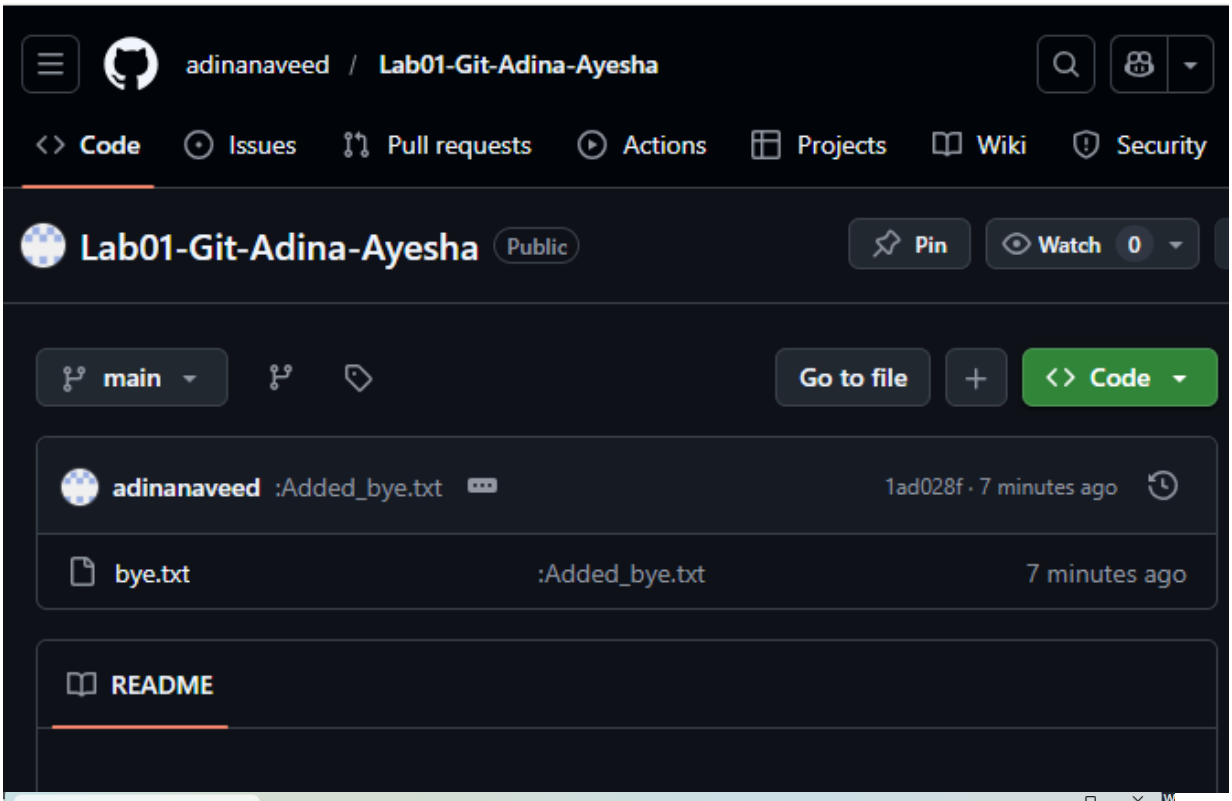
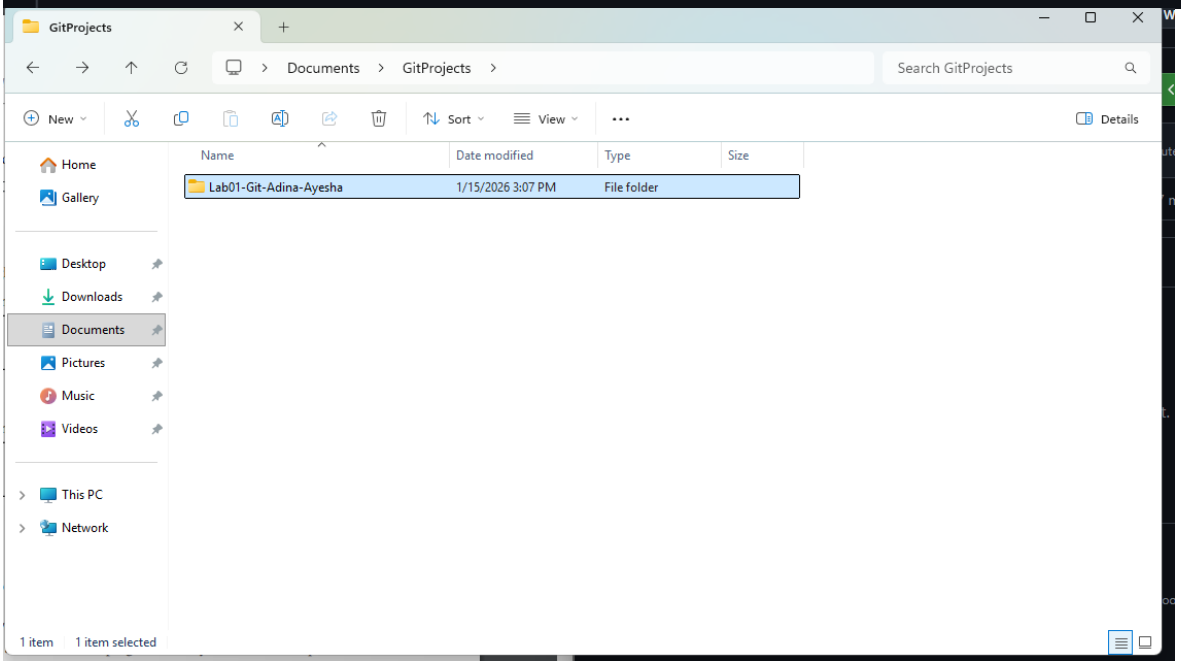
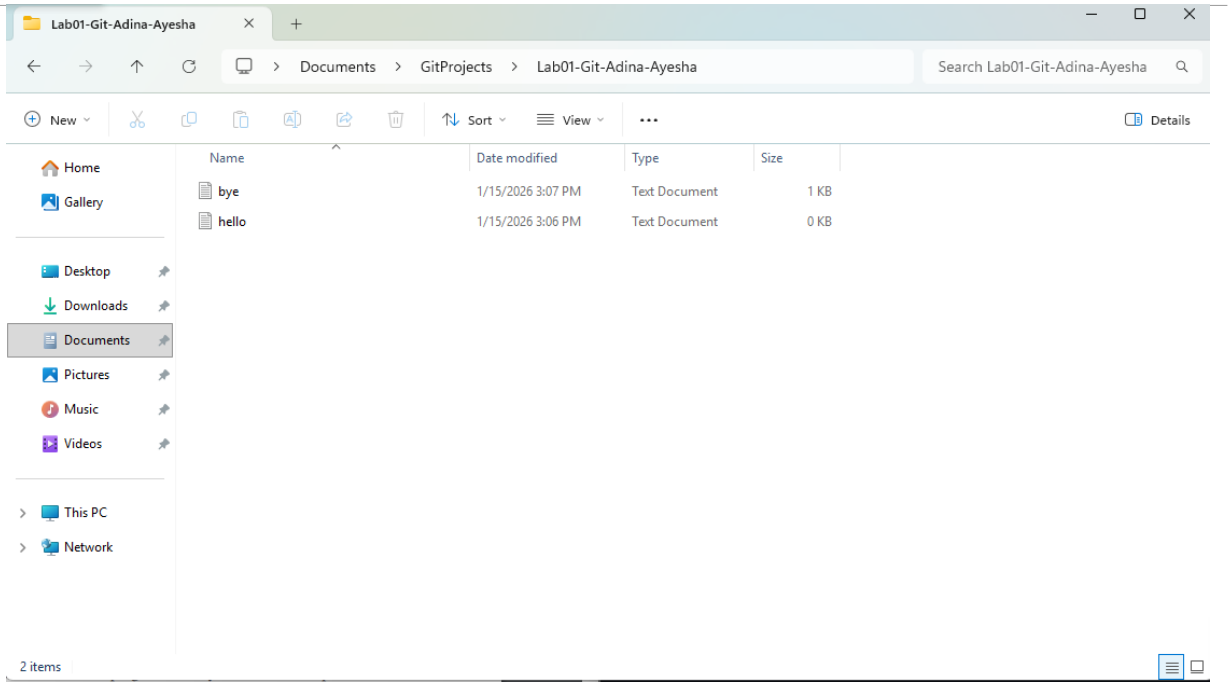
Lab 1: Getting Started with RISC-V (Assembly Language) in VS Code

Task 1:

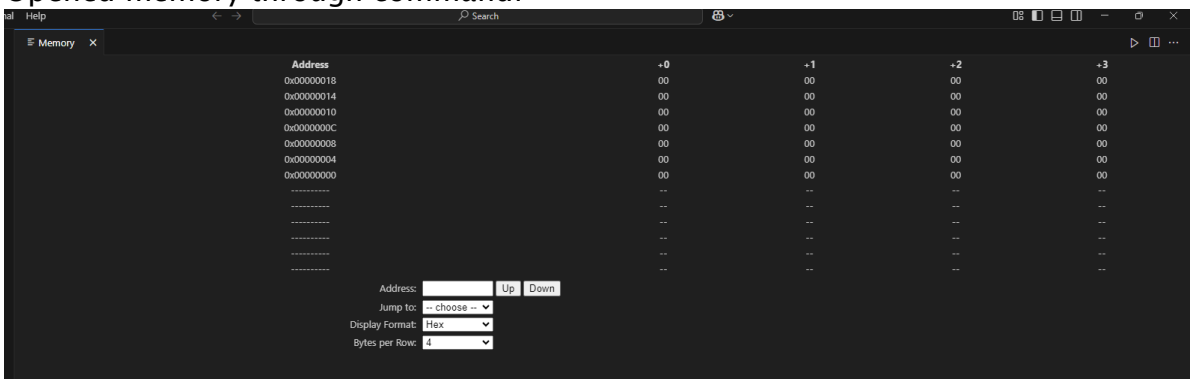





Lab01-Git-Adina-Ayesha
Documents
GitProjects
Lab01-Git-Adina-Ayesha
Search Lab01-Git-Adina-Ayesha
New
Sort
View
Details
Home
Gallery
Desktop
Downloads
Documents
Pictures
Music
Videos
This PC
Network
Name
Date modified
Type
Size
bye
1/15/2026 3:07 PM
Text Document
1 KB
hello
1/15/2026 3:06 PM
Text Document
0 KB
2 items

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

Opened memory through command:



Typed code with register:

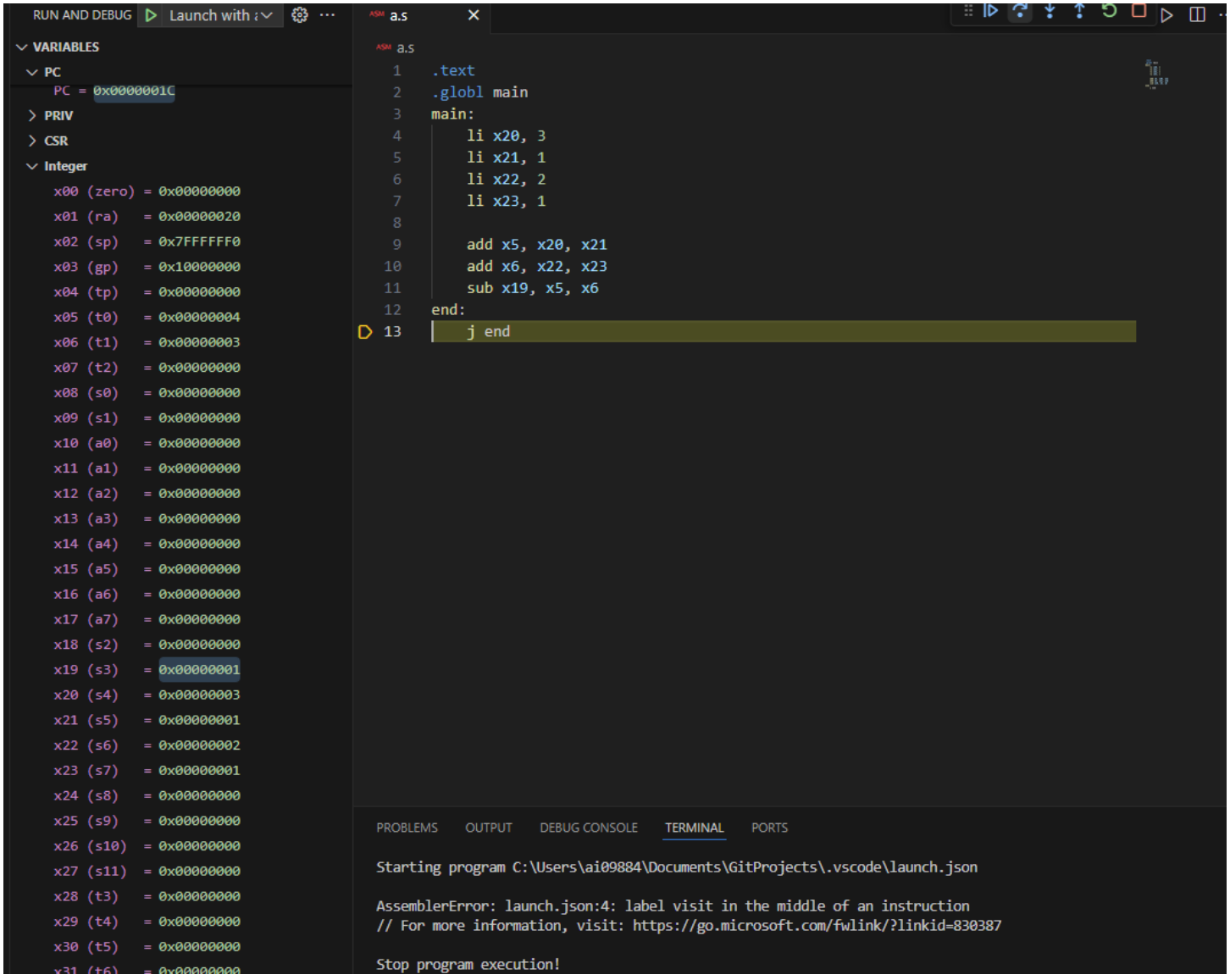


Figure 1.4: RISC-V Memory

Task 3

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

```
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;
```

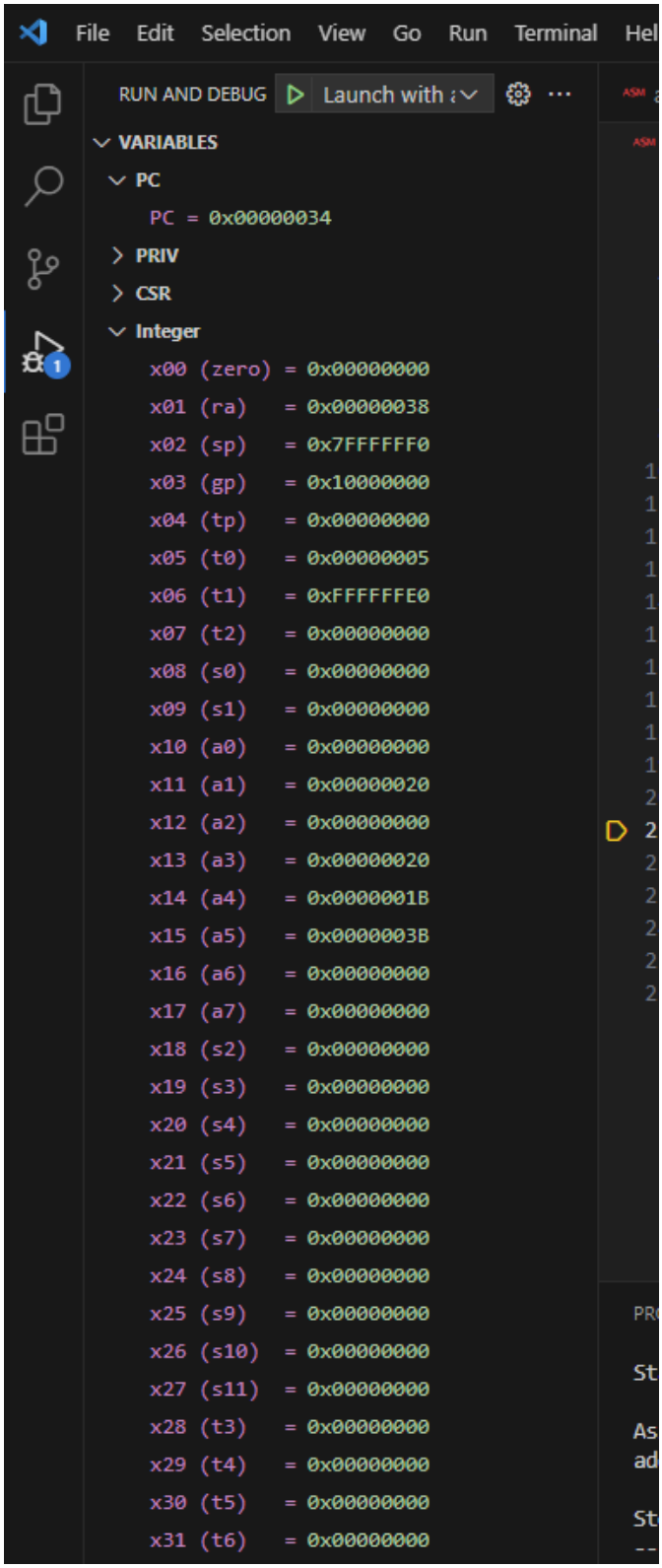
Code

task3.s

```
.text
.globl main
main:
    li x11, 5 #a= 5
    addi x12, x12, 0 #b= 0+0
    li x13, 32
    add x11, x13, x12 #a=b+32

    add x14, x11, x12 # d = a+b
    addi x14, x14, -5 # d = (a+b) - 5

    sub x5, x11, x14 #a-d
    sub x6, x12, x11 #b-a
    add x15, x5, x6 #e = (a-d) + (b-a)
    add x15, x15, x14 # e = ((a-d) + (b-a)) + d

    add x15, x15, x11 # e = a+e
    add x15, x15, x12 # e = a+e+b
    add x15, x15, x14 # e = a+e+b+d
end:
    j end
```

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

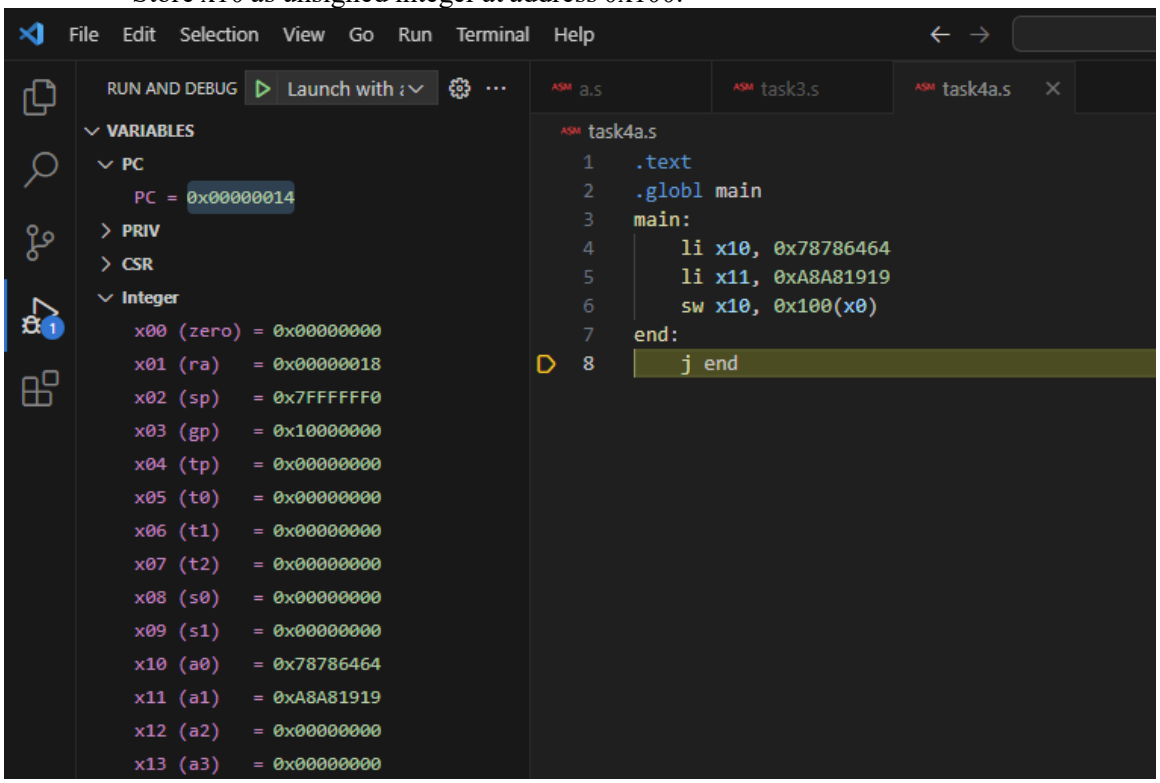
Output


File Edit Selection View Go Run Terminal Hel
RUN AND DEBUG
Launch with
VARIABLES
PC
PC = 0x00000034
PRIV
CSR
Integer
x00 (zero) = 0x00000000
x01 (ra) = 0x00000038
x02 (sp) = 0x7FFFFFF0
x03 (gp) = 0x10000000
x04 (tp) = 0x00000000
x05 (t0) = 0x00000005
x06 (t1) = 0xFFFFFFE0
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

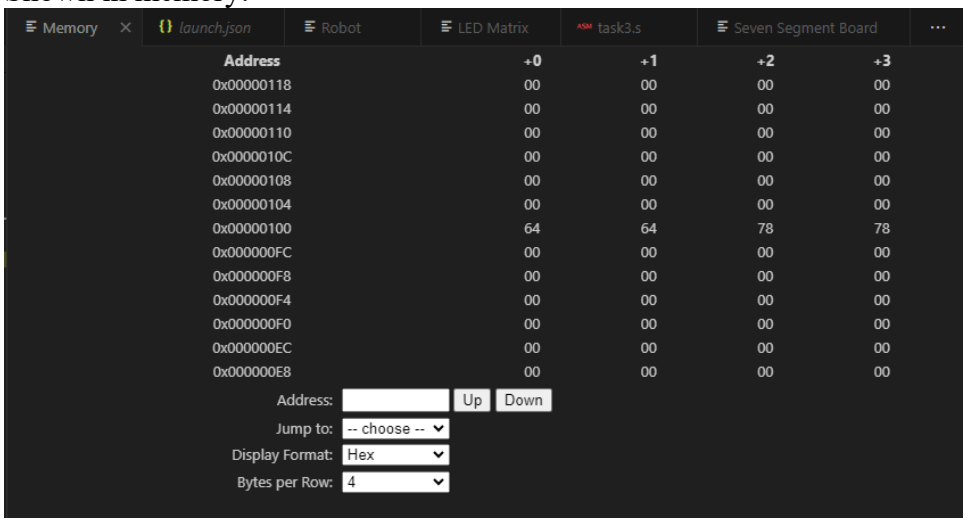
Task 4a

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:

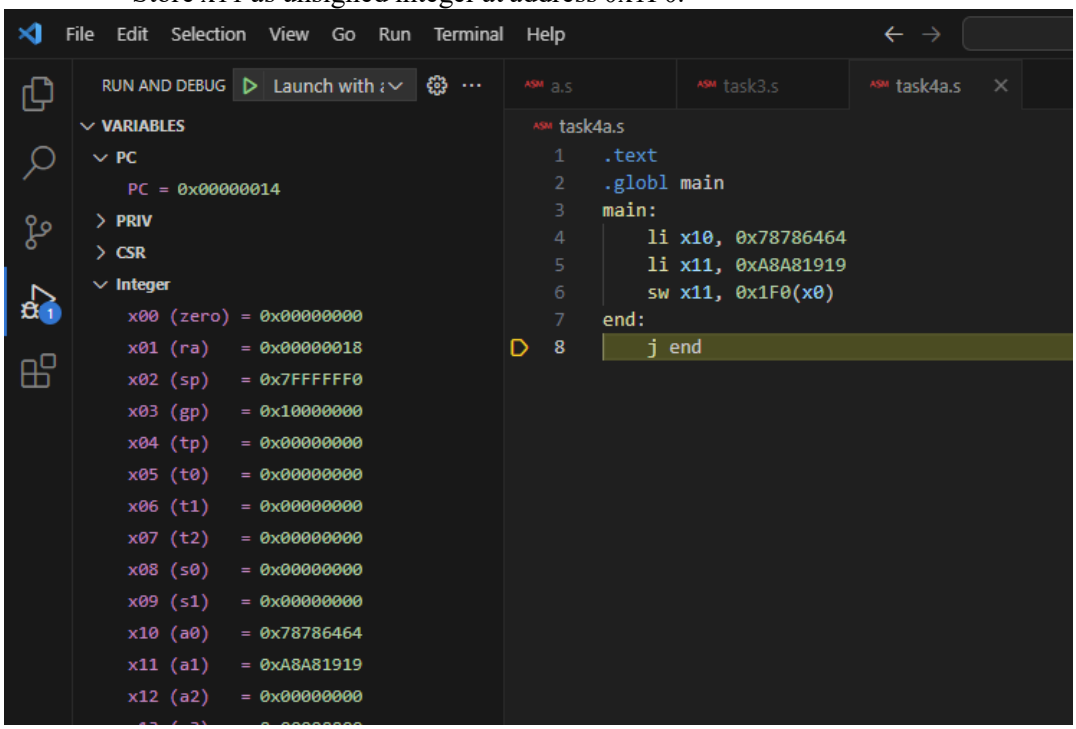
Store x10 as unsigned integer at address 0x100.



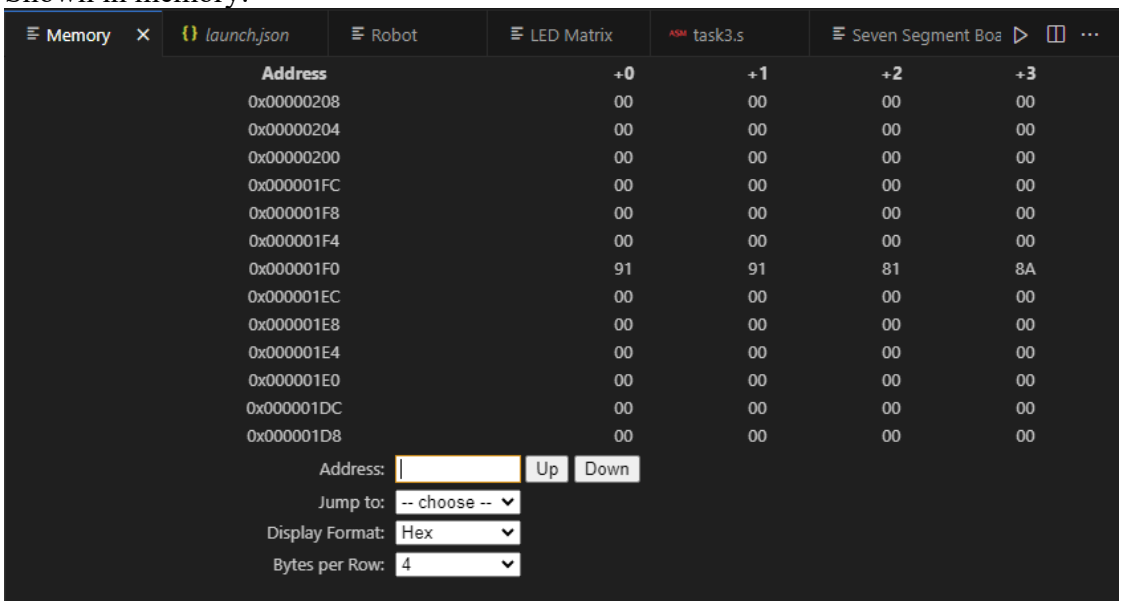
Shown in memory:



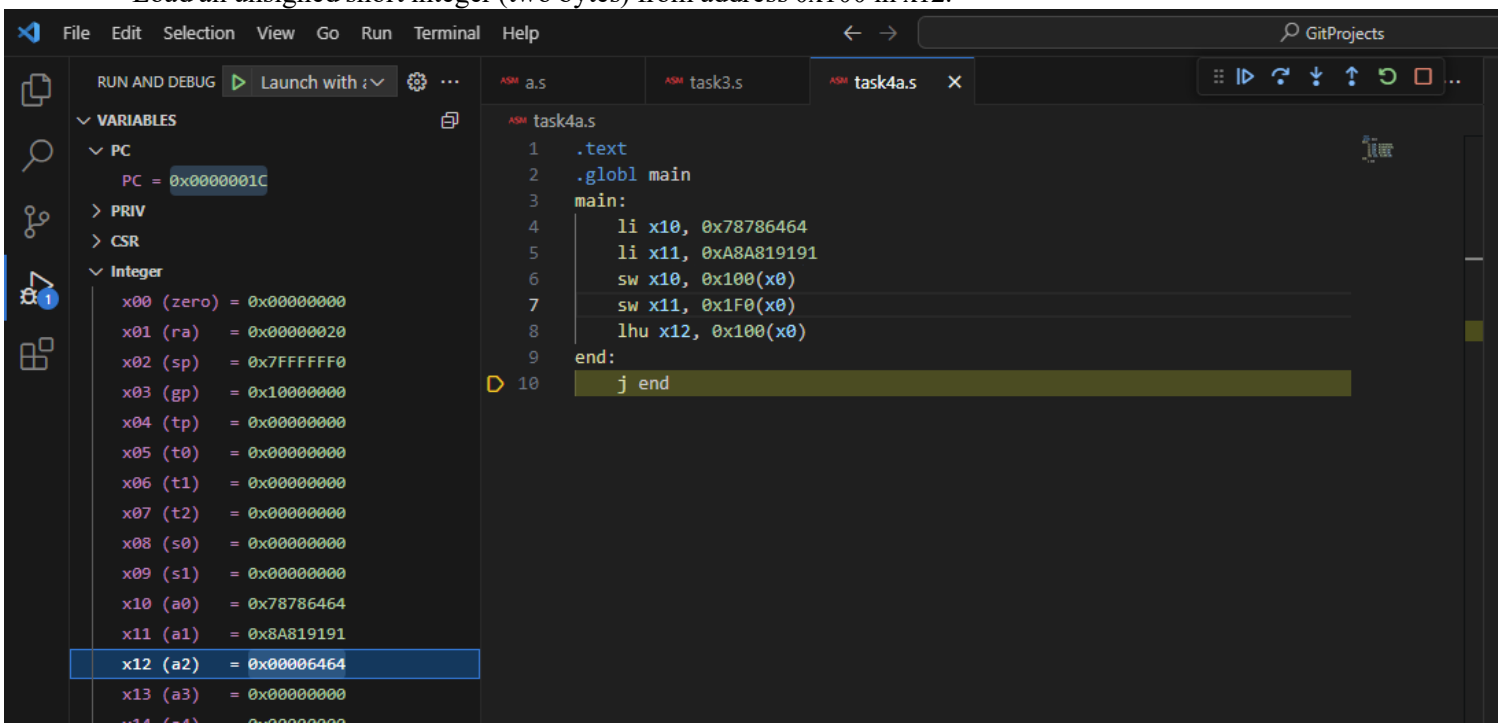
Store x11 as unsigned integer at address 0x1F0.



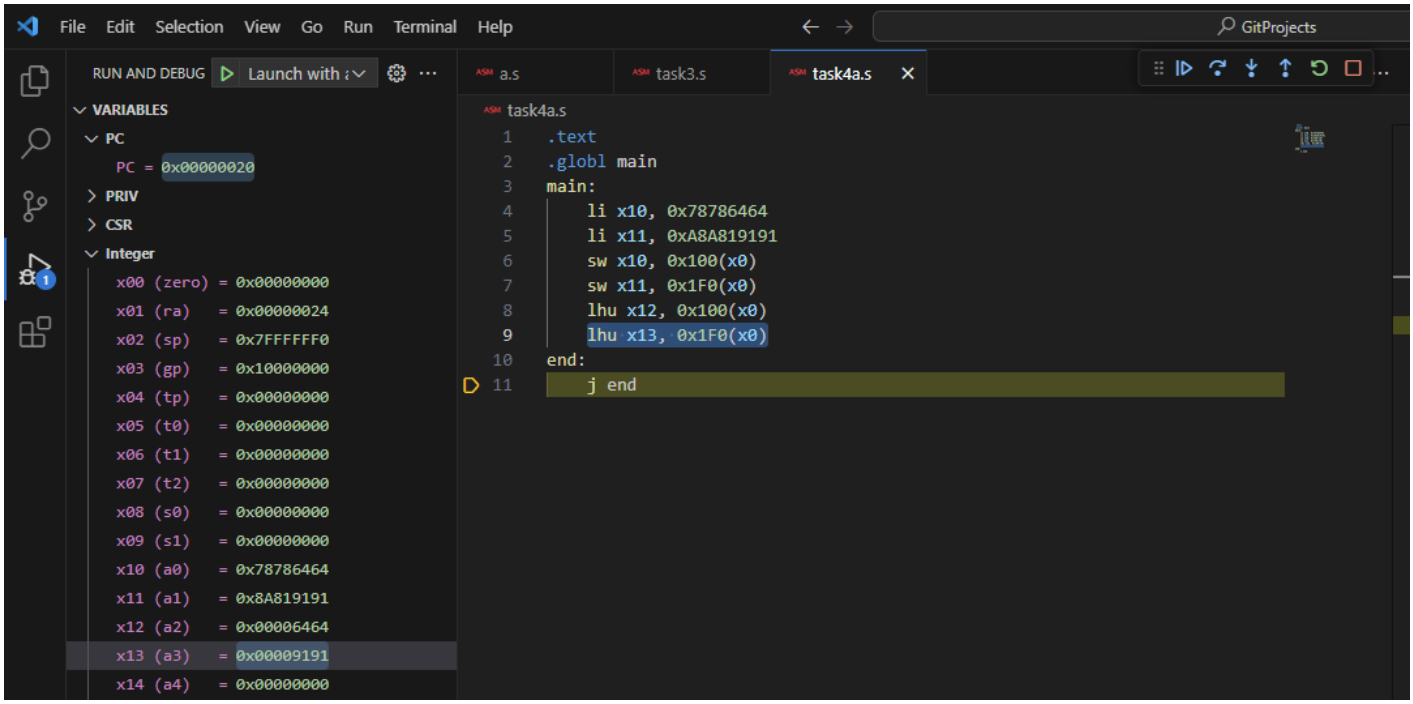
Shown in memory:



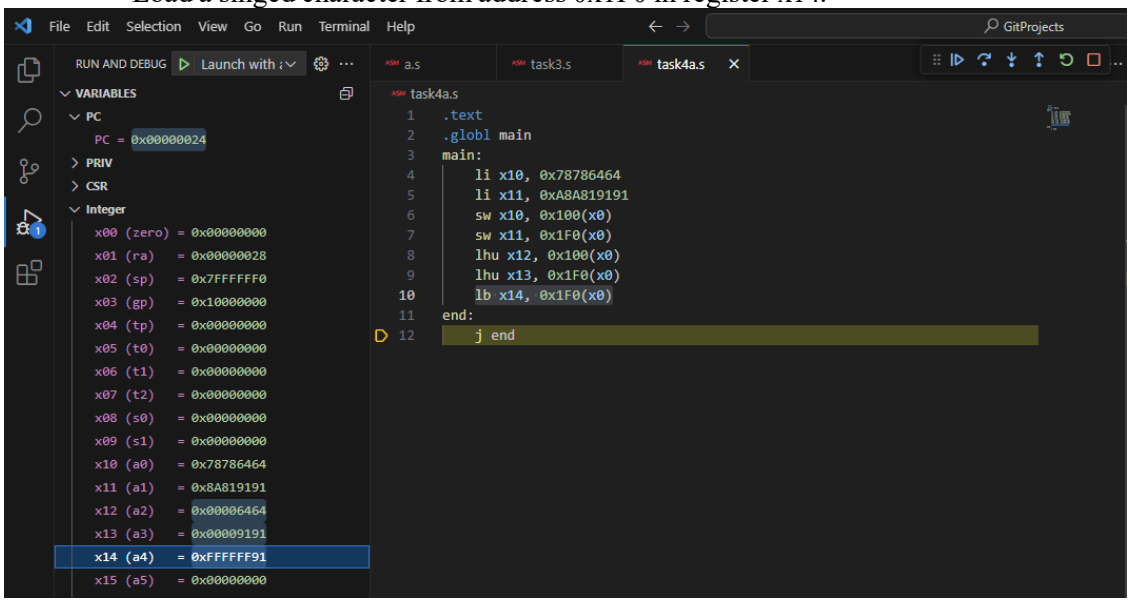
Load an unsigned short integer (two bytes) from address 0x100 in x12.



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



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



*

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 ] ;
```

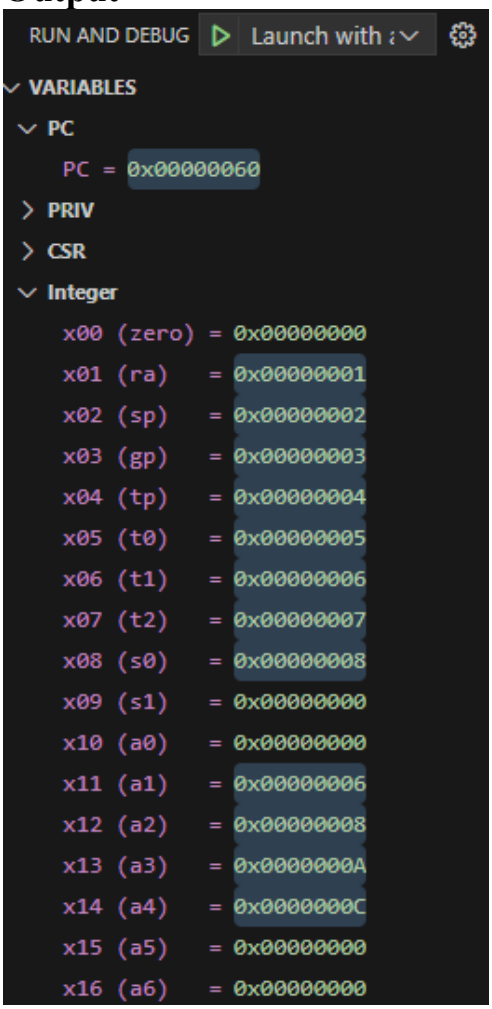
Write equivalent RISC-V code for the piece of code given. You have not studied loops yet, but the above code is manageable without loop instructions. Also assume that A is a character array, B is a short array, and C is an unsigned integer array.

Code

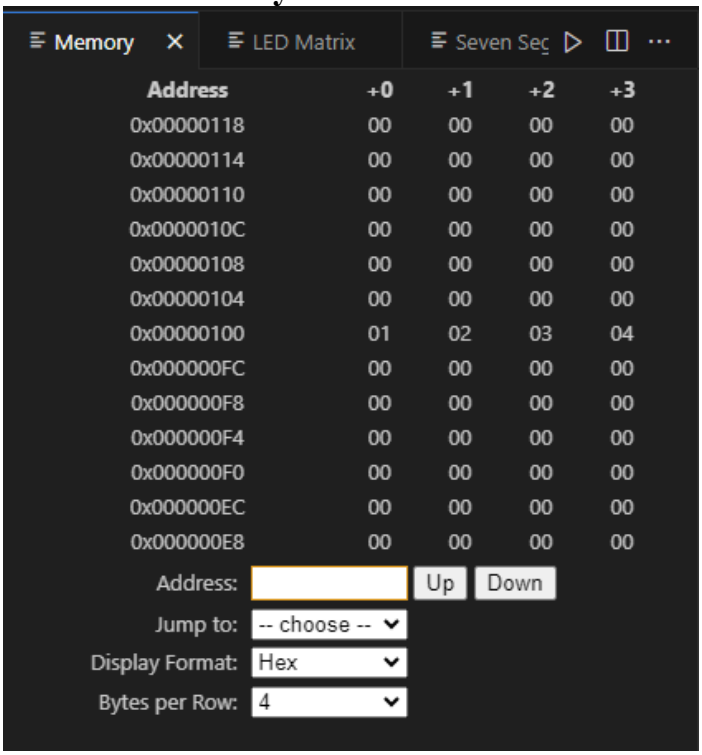
task4b.s

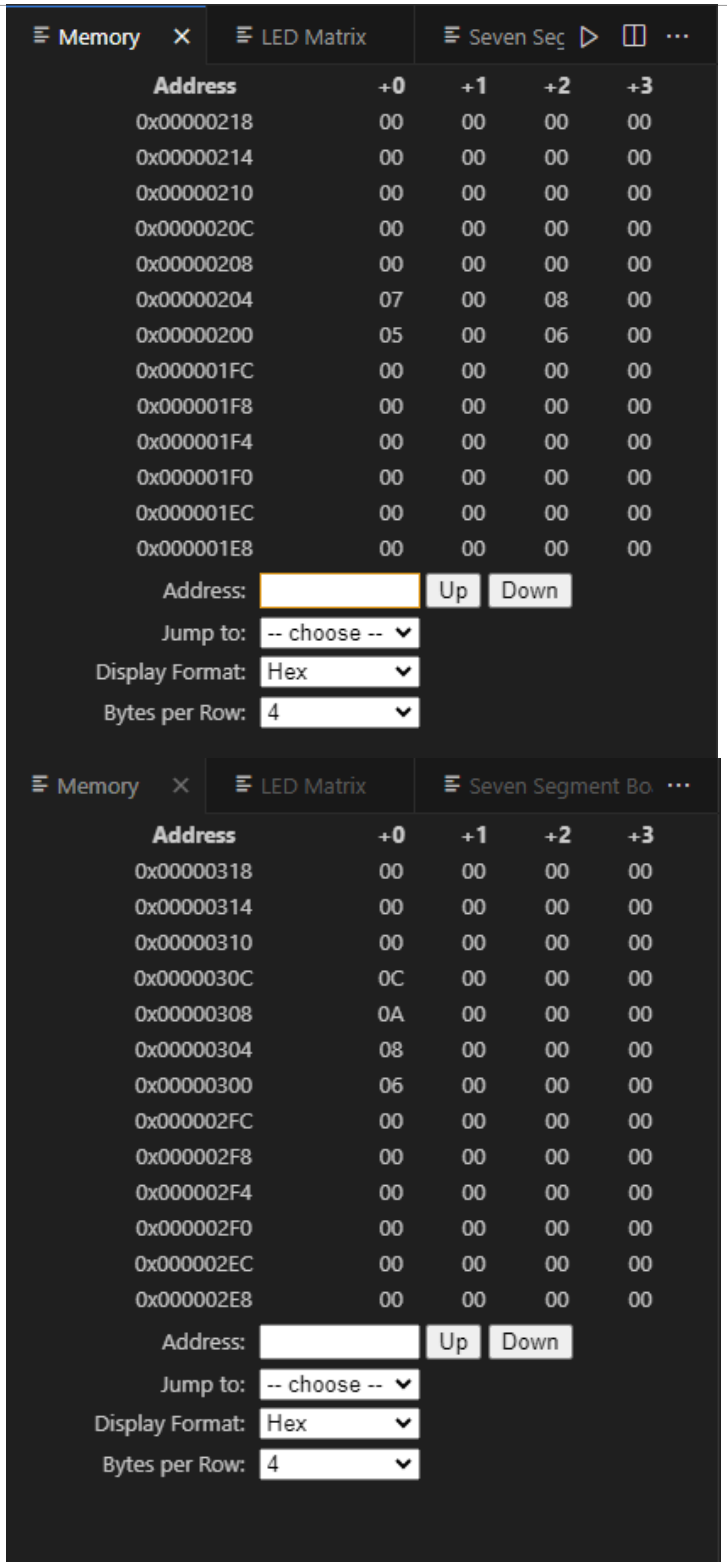
```
.text
.globl main
main:
    #a
    li x1, 1
    li x2, 2
    li x3, 3
    li x4, 4
    #b
    li x5, 5
    li x6, 6
    li x7, 7
    li x8, 8
    #storing a
    sw x1, 0x100(x0)
    sw x2, 0x101(x0)
    sw x3, 0x102(x0)
    sw x4, 0x103(x0)
    #storing b
    sw x5, 0x200(x0)
    sw x6, 0x202(x0)
    sw x7, 0x204(x0)
    sw x8, 0x206(x0)
    # calculating c
    add x11, x1, x5
    add x12, x2, x6
    add x13, x3, x7
    add x14, x4, x8
    # storing c
    sw x11, 0x300(x0)
    sw x12, 0x304(x0)
    sw x13, 0x308(x0)
    sw x14, 0x30C(x0)
end:
    j end
```

Output



Stored in memory:




Memory
LED Matrix
Seven Seg
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
Memory
LED Matrix
Seven Segment Bo
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.