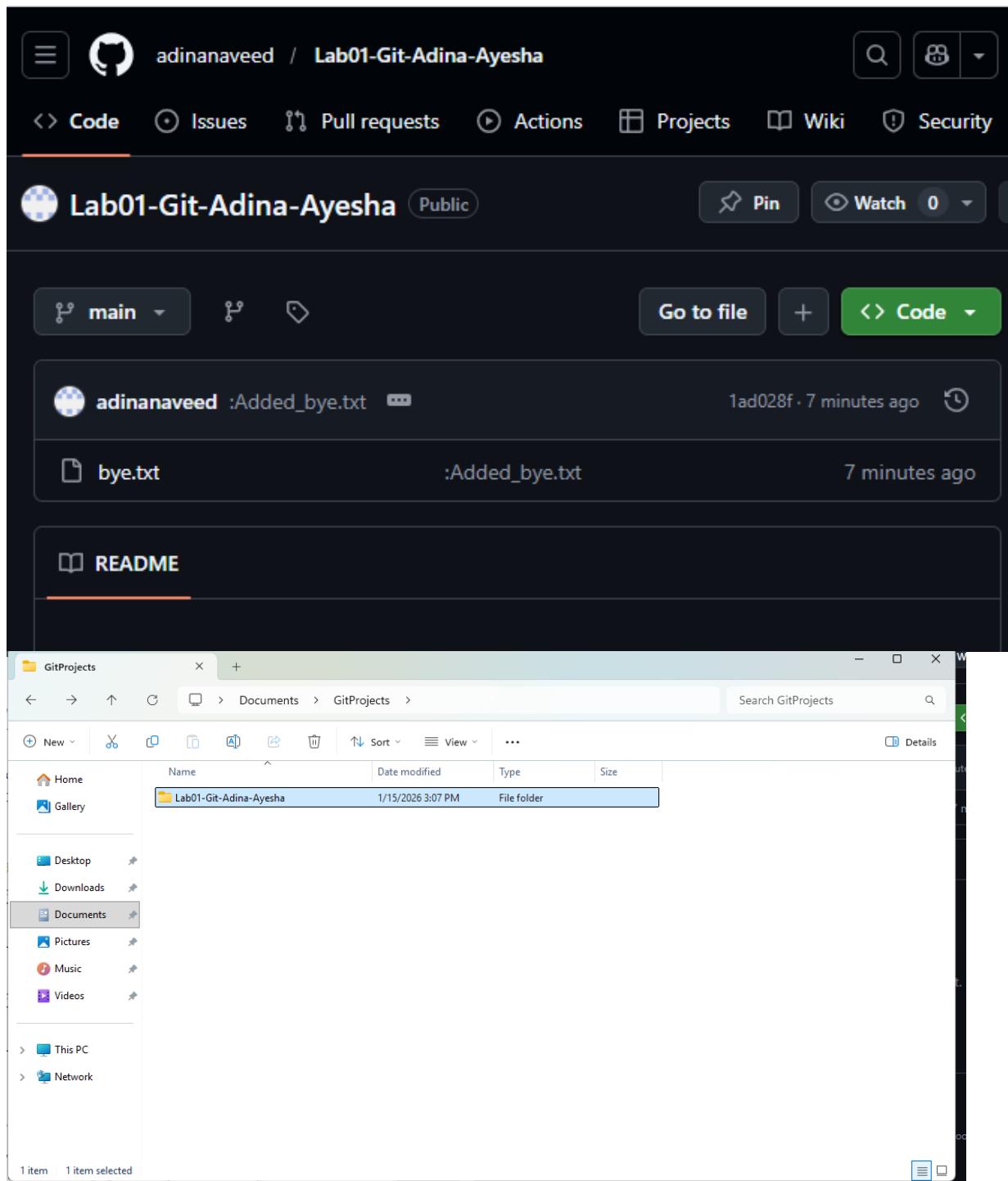


Lab 1: Getting Started with RISC-V (Assembly Language) in VS Code

Task 1:



The image shows two screenshots side-by-side. The top screenshot is a GitHub repository page for 'Lab01-Git-Adina-Ayesha'. It displays a list of files: 'Added_bye.txt' and 'bye.txt'. The bottom screenshot is a Windows File Explorer window showing the same folder structure.

GitHub Repository Screenshot:

- Repository: adinanaveed / Lab01-Git-Adina-Ayesha
- Branch: main
- Files:
 - Added_bye.txt (added 7 minutes ago)
 - bye.txt (added 7 minutes ago)
- README file

Windows File Explorer Screenshot:

- Path: Documents > GitProjects
- Selected folder: Lab01-Git-Adina-Ayesha
- File list:
 - Home
 - Gallery
 - Desktop
 - Downloads
 - Documents
 - Pictures
 - Music
 - Videos
 - This PC
 - Network



The screenshot shows a Windows File Explorer window. The title bar says "Lab01-Git-Adina-Ayesha". The address bar shows the path: "Documents > GitProjects > Lab01-Git-Adina-Ayesha". A search bar at the top right contains the text "Search Lab01-Git-Adina-Ayesha". The main area displays a list of files in the "Lab01-Git-Adina-Ayesha" folder:

Name	Date modified	Type	Size
bye	1/15/2026 3:07 PM	Text Document	1 KB
hello	1/15/2026 3:06 PM	Text Document	0 KB

The left sidebar shows a navigation tree with the following structure:

- Home
- Gallery
- Desktop
- Downloads
- Documents** (selected)
- Pictures
- Music
- Videos

Below the sidebar, there are links to "This PC" and "Network". At the bottom left, it says "2 items". On the bottom right, there are icons for "List" and "Details" view.



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

Opened memory through command:

Typed code with register:

RUN AND DEBUG ▶ Launch with :v ...

VARIABLES

- PC = 0x0000001C
- PRIV
- CSR
- Integer
 - x00 (zero) = 0x00000000
 - x01 (ra) = 0x00000020
 - x02 (sp) = 0x7FFFFFFF
 - x03 (gp) = 0x10000000
 - x04 (tp) = 0x00000000
 - x05 (t0) = 0x00000004
 - x06 (t1) = 0x00000003
 - x07 (t2) = 0x00000000
 - x08 (s0) = 0x00000000
 - 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 (\$2) = 0x00000000
 - x19 (\$3) = 0x00000001
 - x20 (\$4) = 0x00000003
 - x21 (\$5) = 0x00000001
 - x22 (\$6) = 0x00000002
 - x23 (\$7) = 0x000000001
 - x24 (\$8) = 0x00000000
 - x25 (\$9) = 0x00000000
 - x26 (\$10) = 0x00000000
 - x27 (\$11) = 0x00000000
 - x28 (\$12) = 0x00000000
 - x29 (\$13) = 0x00000000
 - x30 (\$14) = 0x00000000
 - x31 (\$15) = 0x00000000

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Starting program C:\Users\ai09884\Documents\GitProjects\.vscode\launch.json

AssemblerError: launch.json:4: label visit in the middle of an instruction
// For more information, visit: https://go.microsoft.com/fwlink/?LinkId=830387

Stop program execution!



Figure 1.4: RISC-V Memory

Task 3

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

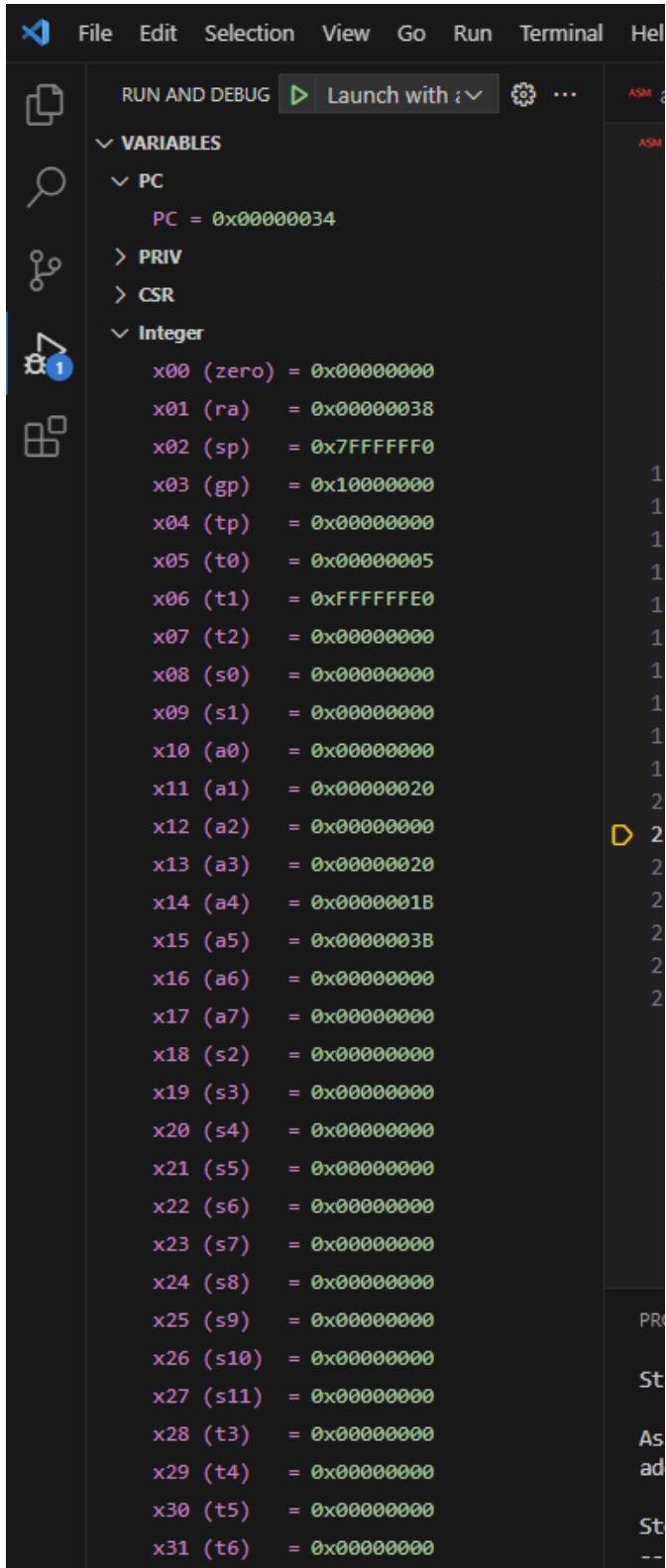
```
1 int a = 5;
2 int b = 0 + 0;
3 a = b + 32;
4 int d = (a + b) - 5;
5 int e = (((a - d) + (b - a)) + d);
6 e = a + b + d + e;
```

Code

```
asm task3.s
1 .text
2 .globl main
3 main:
4     li x11, 5 #a= 5
5     addi x12, x12, 0 #b= 0+0
6     li x13, 32
7     add x11, x13, x12 #a=b+32
8
9     add x14, x11, x12 # d = a+b
10    addi x14, x14, -5 # d = (a+b) - 5
11
12    sub x5, x11, x14 #a-d
13    sub x6, x12, x11 #b-a
14    add x15, x5, x6 #e = (a-d) + (b-a)
15    add x15, x15, x14 # e = ((a-d) + (b-a)) + d
16
17    add x15, x15, x11 # e = a+e
18    add x15, x15, x12 # e = a+e+b
19    add x15, x15, x14 # e = a+e+b+d
20 end:
21     j end
```

Decimal=5 . Hex=5
Dec=0 , Hex=0;
d=27 → Hex = 1B
a = 32 → hex = 20
e = 59 → hex = 3B

Output



The screenshot shows a debugger's variable view window. The title bar includes 'RUN AND DEBUG' and 'Launch with'. The left sidebar has icons for file, search, and connections. The main area is titled 'VARIABLES' and contains a tree view of variables under 'PC', 'PRIV', 'CSR', and 'Integer'. The 'Integer' node is expanded, listing 32 registers (x00 to x31) with their current values in hex format.

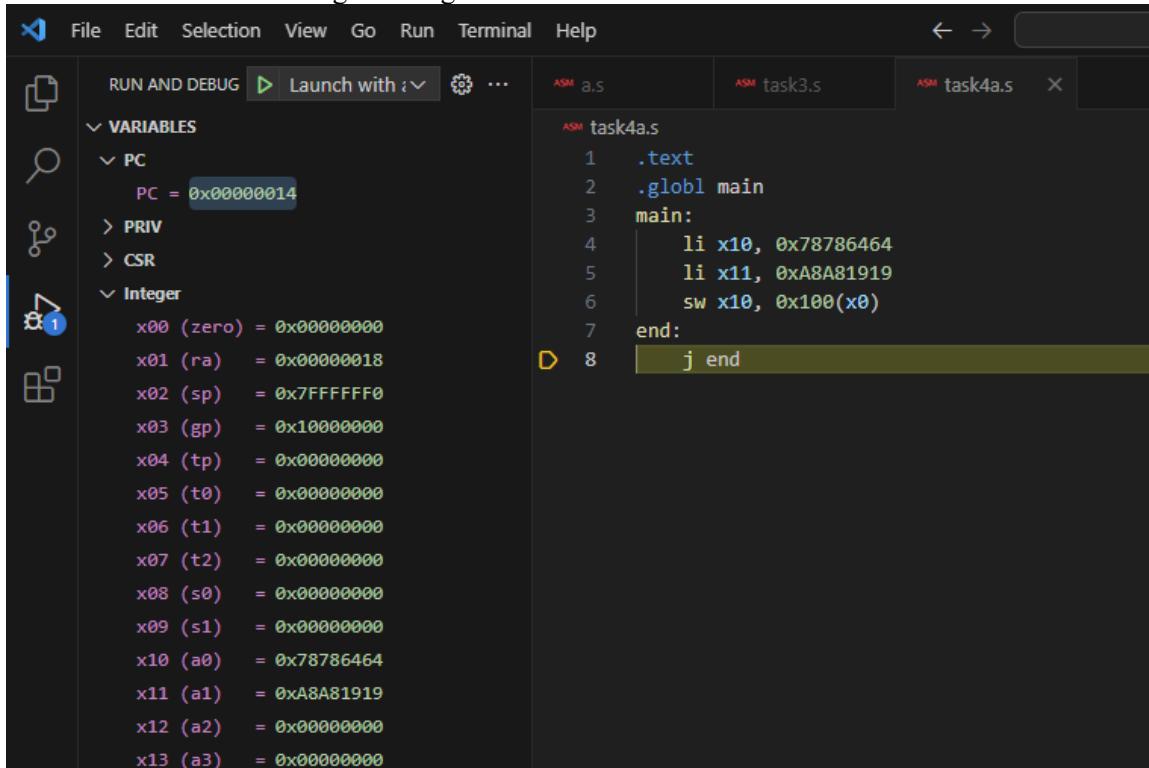
Register	Value	Type
x00 (zero)	0x00000000	
x01 (ra)	0x00000038	
x02 (sp)	0x7FFFFFFF0	
x03 (gp)	0x10000000	
x04 (tp)	0x00000000	
x05 (t0)	0x00000005	
x06 (t1)	0xFFFFFFF0	
x07 (t2)	0x00000000	
x08 (s0)	0x00000000	
x09 (s1)	0x00000000	
x10 (a0)	0x00000000	
x11 (a1)	0x00000020	
x12 (a2)	0x00000000	
x13 (a3)	0x00000020	
x14 (a4)	0x0000001B	
x15 (a5)	0x0000003B	
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)	0x00000000	
x29 (t4)	0x00000000	
x30 (t5)	0x00000000	
x31 (t6)	0x00000000	

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.

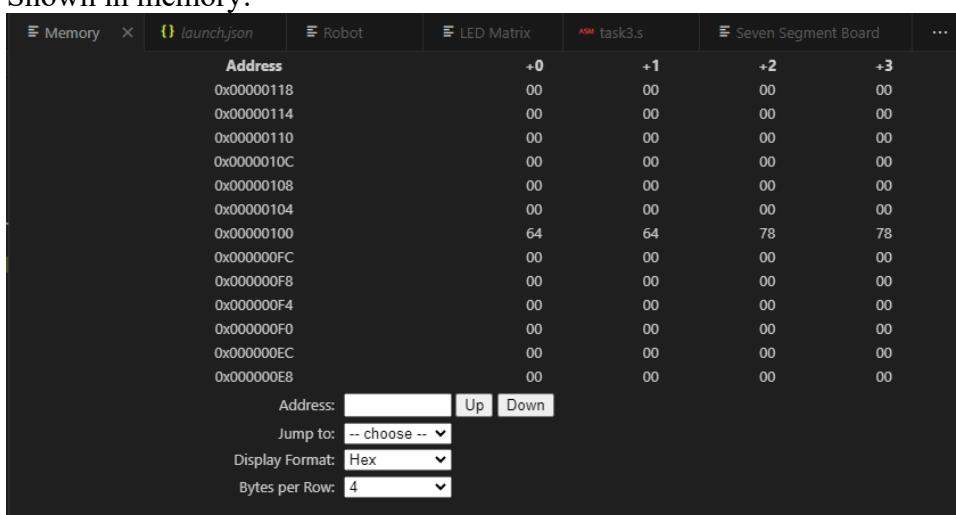


The screenshot shows a debugger interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Left Sidebar:** RUN AND DEBUG, Launch with, ...
Variables section:
 - PC: 0x000000014
 - PRIV
 - CSR
 - Integer
 - x00 (zero) = 0x00000000
 - x01 (ra) = 0x00000018
 - x02 (sp) = 0xFFFFFFF0
 - x03 (gp) = 0x10000000
 - x04 (tp) = 0x00000000
 - x05 (t0) = 0x00000000
 - x06 (t1) = 0x00000000
 - x07 (t2) = 0x00000000
 - x08 (s0) = 0x00000000
 - x09 (s1) = 0x00000000
 - x10 (a0) = 0x78786464
 - x11 (a1) = 0xA8A81919
 - x12 (a2) = 0x00000000
 - x13 (a3) = 0x00000000
- ASM Task4.s:**

```
1 .text
2 .globl main
3 main:
4     li x10, 0x78786464
5     li x11, 0xA8A81919
6     sw x10, 0x100(x0)
7 end:
8     j end
```
- Right Sidebar:** ASM task3.s, ASM task4.s

Shown in memory:



Address	+0	+1	+2	+3
0x00000018	00	00	00	00
0x00000014	00	00	00	00
0x00000010	00	00	00	00
0x00000010C	00	00	00	00
0x000000108	00	00	00	00
0x000000104	00	00	00	00
0x000000100	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

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

Store x11 as unsigned integer at address 0x1F0.

The screenshot shows a debugger interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Tool Buttons:** RUN AND DEBUG, Launch with ..., Variables, Breakpoints, Step Into, Step Over, Step Out, Stop.
- Variables View:**
 - PC:** PC = 0x00000014
 - PRIV:**
 - CSR:**
 - Integer:**
 - x00 (zero) = 0x00000000
 - x01 (ra) = 0x00000018
 - x02 (sp) = 0x7FFFFFF0
 - x03 (gp) = 0x10000000
 - x04 (tp) = 0x00000000
 - x05 (t0) = 0x00000000
 - x06 (t1) = 0x00000000
 - x07 (t2) = 0x00000000
 - x08 (s0) = 0x00000000
 - x09 (s1) = 0x00000000
 - x10 (a0) = 0x78786464
 - x11 (a1) = 0xA8A81919
 - x12 (a2) = 0x00000000
- Assembly View:**

```

1 .text
2 .globl main
3 main:
4     li x10, 0x78786464
5     li x11, 0xA8A81919
6     sw x11, 0x1F0(x0)
7 end:
8     j end

```

Shown in memory:

The memory dump table shows the following data:

Address	+0	+1	+2	+3
0x000000208	00	00	00	00
0x000000204	00	00	00	00
0x000000200	00	00	00	00
0x000001FC	00	00	00	00
0x000001F8	00	00	00	00
0x000001F4	00	00	00	00
0x000001F0	91	91	81	8A
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

Controls at the bottom:

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



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

The screenshot shows a debugger interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Toolbars:** RUN AND DEBUG, Launch with, ... (with a dropdown menu), ASM task4.s, ASM task3.s, ASM task4a.s (active tab), GitProjects.
- Variables View:** Shows the PC at 0x0000001C and the Integer register x12 (a2) highlighted at 0x00006464.
- Assembly Code:**

```
.text
.globl main
main:
    li x10, 0x78786464
    li x11, 0xA8A819191
    sw x10, 0x100(x0)
    sw x11, 0x1F0(x0)
    lhu x12, 0x100(x0)
end:
    j end
```

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

The screenshot shows a debugger interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Toolbars:** RUN AND DEBUG, Launch with, ... (with a dropdown menu), ASM a.s, ASM task3.s, ASM task4a.s (active tab), GitProjects.
- Variables View:** Shows the PC at 0x00000020 and the Integer register x13 (a3) highlighted at 0x00009191.
- Assembly Code:**

```
.text
.globl main
main:
    li x10, 0x78786464
    li x11, 0xA8A819191
    sw x10, 0x100(x0)
    sw x11, 0x1F0(x0)
    lhu x12, 0x100(x0)
    lhu x13, 0x1F0(x0)
end:
    j end
```

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

The screenshot shows a debugger interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Toolbar:** RUN AND DEBUG, Launch with, ...
- Project List:** a.s, task3.s, task4a.s (highlighted).
- Variables View:**
 - PC:** 0x00000024
 - PRIV:**
 - CSR:**
 - Integer:**
 - x00 (zero) = 0x00000000
 - x01 (ra) = 0x00000028
 - x02 (sp) = 0xFFFFFFF0
 - x03 (gp) = 0x10000000
 - x04 (tp) = 0x00000000
 - x05 (t0) = 0x00000000
 - x06 (t1) = 0x00000000
 - x07 (t2) = 0x00000000
 - x08 (s0) = 0x00000000
 - x09 (s1) = 0x00000000
 - x10 (a0) = 0x78786464
 - x11 (a1) = 0x8A819191
 - x12 (a2) = 0x00006464
 - x13 (a3) = 0x00009191
 - x14 (a4)** = 0xFFFFFFF91 (highlighted)
 - x15 (a5) = 0x00000000
- Assembly Code:**

```

1 .text
2 .globl main
3 main:
4 li x10, 0x78786464
5 li x11, 0xA8A819191
6 sw x10, 0x100(x0)
7 sw x11, 0x1F0(x0)
8 lhu x12, 0x100(x0)
9 lhu x13, 0x1F0(x0)
10 lb x14, 0x1F0(x0)
11 end:
12 j end

```

*

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

```
ASM task4b.s  X
ASM task4b.s
1 .text
2 .globl main
3 main:
4     #a
5     li x1, 1
6     li x2, 2
7     li x3, 3
8     li x4, 4
9     #b
10    li x5, 5
11    li x6, 6
12    li x7, 7
13    li x8, 8
14    #storing a
15    sw x1, 0x100(x0)
16    sw x2, 0x101(x0)
17    sw x3, 0x102(x0)
18    sw x4, 0x103(x0)
19    #storing b
20    sw x5, 0x200(x0)
21    sw x6, 0x202(x0)
22    sw x7, 0x204(x0)
23    sw x8, 0x206(x0)
24    # calculating c
25    add x11, x1, x5
26    add x12, x2, x6
27    add x13, x3, x7
28    add x14, x4, x8
29    # storing c
30    sw x11, 0x300(x0)
31    sw x12, 0x304(x0)
32    sw x13, 0x308(x0)
33    sw x14, 0x30C(x0)
34 end:
35 | j end
36
37
```

Output

RUN AND DEBUG  Launch with  

▼ VARIABLES

 ▼ PC
 PC = 0x000000060

 > PRIV

 > CSR

 ▼ Integer

 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
 x10 (a0) = 0x00000000
 x11 (a1) = 0x00000006
 x12 (a2) = 0x00000008
 x13 (a3) = 0x0000000A
 x14 (a4) = 0x0000000C
 x15 (a5) = 0x00000000
 x16 (a6) = 0x00000000

Stored in memory:

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

Address: Up Down

Jump to: -- choose --

Display Format: Hex

Bytes per Row: 4

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	07	00	08	00
0x00000200	05	00	06	00
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

Address: Up Down

Jump to: -- choose --

Display Format: Hex

Bytes per Row: 4

Address	+0	+1	+2	+3
0x00000318	00	00	00	00
0x00000314	00	00	00	00
0x00000310	00	00	00	00
0x0000030C	0C	00	00	00
0x00000308	0A	00	00	00
0x00000304	08	00	00	00
0x00000300	06	00	00	00
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: -- choose --

Display Format: Hex

Bytes per Row: 4



Assessment Rubric

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

Name: Ayesha Imran, Adina Naveed	Student ID: ai09884, an09842	section*: T3
----------------------------------	------------------------------	--------------

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.

