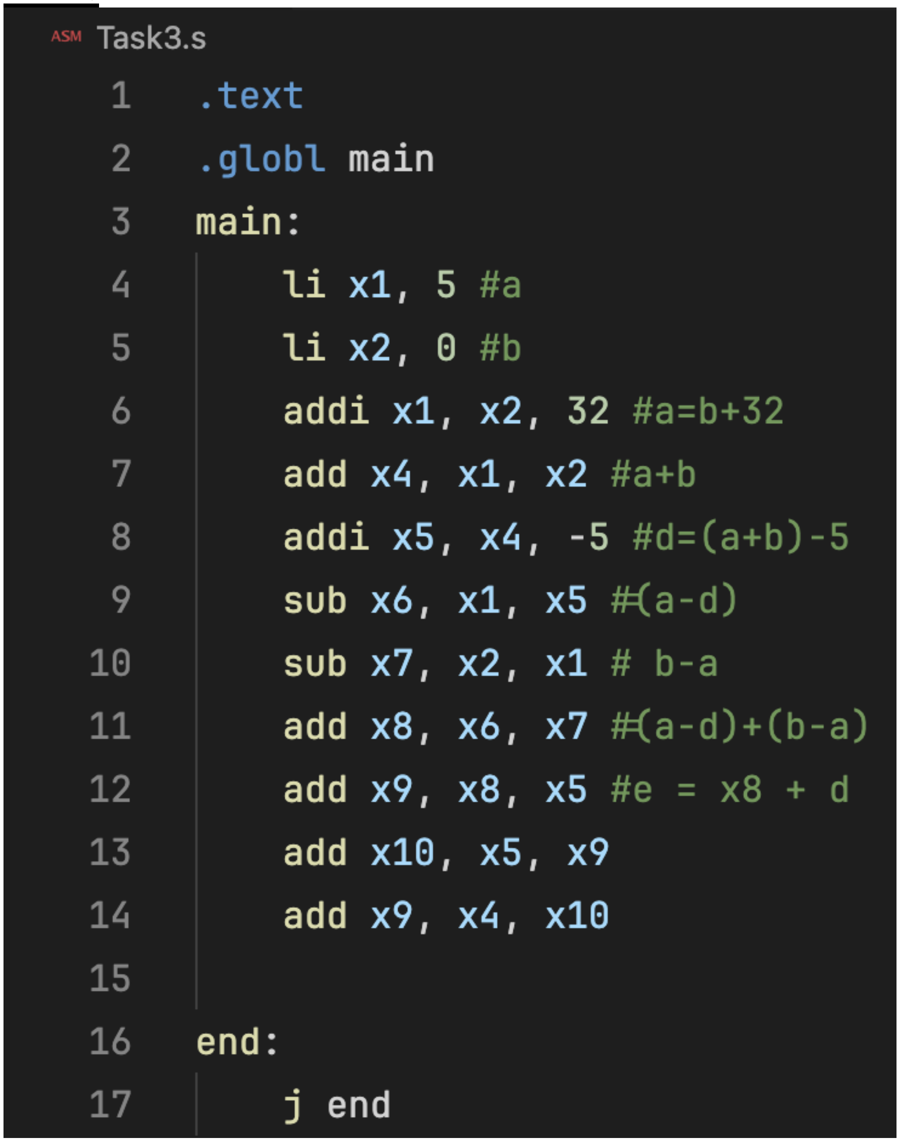
Lab 1 - Getting Started with RISC-V (Assembly Language) in VS Code

Name: Shaheer Qureshi, Areeba Izhar	Student ID: sq09647, ai09625

Task 3:

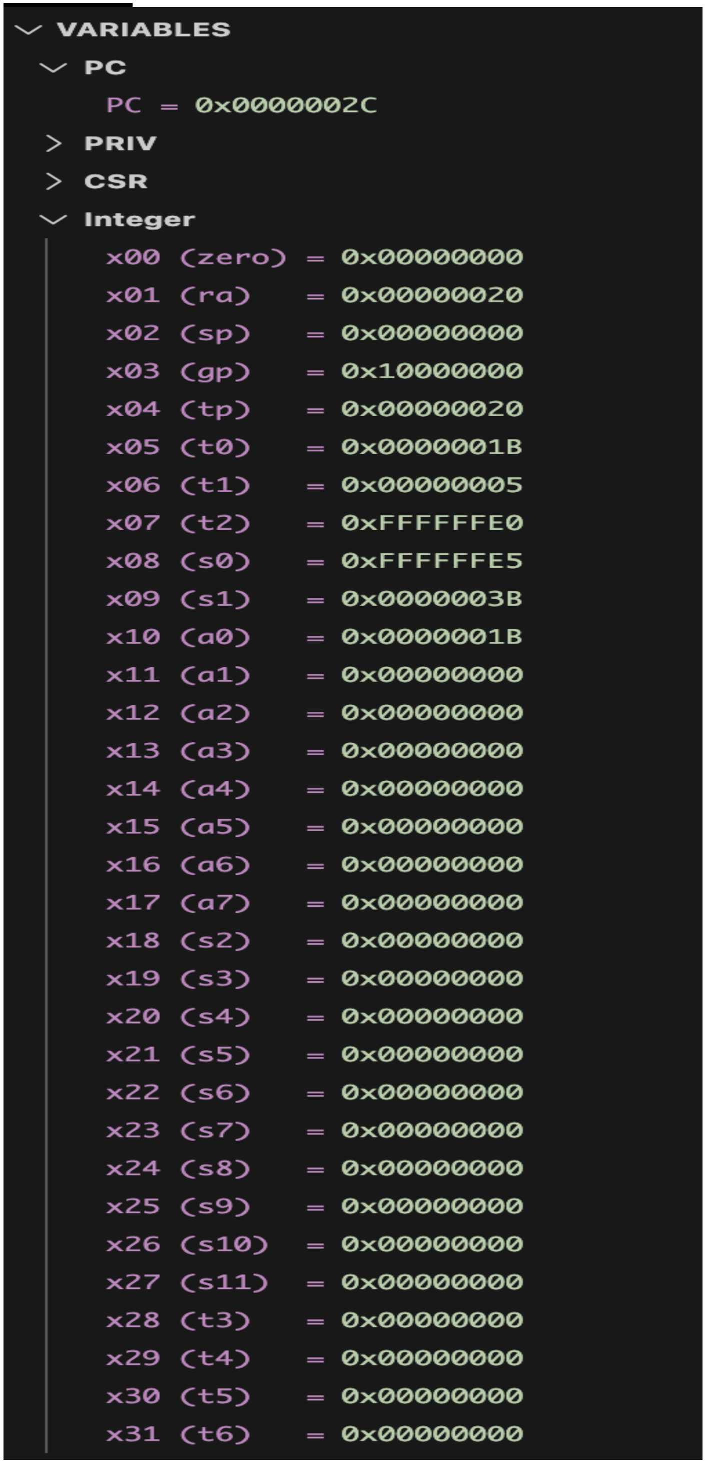
Code:



Task3.s

```
.text
.globl main
main:
    li x1, 5 #a
    li x2, 0 #b
    addi x1, x2, 32 #a=b+32
    add x4, x1, x2 #a+b
    addi x5, x4, -5 #d=(a+b)-5
    sub x6, x1, x5 #(a-d)
    sub x7, x2, x1 # b-a
    add x8, x6, x7 #(a-d)+(b-a)
    add x9, x8, x5 #e = x8 + d
    add x10, x5, x9
    add x9, x4, x10

end:
    j end
```

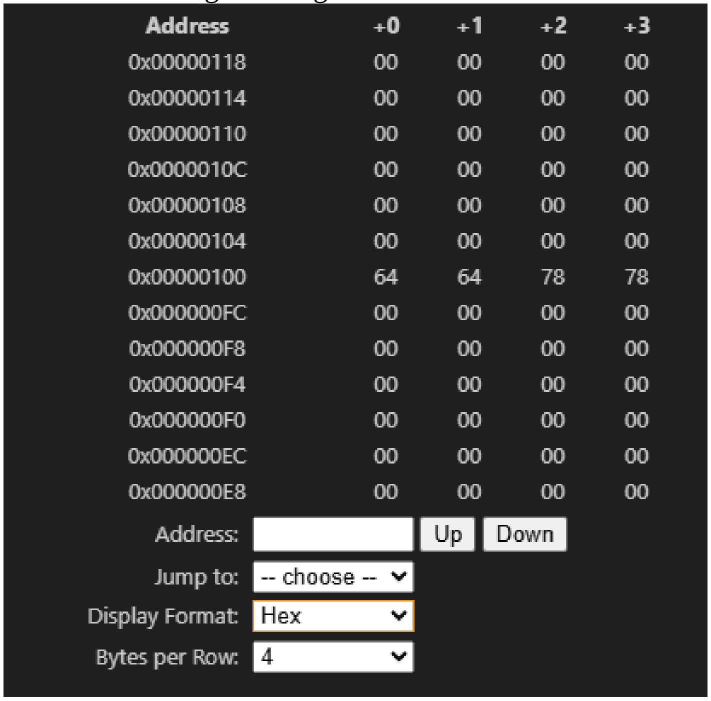
Results:



```
﹀ VARIABLES
  ﹀ PC
      PC = 0x0000002C
  > PRIV
  > CSR
  ﹀ Integer
      x00 (zero) = 0x00000000
      x01 (ra)   = 0x00000020
      x02 (sp)   = 0x00000000
      x03 (gp)   = 0x10000000
      x04 (tp)   = 0x00000020
      x05 (t0)   = 0x0000001B
      x06 (t1)   = 0x00000005
      x07 (t2)   = 0xFFFFFFE0
      x08 (s0)   = 0xFFFFFFE5
      x09 (s1)   = 0x0000003B
      x10 (a0)   = 0x0000001B
      x11 (a1)   = 0x00000000
      x12 (a2)   = 0x00000000
      x13 (a3)   = 0x00000000
      x14 (a4)   = 0x00000000
      x15 (a5)   = 0x00000000
      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:

1. Store x10 as unsigned integer at address 0x100.



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	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

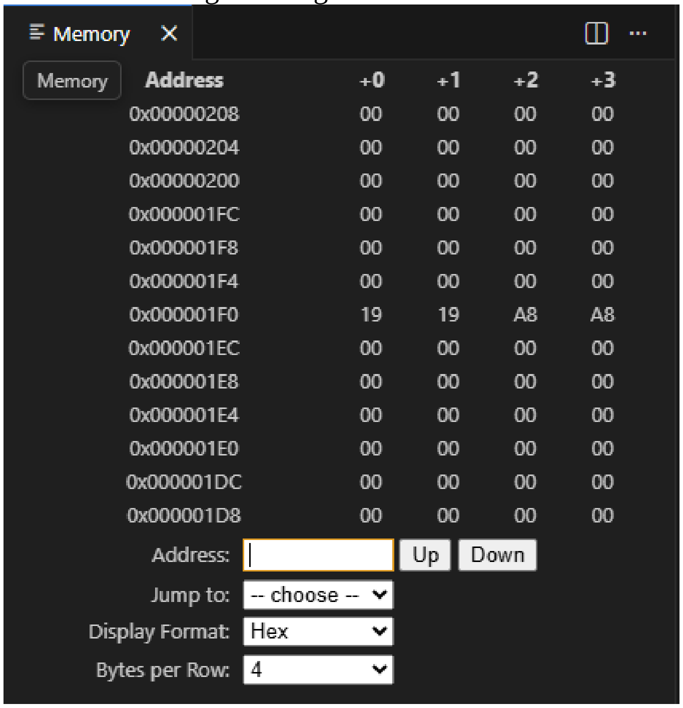
Address:

Jump to: -- choose --

Display Format: Hex

Bytes per Row: 4

2. Store x11 as unsigned integer at address 0x1F0.



Address	+0	+1	+2	+3
0x00000208	00	00	00	00
0x00000204	00	00	00	00
0x00000200	00	00	00	00
0x000001FC	00	00	00	00
0x000001F8	00	00	00	00
0x000001F4	00	00	00	00
0x000001F0	19	19	A8	A8
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

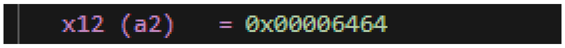
Address:

Jump to: -- choose --

Display Format: Hex

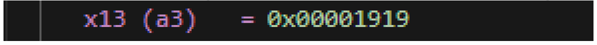
Bytes per Row: 4

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



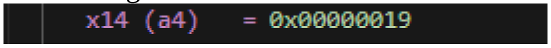
```
x12 (a2)    = 0x00006464
```

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

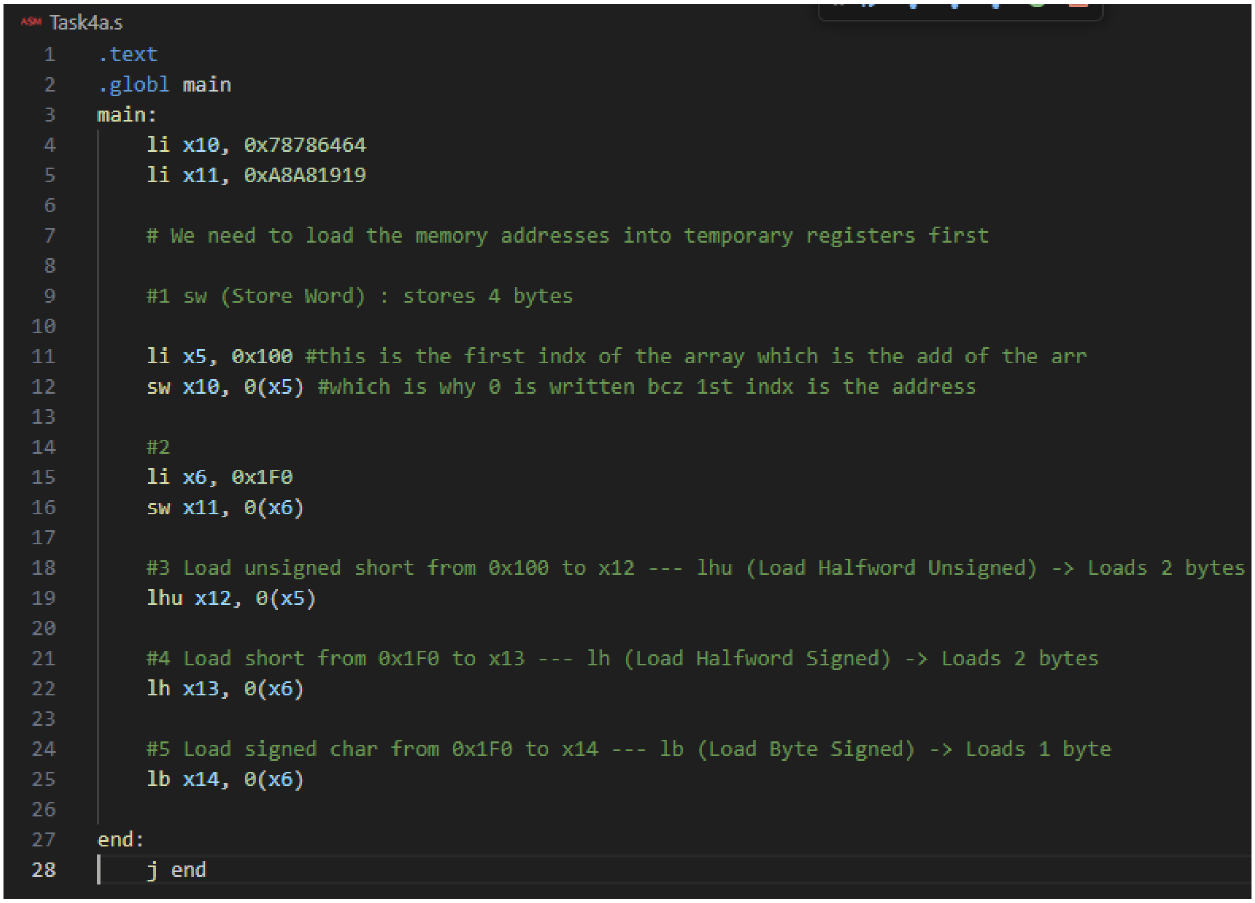


```
x13 (a3)    = 0x00001919
```

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



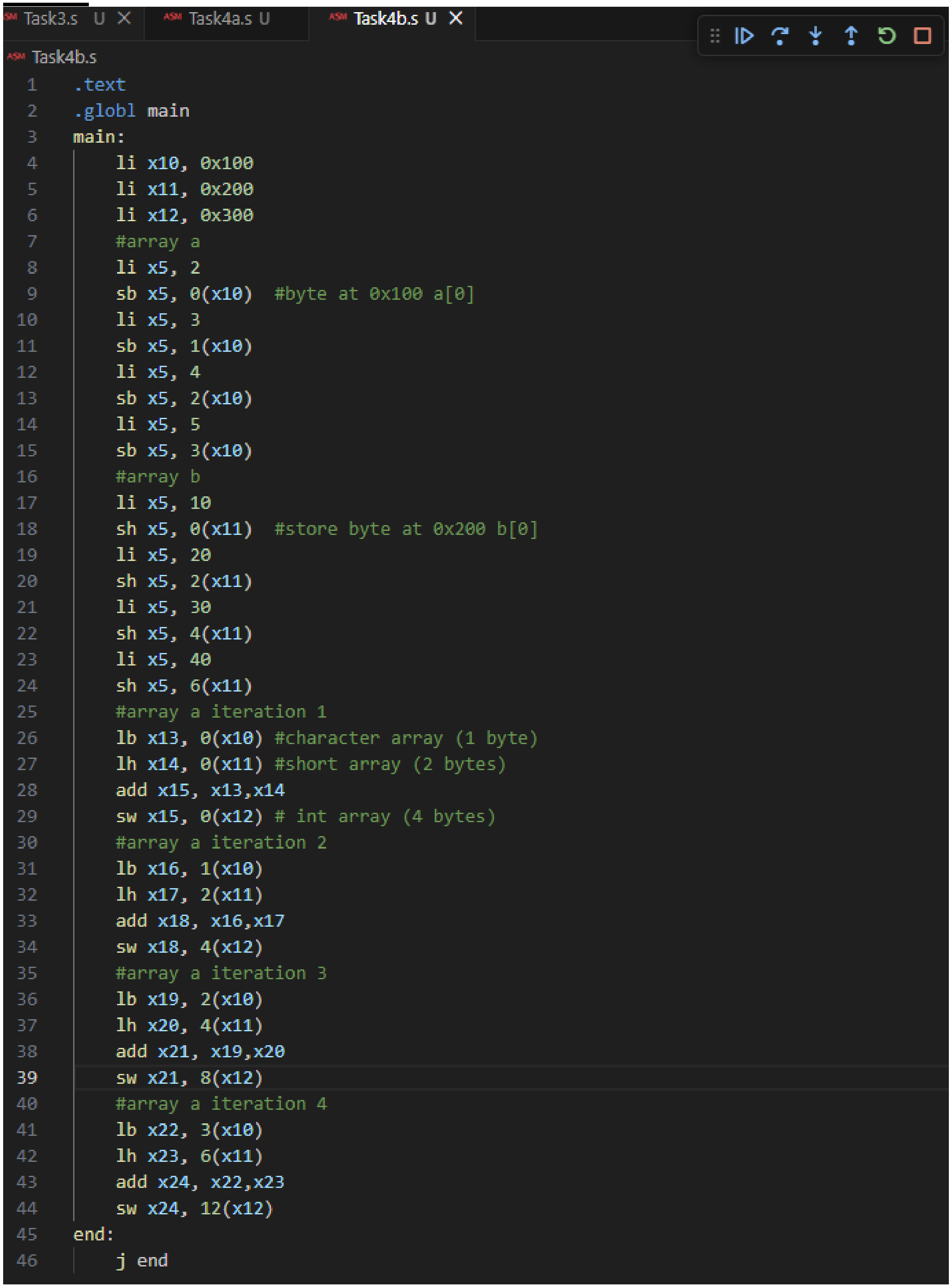
```
x14 (a4)    = 0x00000019
```



```
.text
.globl main
main:
    li x10, 0x78786464
    li x11, 0xA8A81919

    # We need to load the memory addresses into temporary registers first

    #1 sw (Store Word) : stores 4 bytes

    li x5, 0x100 #this is the first indx of the array which is the add of the arr
    sw x10, 0(x5) #which is why 0 is written bcz 1st indx is the address

    #2
    li x6, 0x1F0
    sw x11, 0(x6)

    #3 Load unsigned short from 0x100 to x12 --- lhu (Load Halfword Unsigned) -> Loads 2 bytes
    lhu x12, 0(x5)

    #4 Load short from 0x1F0 to x13 --- lh (Load Halfword Signed) -> Loads 2 bytes
    lh x13, 0(x6)

    #5 Load signed char from 0x1F0 to x14 --- lb (Load Byte Signed) -> Loads 1 byte
    lb x14, 0(x6)

end:
    j end
```

Task 4b:

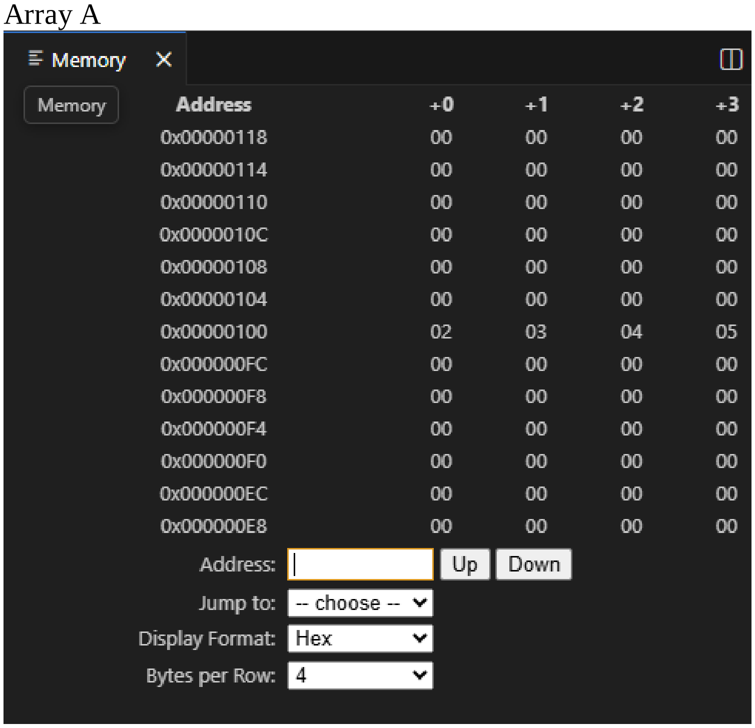
Code:



```
.text
.globl main
main:
    li x10, 0x100
    li x11, 0x200
    li x12, 0x300
    #array a
    li x5, 2
    sb x5, 0(x10)  #byte at 0x100 a[0]
    li x5, 3
    sb x5, 1(x10)
    li x5, 4
    sb x5, 2(x10)
    li x5, 5
    sb x5, 3(x10)
    #array b
    li x5, 10
    sh x5, 0(x11)  #store byte at 0x200 b[0]
    li x5, 20
    sh x5, 2(x11)
    li x5, 30
    sh x5, 4(x11)
    li x5, 40
    sh x5, 6(x11)
    #array a iteration 1
    lb x13, 0(x10) #character array (1 byte)
    lh x14, 0(x11) #short array (2 bytes)
    add x15, x13,x14
    sw x15, 0(x12) # int array (4 bytes)
    #array a iteration 2
    lb x16, 1(x10)
    lh x17, 2(x11)
    add x18, x16,x17
    sw x18, 4(x12)
    #array a iteration 3
    lb x19, 2(x10)
    lh x20, 4(x11)
    add x21, x19,x20
    sw x21, 8(x12)
    #array a iteration 4
    lb x22, 3(x10)
    lh x23, 6(x11)
    add x24, x22,x23
    sw x24, 12(x12)
end:
    j end
```

Results:

Array A



```
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   02  03  04  05
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
```

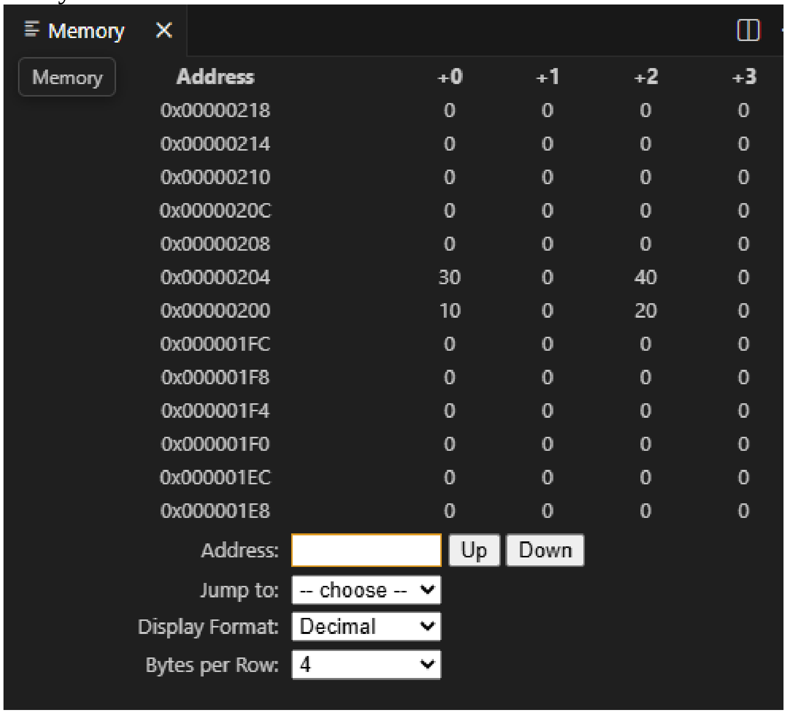
Address:

Jump to: -- choose --

Display Format: Hex

Bytes per Row: 4

Array B



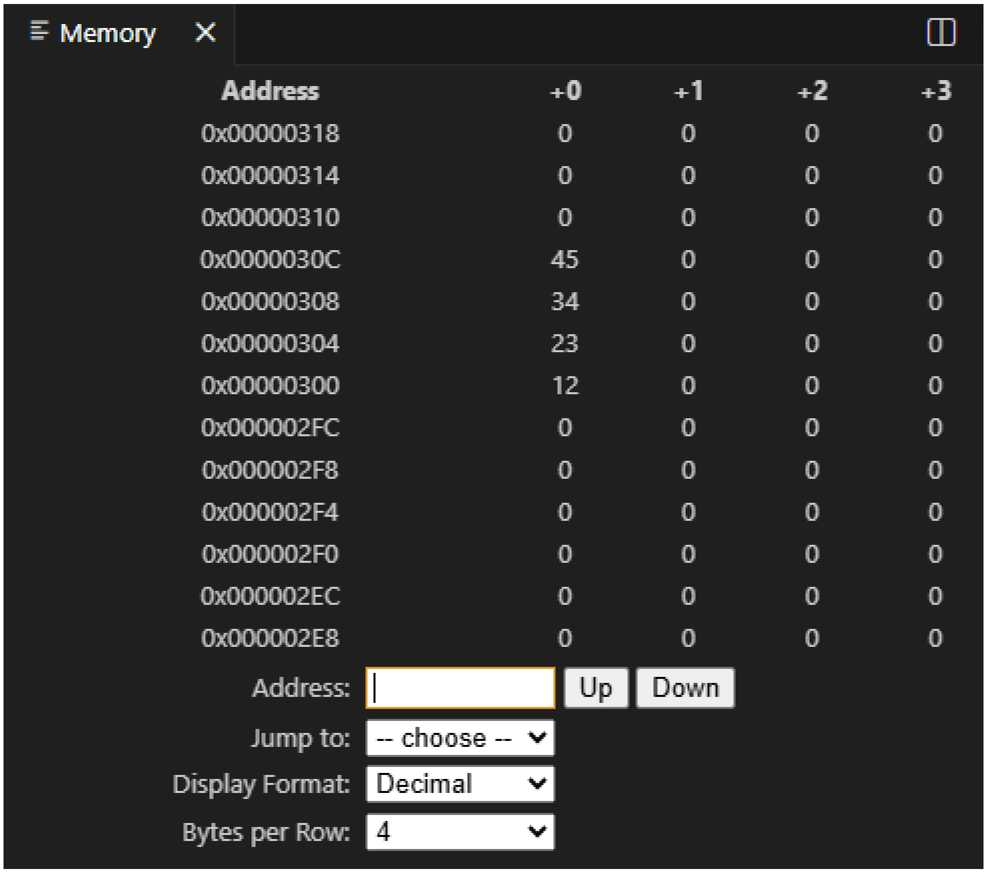
```
Address      +0  +1  +2  +3
0x00000218   0   0   0   0
0x00000214   0   0   0   0
0x00000210   0   0   0   0
0x0000020C   0   0   0   0
0x00000208   0   0   0   0
0x00000204   30  0   40  0
0x00000200   10  0   20  0
0x000001FC   0   0   0   0
0x000001F8   0   0   0   0
0x000001F4   0   0   0   0
0x000001F0   0   0   0   0
0x000001EC   0   0   0   0
0x000001E8   0   0   0   0
```

Address:

Jump to: -- choose --

Display Format: Decimal

Bytes per Row: 4



Address	+0	+1	+2	+3
0x00000318	0	0	0	0
0x00000314	0	0	0	0
0x00000310	0	0	0	0
0x0000030C	45	0	0	0
0x00000308	34	0	0	0
0x00000304	23	0	0	0
0x00000300	12	0	0	0
0x000002FC	0	0	0	0
0x000002F8	0	0	0	0
0x000002F4	0	0	0	0
0x000002F0	0	0	0	0
0x000002EC	0	0	0	0
0x000002E8	0	0	0	0

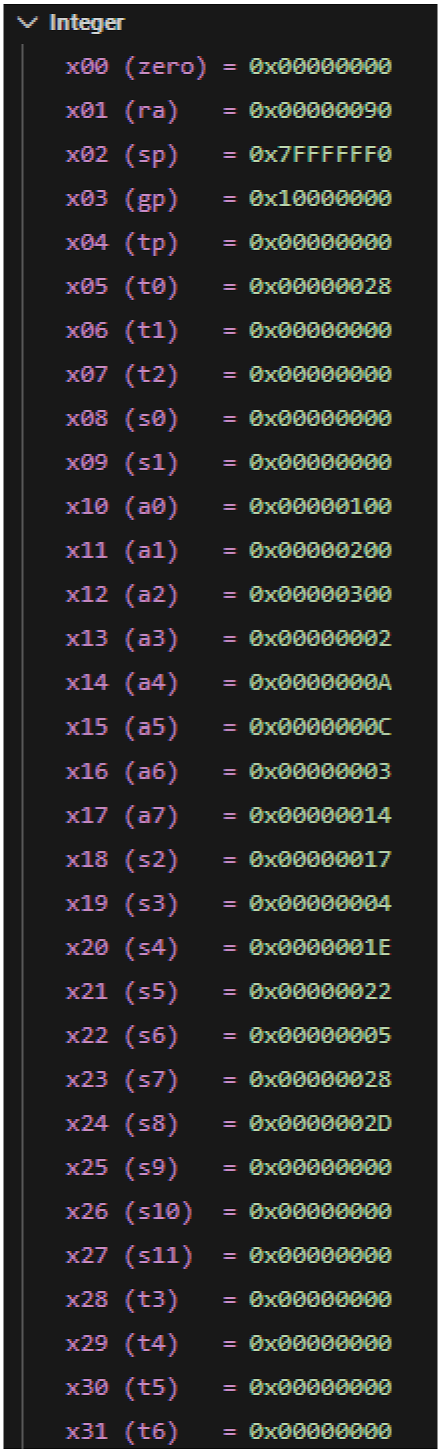
Address:

Jump to: -- choose --

Display Format: Decimal

Bytes per Row: 4

Array C



```
Integer
  x00 (zero) = 0x00000000
  x01 (ra)   = 0x00000090
  x02 (sp)   = 0x7FFFFFF0
  x03 (gp)   = 0x10000000
  x04 (tp)   = 0x00000000
  x05 (t0)   = 0x00000028
  x06 (t1)   = 0x00000000
  x07 (t2)   = 0x00000000
  x08 (s0)   = 0x00000000
  x09 (s1)   = 0x00000000
  x10 (a0)   = 0x00000100
  x11 (a1)   = 0x00000200
  x12 (a2)   = 0x00000300
  x13 (a3)   = 0x00000002
  x14 (a4)   = 0x0000000A
  x15 (a5)   = 0x0000000C
  x16 (a6)   = 0x00000003
  x17 (a7)   = 0x00000014
  x18 (s2)   = 0x00000017
  x19 (s3)   = 0x00000004
  x20 (s4)   = 0x0000001E
  x21 (s5)   = 0x00000022
  x22 (s6)   = 0x00000005
  x23 (s7)   = 0x00000028
  x24 (s8)   = 0x0000002D
  x25 (s9)   = 0x00000000
  x26 (s10)  = 0x00000000
  x27 (s11)  = 0x00000000
  x28 (t3)   = 0x00000000
  x29 (t4)   = 0x00000000
  x30 (t5)   = 0x00000000
  x31 (t6)   = 0x00000000
```

Assessment Rubric
Lab 1: Getting Started with RISC-V (Assembly Language) n VS Code

Name	Student ID:	Section:

Points Distribution:

	Task No.	**LR 2 (Code)**	**LR 5 (Results)**
In-Lab	**Task 1**	-	/15
	Task 2	-	/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**	