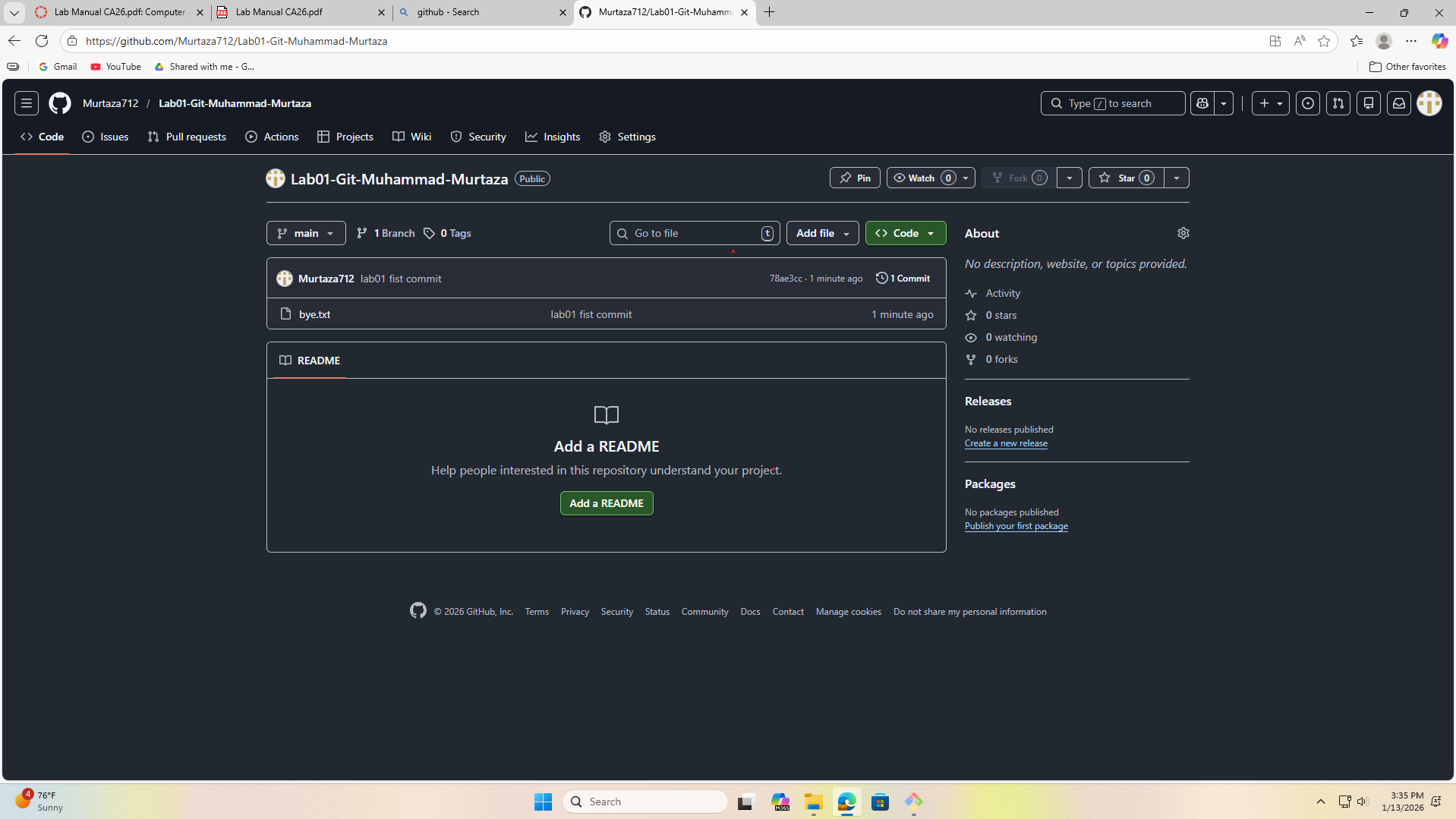
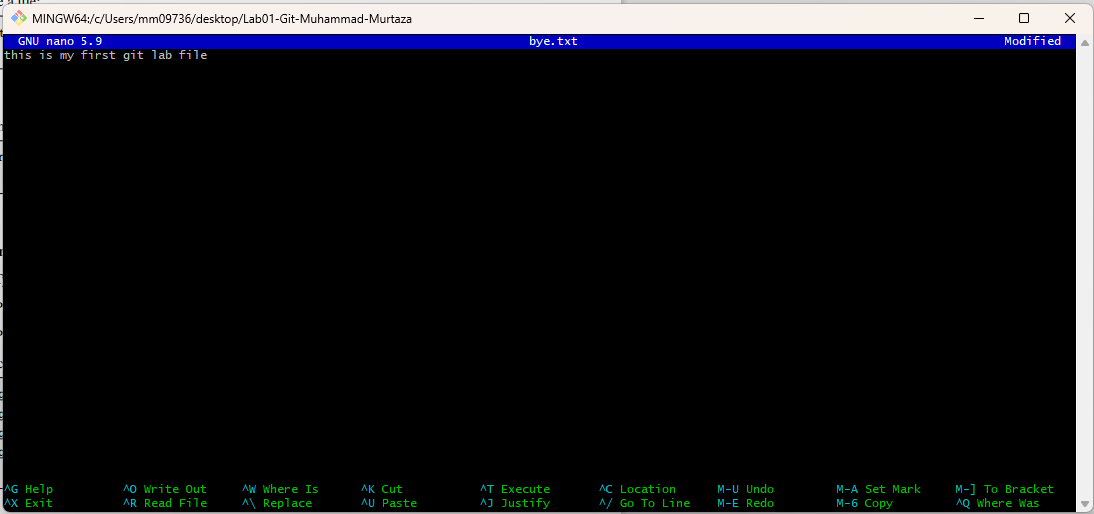
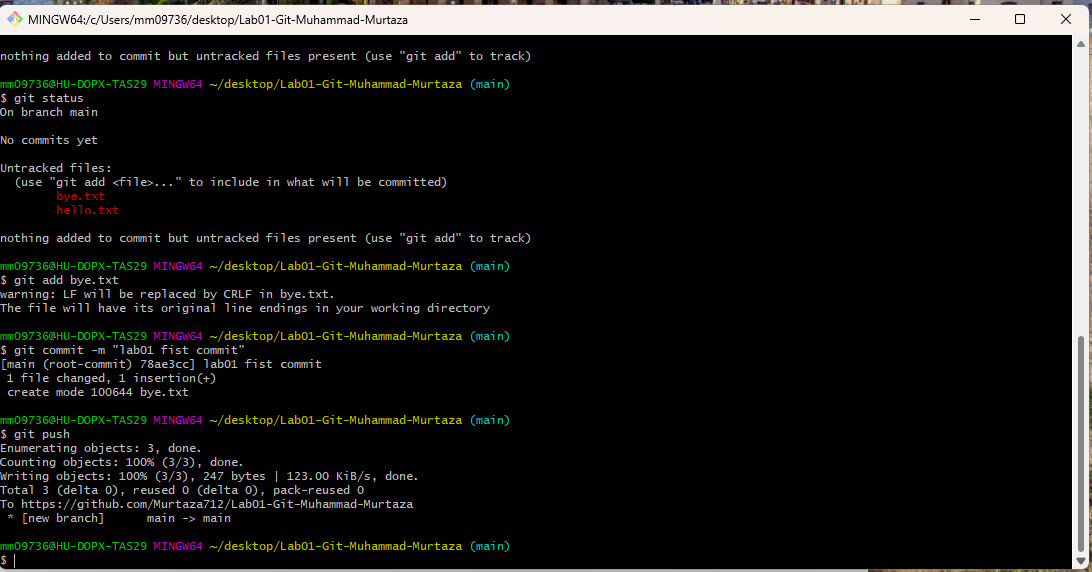
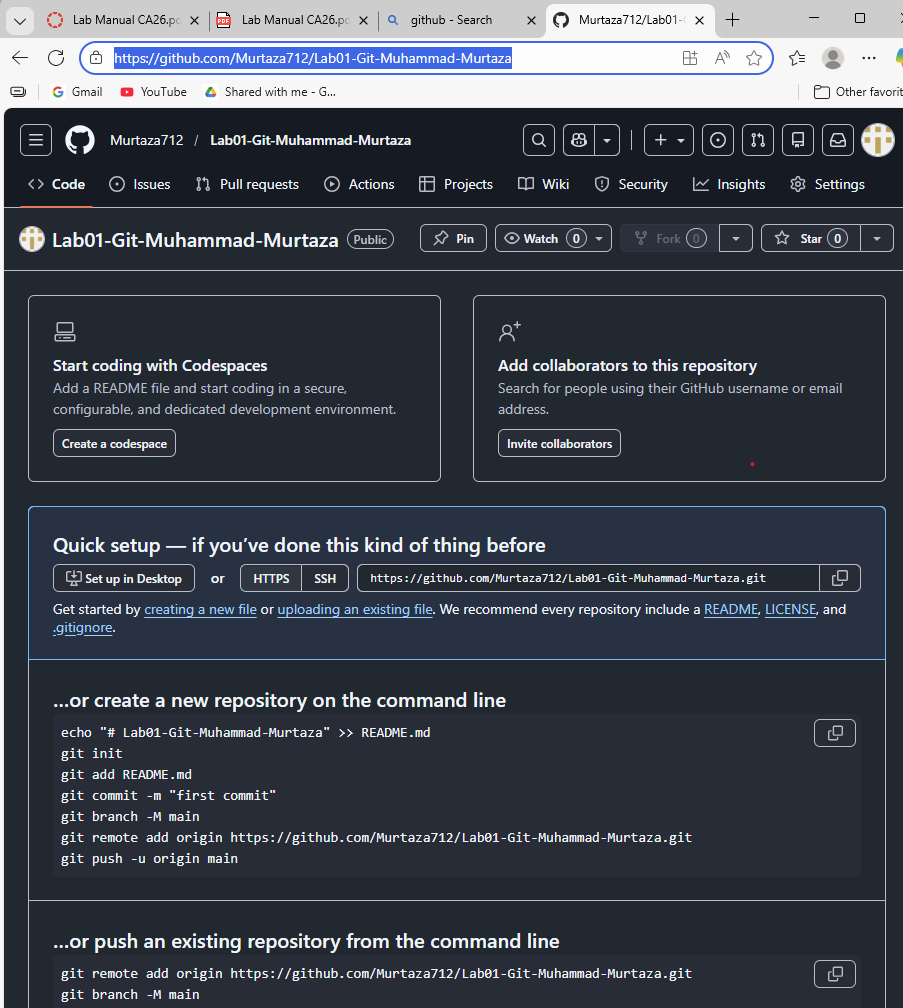
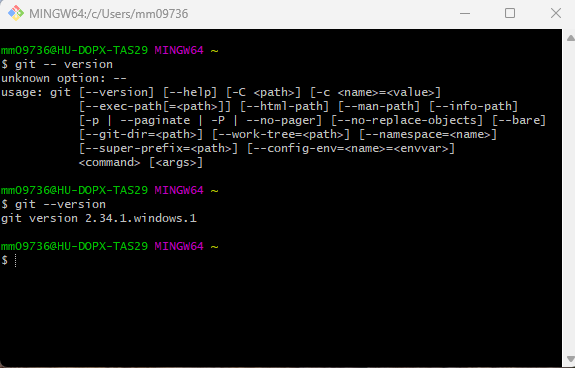
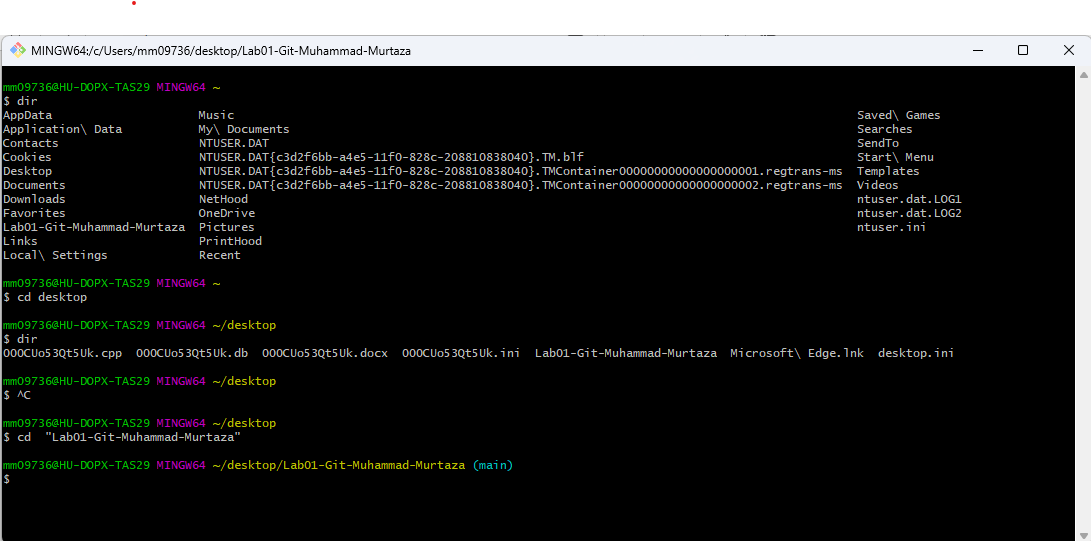
# Lab 1: Getting Started with RISC-V (Assembly Language) in VS Code

# Task 1:

­

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

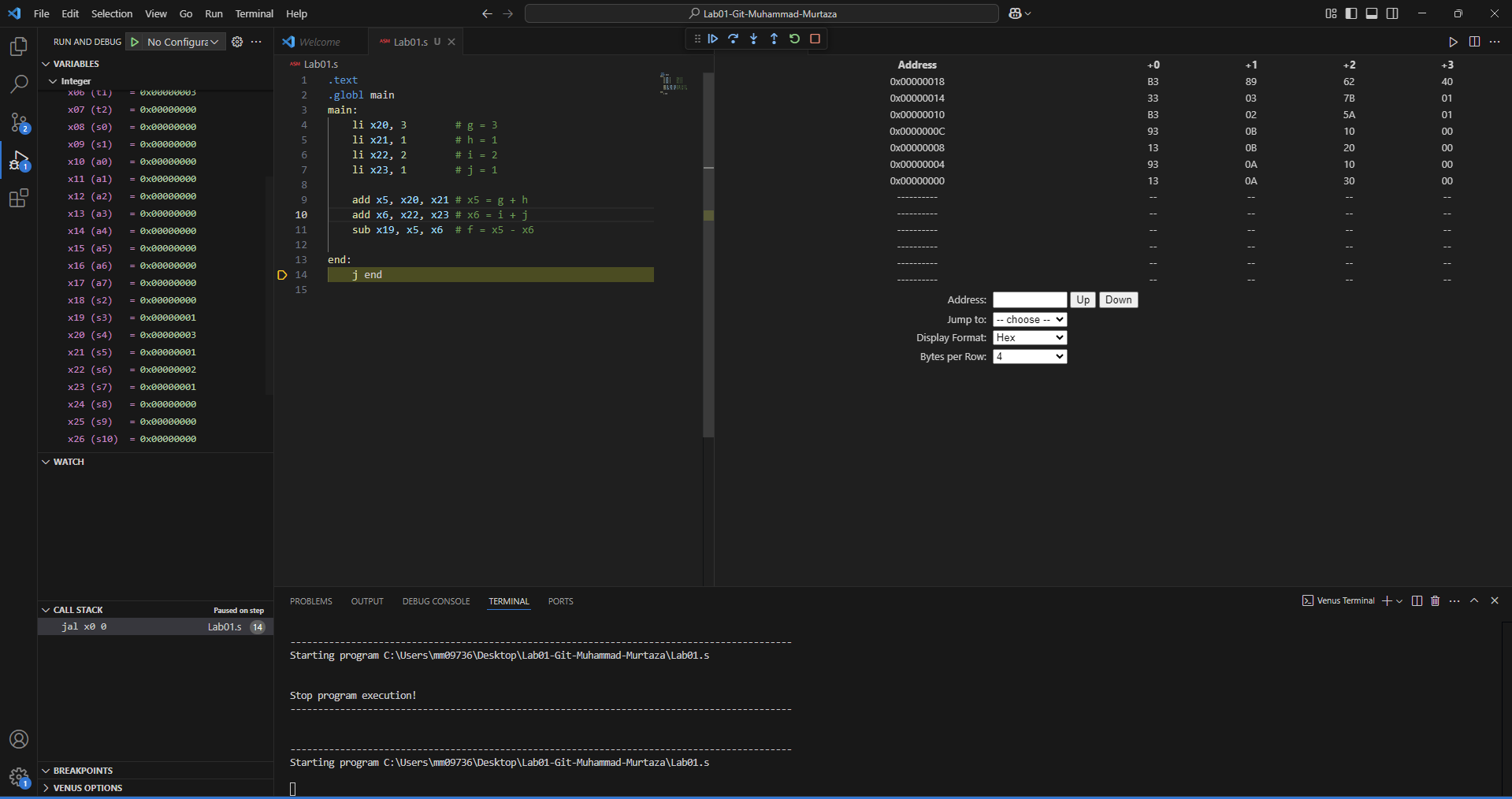
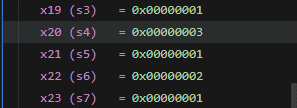
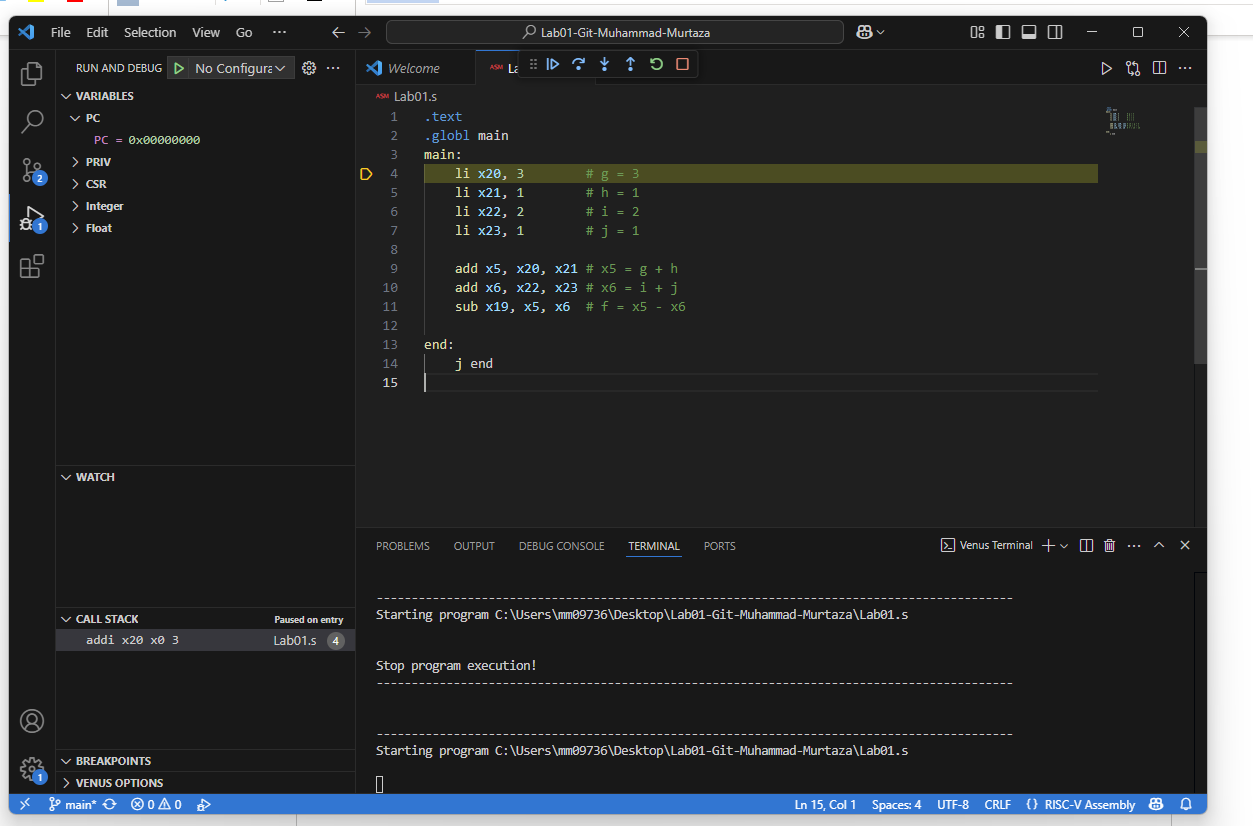
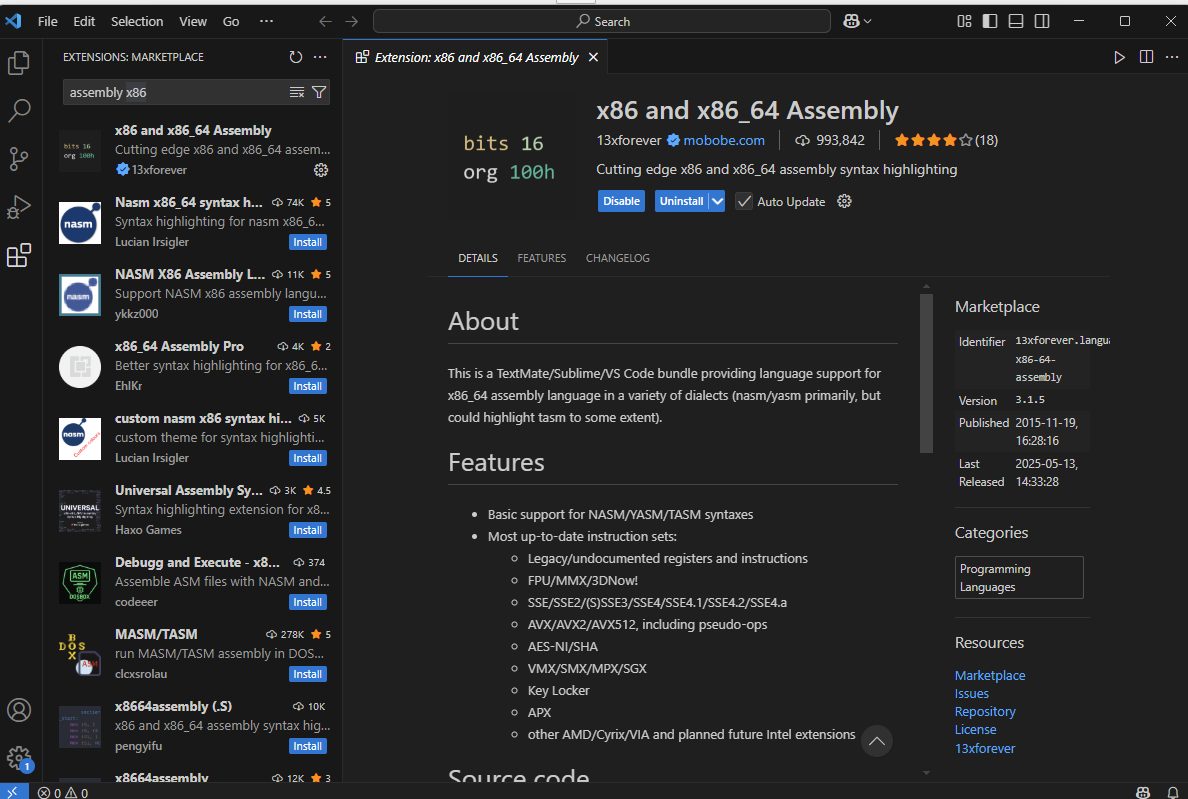
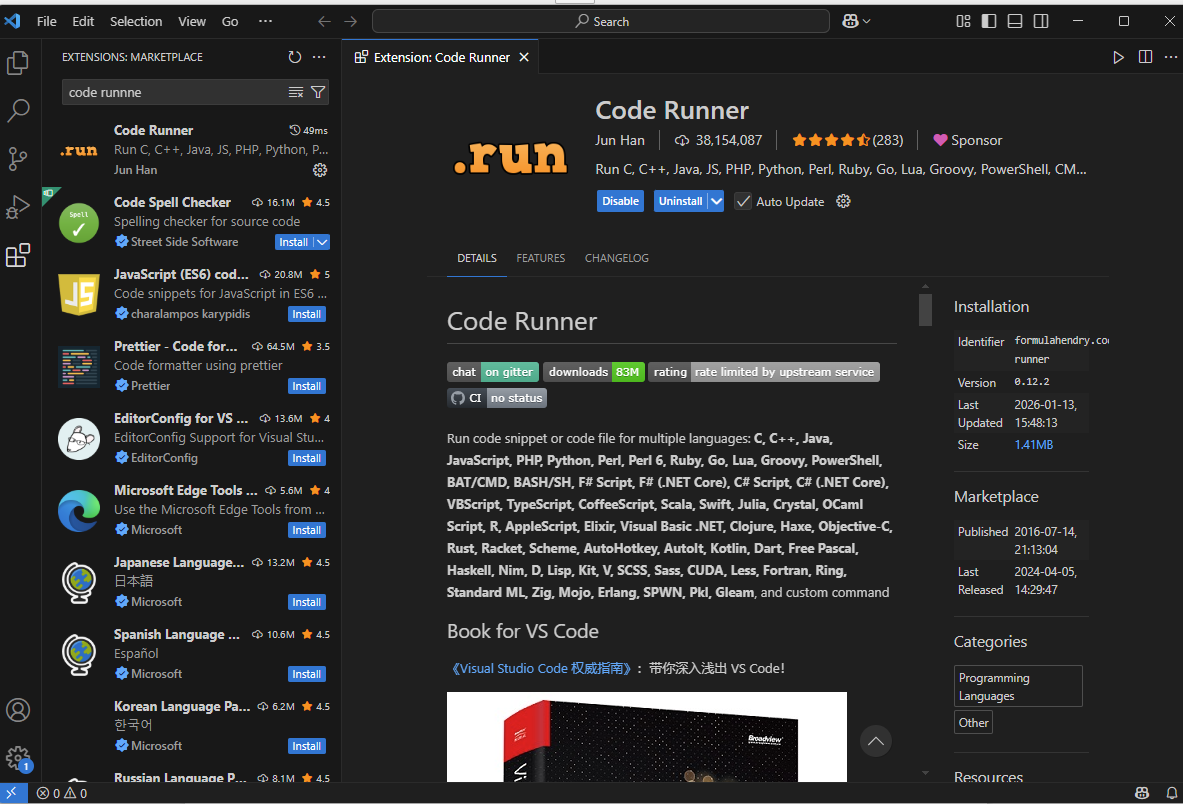
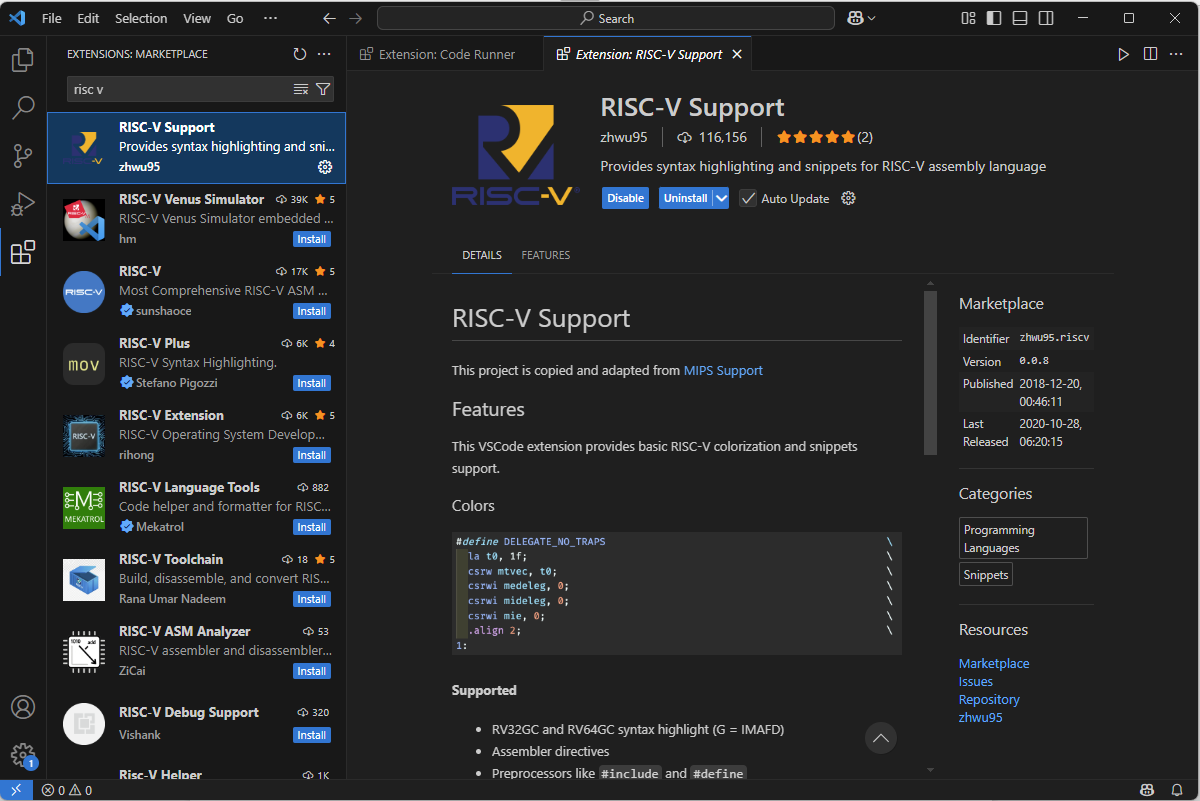
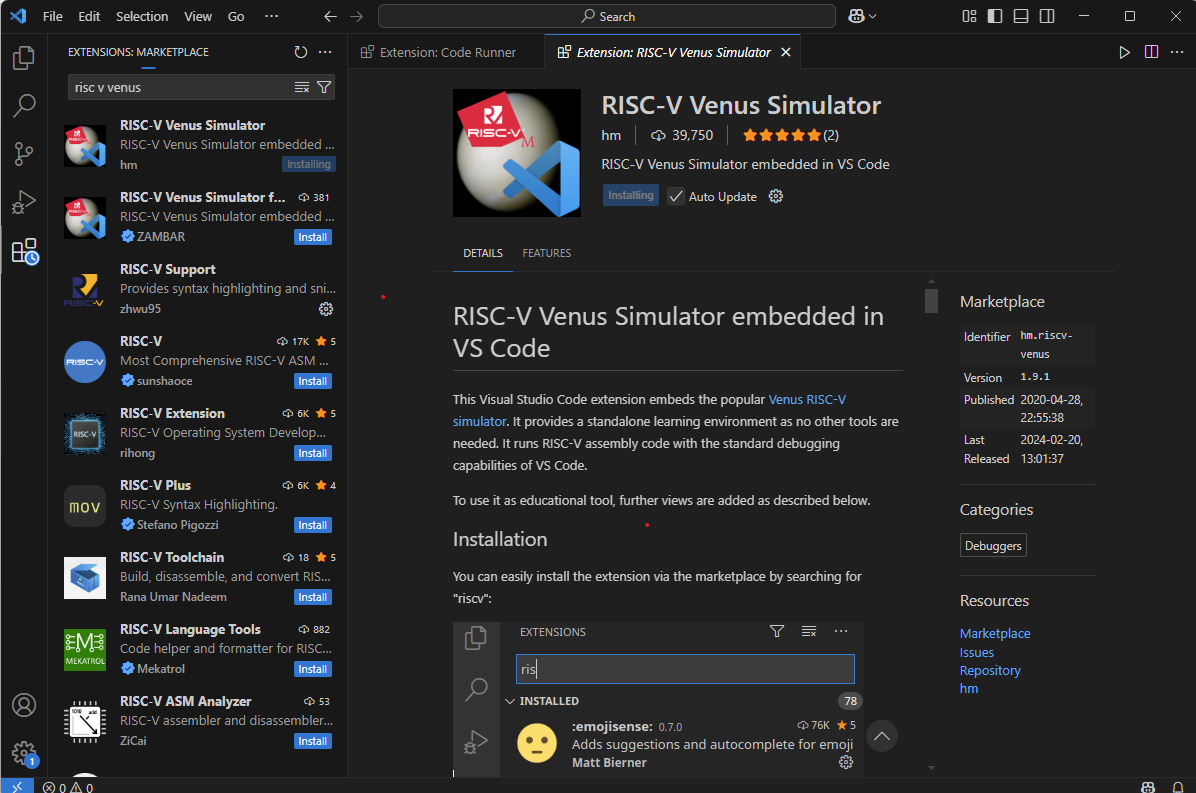


Figure 1.4: RISC-V Memory

## Task 3

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

1

int a = 5;

int b = 0 + 0; a = b + 32;

int d = (a + b) - 5;

int e = (((a - d) + (b - a)) + d); e = a + b + d + e;

2

3

4

5

## Task 3

## Code

.text

.globl main

main:

    # x20=a, x21=b, x22=d, x23=e, x5= temp1, x6=temp2

    li x20, 5 #a=5

    li x21, 0 #b=0

    addi x20, x21, 32 #a = b+32

    ##########################

    add x22, x20, x21 #d=a+b

    addi x22, x22, -5 #d-5

    ##########################

    sub x5, x20,x22 #temp1=a-d

    sub x6, x21, x20 #temp2=b-a

    add x23, x5, x6 #e=temp1+temp2

    add x23, x23, x22 #e = (temp1+temp2)+d

    ##########################

    add x23, x23, x20 #a+e

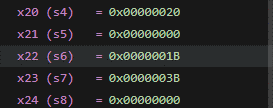
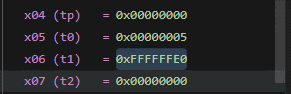
    add x23, x23, x21 #a+b+e

    add x23, x23, x22 #a+b+d+e, e already in x23, we dont need to add it again. we add in the same location so this translate to the same thing

end:

    j end

## Output



## 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.

    li x10 0x78786464

    sw x10, 0x100(x0)



Store x11 as unsigned integer at address 0x1F0.

    li x11 0xA8A81919

    sw x11, 0x1F0(x0)



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

  lhu x12, 0x100(x0)



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

    lh x13, 0x1F0(x0)



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

    lb x14, 0x1F0(x0)



**\***

Task 4b – - Loop unrolling

1

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

2

3

4

5

6

7

## Code

## Output

# 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.*](#_bookmark22)