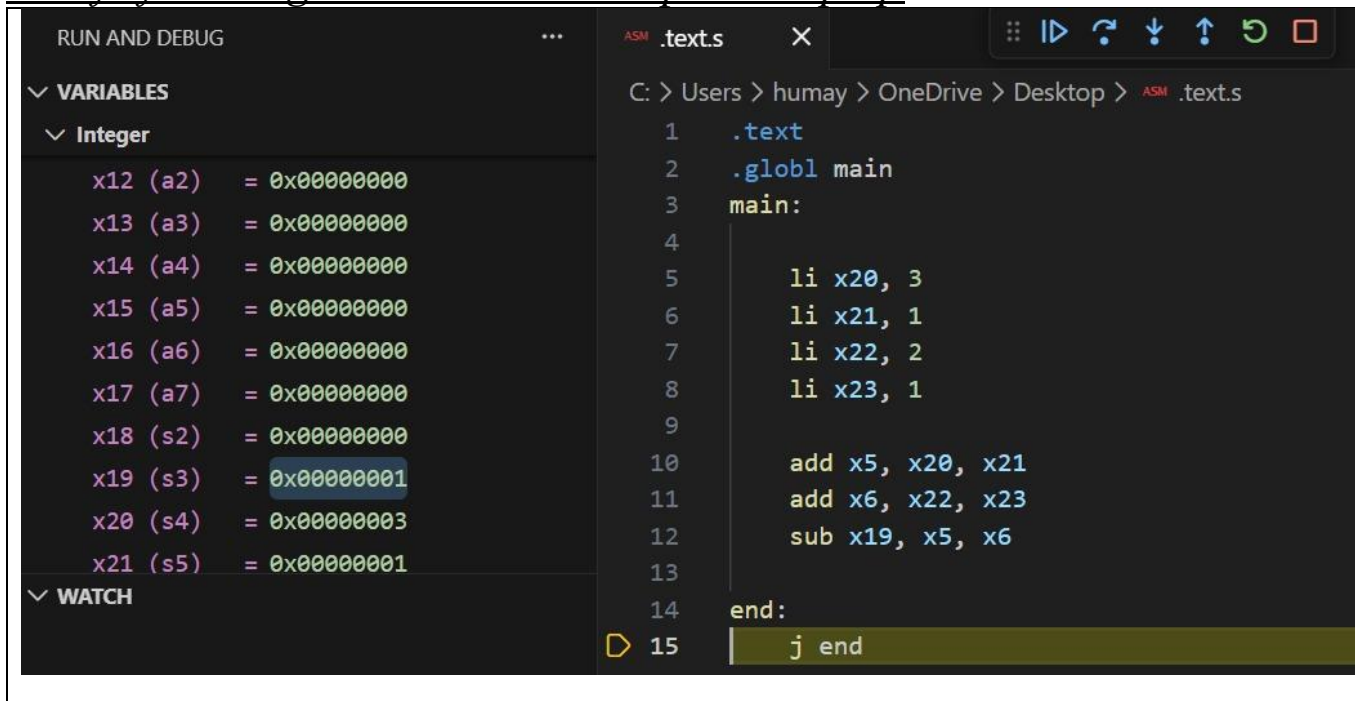
The repository also shows a file named "bye.txt" with the commit message "Added\_bye.txt".

Below the repository interface, there is a sidebar with several extensions and tools:

- Code Runner**: Run C, C++, Java, JS, PHP, Python, P... (252ms)
- RISC-V Support**: Provides syntax highlighting and sni... (zhuw95)
- RISC-V Venus Simulator**: RISC-V Venus Simulator embedded ... (821ms)
- x86 and x86\_64 Assembly**: Cutting edge x86 and x86\_64 assem... (13xforever)

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

*Proof of running the manual's example on laptop*



The screenshot shows the VS Code interface with the 'RUN AND DEBUG' sidebar on the left and the '.text.s' assembly file in the main editor. The assembly code is as follows:

```
1 .text
2 .globl main
3 main:
4
5     li x20, 3
6     li x21, 1
7     li x22, 2
8     li x23, 1
9
10    add x5, x20, x21
11    add x6, x22, x23
12    sub x19, x5, x6
13
14 end:
15 j end
```

The 'VARIABLES' section in the sidebar shows the following register values:

Register	Value
x12 (a2)	0x00000000
x13 (a3)	0x00000000
x14 (a4)	0x00000000
x15 (a5)	0x00000000
x16 (a6)	0x00000000
x17 (a7)	0x00000000
x18 (s2)	0x00000000
x19 (s3)	0x00000001
x20 (s4)	0x00000003
x21 (s5)	0x00000001

The 'WATCH' section is currently empty.

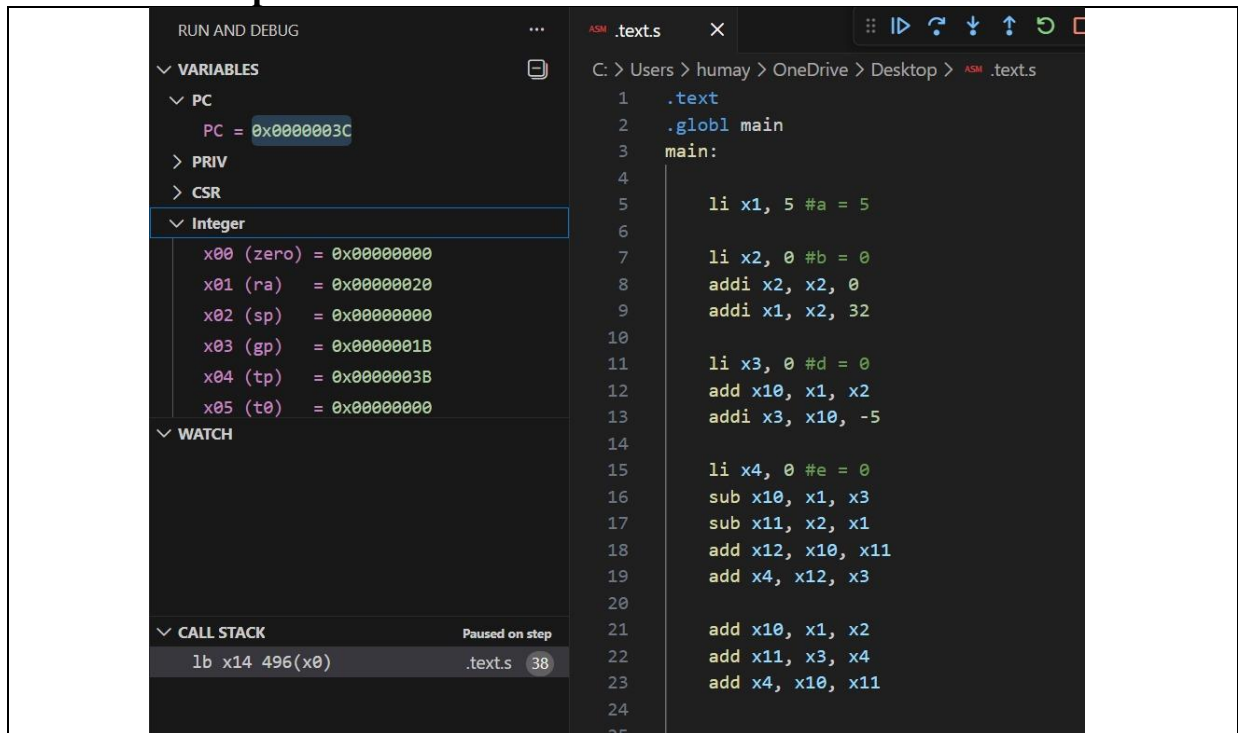
Figure 1.4: RISC-V Memory

### Task 3

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

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

### Code AND Output



The screenshot displays a RISC-V IDE interface with the following components:

- Variables Panel:**
  - PC:** 0x0000003C
  - Integer Registers:**
    - x00 (zero) = 0x00000000
    - x01 (ra) = 0x00000020
    - x02 (sp) = 0x00000000
    - x03 (gp) = 0x0000001B
    - x04 (tp) = 0x0000003B
    - x05 (t0) = 0x00000000
- Watch Panel:** Empty.
- Call Stack:** Paused on step 38, showing 'lb x14 496(x0)' from '.text.s'.
- Assembly Code Editor:**

```
1 .text
2 .globl main
3 main:
4
5     li x1, 5 #a = 5
6
7     li x2, 0 #b = 0
8     addi x2, x2, 0
9     addi x1, x2, 32
10
11    li x3, 0 #d = 0
12    add x10, x1, x2
13    addi x3, x10, -5
14
15    li x4, 0 #e = 0
16    sub x10, x1, x3
17    sub x11, x2, x1
18    add x12, x10, x11
19    add x4, x12, x3
20
21    add x10, x1, x2
22    add x11, x3, x4
23    add x4, x10, x11
24
25
```



## 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:

Store x10 as unsigned integer at address 0x100.

x10 (a0)	= 0x78786464	li x10, 0x78786464	Address	+0	+1	+2	+3
		sw x10, 0x100(x0)	0x00000100	64	64	78	78

Store x11 as unsigned integer at address 0x1F0.

x11 (a1)	= 0xA8A81919	li x11, 0xA8A81919	Address	+0	+1	+2	+3
		sw x11, 0x1F0(x0)	0x000001F0	19	19	A8	A8

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

x12 (a2)	= 0x00006464	lhu x12, 0x100(x0)	Address	+0	+1	+2	+3
	41		0x00000100	64	64	78	78

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

x13 (a3)	= 0x00001919	lh x13, 0x1F0(x0)	Address	+0	+1	+2	+3
	43		0x000001F0	19	19	A8	A8

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

x14 (a4)	= 0x00000019	lb x14, 0x1F0(x0)	Address	+0	+1	+2	+3
	45		0x000001F0	19	19	A8	A8

✻

### Task 4b – - Loop unrolling

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

```
for (int i=0 ; i<4; i++ )
c [ i ]=a [ i ]+b [ i ] ; # c [ 0 ]=a [ 0 ]+b [ 0 ] ;
```

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 AND Output

**VARIABLES**

- Integer
  - x01 (ra) = 0x00000050
  - x02 (sp) = 0x7FFFFFF0
  - x03 (gp) = 0x10000000
  - x04 (tp) = 0x00000000
  - x05 (t0) = 0x00000100
  - x06 (t1) = 0x00000200
  - x07 (t2) = 0x00000300
  - x08 (s0) = 0x00000000
  - x09 (s1) = 0x00000000

**WATCH**

- jal x0 0

**CALL STACK** Paused on step

- jal x0 0 .text.s 79

**C: > Users > hamy > OneDrive > Desktop > ASM .text.s**

```

3      main:
46
47      li x5, 0x100
48      li x6, 0x200
49      li x7, 0x300
50      # when i = 0
51      lb x28, 0(x5)
52      lh x29, 0(x6)
53      add x30, x28, x29
54      sw x30, 0(x7)
55      # when i = 1
56      lb x28, 1(x5)
57      lh x29, 2(x6)
58      add x30, x28, x29
59      sw x30, 4(x7)
60      # when i = 2
61      lb x28, 2(x5)
62      lh x29, 4(x6)
63      add x30, x28, x29
64      sw x30, 8(x7)
65      # when i = 3
66      lb x28, 3(x5)
67      lh x29, 6(x6)
68      add x30, x28, x29
69      sw x30, 12(x7)
  
```



## Assessment Rubric

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

<b>Name:</b>	<b>Student ID:</b>	<b>section*:</b>
--------------	--------------------	------------------

#### 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.*

