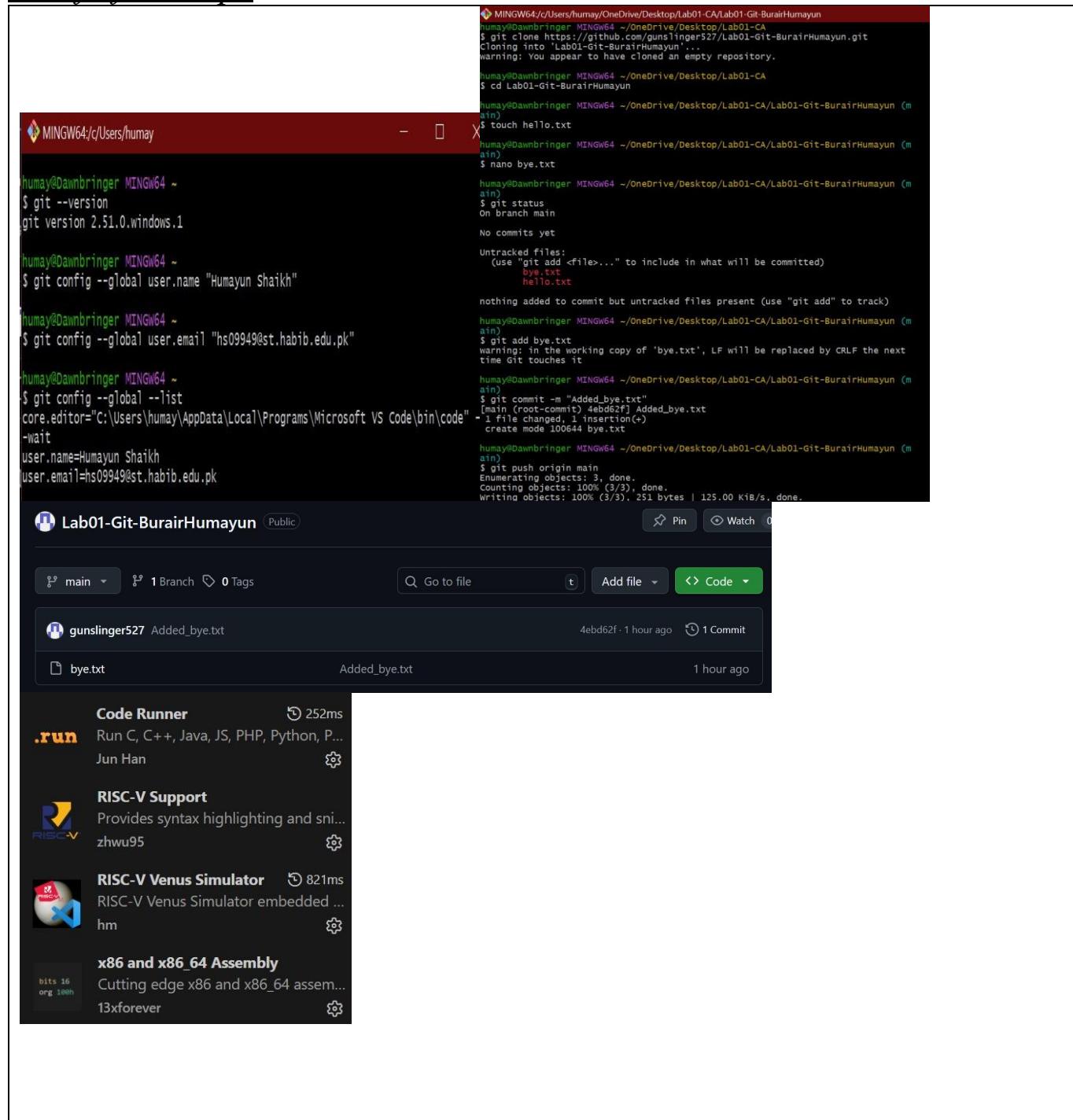


By: Humayun Shaikh and Burair Haidar

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

Task 1:

Proof of all steps



The screenshot shows a GitHub repository named "Lab01-Git-BurairHumayun". The repository has one branch ("main") and no tags. The commit history shows a single commit by "gunslinger527" titled "Added_bye.txt" made 1 hour ago. The commit message indicates the addition of files "bye.txt" and "Added_bye.txt". Below the commit history, there are several extensions listed for the Code Runner, RISC-V Support, RISC-V Venus Simulator, and x86 and x86_64 Assembly.

```
MINGW64:/c/Users/humay
humay@Dawnbringer MINGW64 ~/OneDrive/Desktop/Lab01-CA/Lab01-Git-BurairHumayun
$ git clone https://github.com/gunslinger527/Lab01-Git-BurairHumayun.git
Cloning into 'Lab01-Git-BurairHumayun'...
warning: You appear to have cloned an empty repository.

humay@Dawnbringer MINGW64 ~/OneDrive/Desktop/Lab01-CA/Lab01-Git-BurairHumayun
$ cd Lab01-Git-BurairHumayun

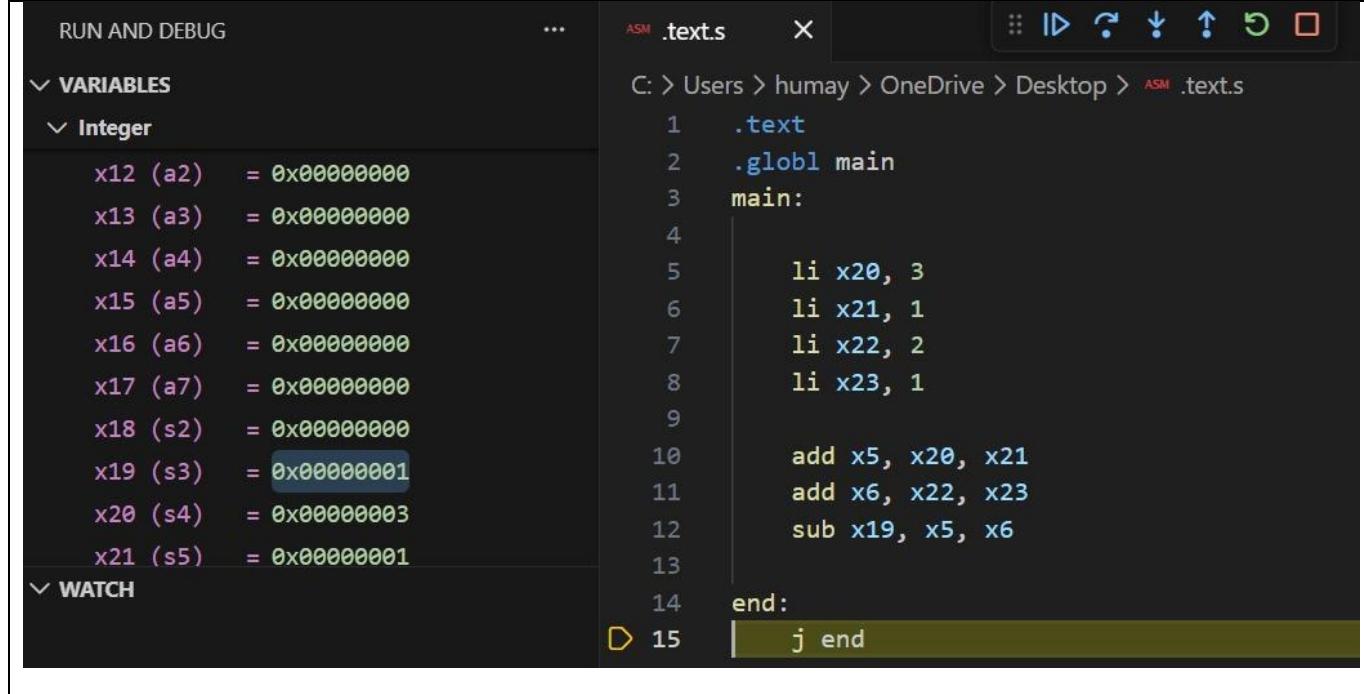
humay@Dawnbringer MINGW64 ~/OneDrive/Desktop/Lab01-CA/Lab01-Git-BurairHumayun (main)
$ touch hello.txt
humay@Dawnbringer MINGW64 ~/OneDrive/Desktop/Lab01-CA/Lab01-Git-BurairHumayun (main)
$ nano bye.txt
humay@Dawnbringer MINGW64 ~/OneDrive/Desktop/Lab01-CA/Lab01-Git-BurairHumayun (main)
$ git status
On branch main
No commits yet
Untracked files:
  (use "git add <file>" to include in what will be committed)
    bye.txt
    hello.txt
nothing added to commit but untracked files present (use "git add" to track)

humay@Dawnbringer MINGW64 ~/OneDrive/Desktop/Lab01-CA/Lab01-Git-BurairHumayun (main)
$ git add bye.txt
warning: in the working copy of 'bye.txt', LF will be replaced by CRLF the next time Git touches it
humay@Dawnbringer MINGW64 ~/OneDrive/Desktop/Lab01-CA/Lab01-Git-BurairHumayun (main)
$ git commit -m "Added_bye.txt"
[main (root-commit) 4ebd62f] Added_bye.txt
- 1 file changed, 1 insertion(+)
  create mode 100644 bye.txt
humay@Dawnbringer MINGW64 ~/OneDrive/Desktop/Lab01-CA/Lab01-Git-BurairHumayun (main)
$ git push origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 251 bytes | 125.00 KiB/s, done.

Lab01-Git-BurairHumayun (Public)
main 1 Branch 0 Tags
Go to file Add file Code
gunslinger527 Added_bye.txt 4ebd62f · 1 hour ago 1 Commit
bye.txt Added_bye.txt 1 hour ago
Code Runner .run 252ms
Run C, C++, Java, JS, PHP, Python, P... Jun Han
RISC-V Support RISC-V Provides syntax highlighting and sni... zhwu95
RISC-V Venus Simulator RISC-V Venus Simulator embedded ... hm
x86 and x86_64 Assembly bits 32 821ms
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 RISC-V assembly file `.text.s` open. The left sidebar displays a list of variables and their integer values:

Variable	Type	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 assembly code in the editor 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
```

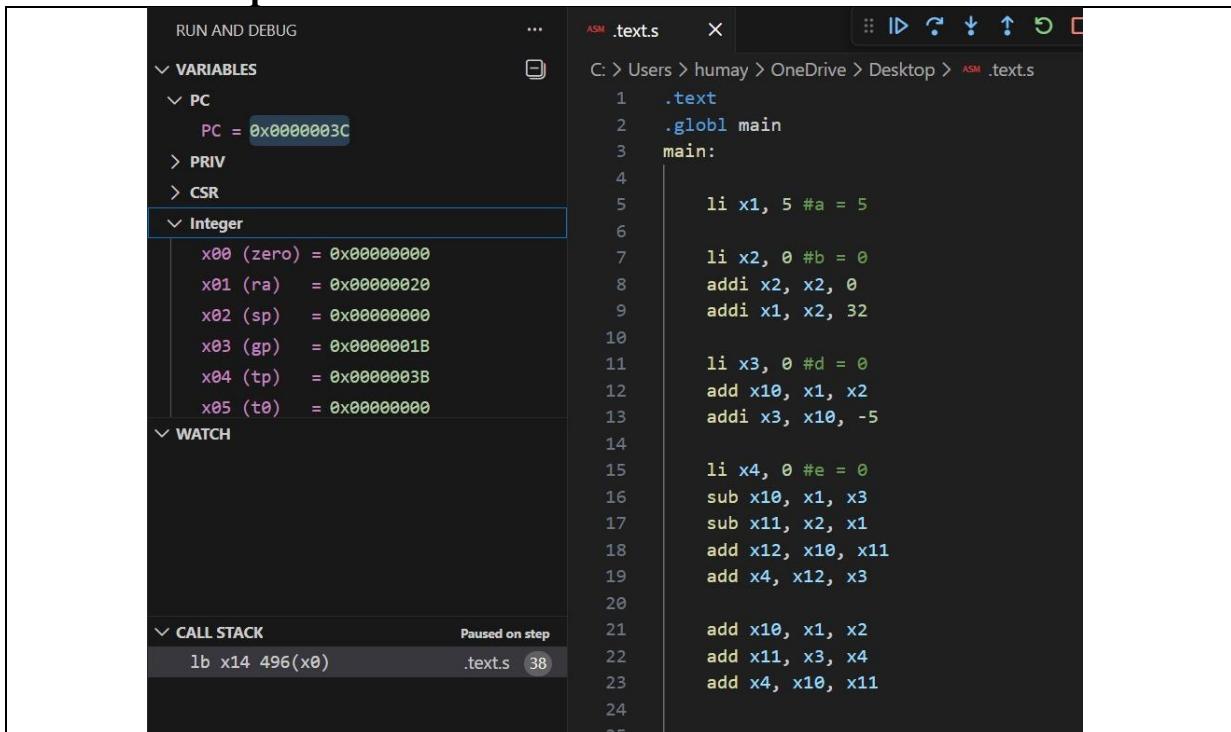
Figure 1.4: RISC-V Memory

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 shows a debugger interface with two main panes. The left pane is titled 'RUN AND DEBUG' and contains a 'VARIABLES' section. Under 'VARIABLES', there is a 'PC' entry with the value `0x0000003C`. Below it is a 'CSR' section. Under 'CSR', there is an 'Integer' section containing the following register values:

x00 (zero)	= 0x00000000
x01 (ra)	= 0x00000020
x02 (sp)	= 0x00000000
x03 (gp)	= 0x0000001B
x04 (tp)	= 0x0000003B
x05 (t0)	= 0x00000000

The right pane is titled 'ASM .text.s' and shows the assembly code for the 'main' function:

```
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	sw x10, 0x100(x0)	Address 0x00000100	+0 64	+1 64	+2 78	+3 78
-----------------------	--------------------	-------------------	--------------------	-------	-------	-------	-------

Store x11 as unsigned integer at address 0x1F0.

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

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

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

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

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

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

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

*

Task 4b -- Loop unrolling

```

1
2 Assume there are three character arrays a, b, and c located at addresses 0
   x100, 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 AND Output

<div style="background-color: black; color: white; padding: 5px; font-size: 0.9em;">VARIABLES</div> <div style="background-color: black; color: white; padding: 5px; font-size: 0.9em;">Integer</div> <pre> x00 (ra) = 0x00000000 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 </pre> <div style="background-color: black; color: white; padding: 5px; font-size: 0.9em;">WATCH</div> <div style="background-color: black; color: white; padding: 5px; font-size: 0.9em;">CALL STACK</div> <div style="background-color: black; color: white; padding: 5px; font-size: 0.9em;">Paused on step</div> <pre> jal x0 0 .text.s 79 </pre>	<div style="background-color: black; color: white; padding: 5px; font-size: 0.9em;">C: > Users > humay > OneDrive > Desktop > ASM .text.s</div> <pre> 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) </pre>
---	---



Assessment Rubric

Lab 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.

