

**Iqra University Islamabad**

**Name:** Aksah Akhalq

**Student ID:** 41593

**Submitted To:** Sir Asad Ullah Khan

**Submission Date:** November 20, 2025

**Department:** Computer Science

**LAB TASK 5**

TITLE Memory Operations Lab (memory\_ops.asm)

; Program Description: This program demonstrates storing,  
incrementing, and adding  
; bytes in memory.

.686

.MODEL FLAT, STDCALL

.STACK

INCLUDE Irvine32.inc

.DATA

; Task 1: Ten bytes initialized to 1

byteArray1 BYTE 10, 20, 30, 40, 50, 60, 70, 80, 90, 0Ah

; Task 2: Sixteen bytes initialized to 0

byteArray2 BYTE 16 DUP(0)

; Task 3: Variable to store sum

sum DWORD 0

.CODE

main PROC

; Task 2: Increment each byte in byteArray2 by 1

mov esi, OFFSET byteArray2

mov ecx, 16

L1:

L2:

inc BYTE PTR [esi]

inc esi

loop L1

; Task 3: Sum ten bytes from byteArray1

mov esi, OFFSET byteArray1

mov ecx, 10

mov eax, 0

movzx ebx, BYTE PTR [esi]

add eax, ebx

inc esi

loop L2

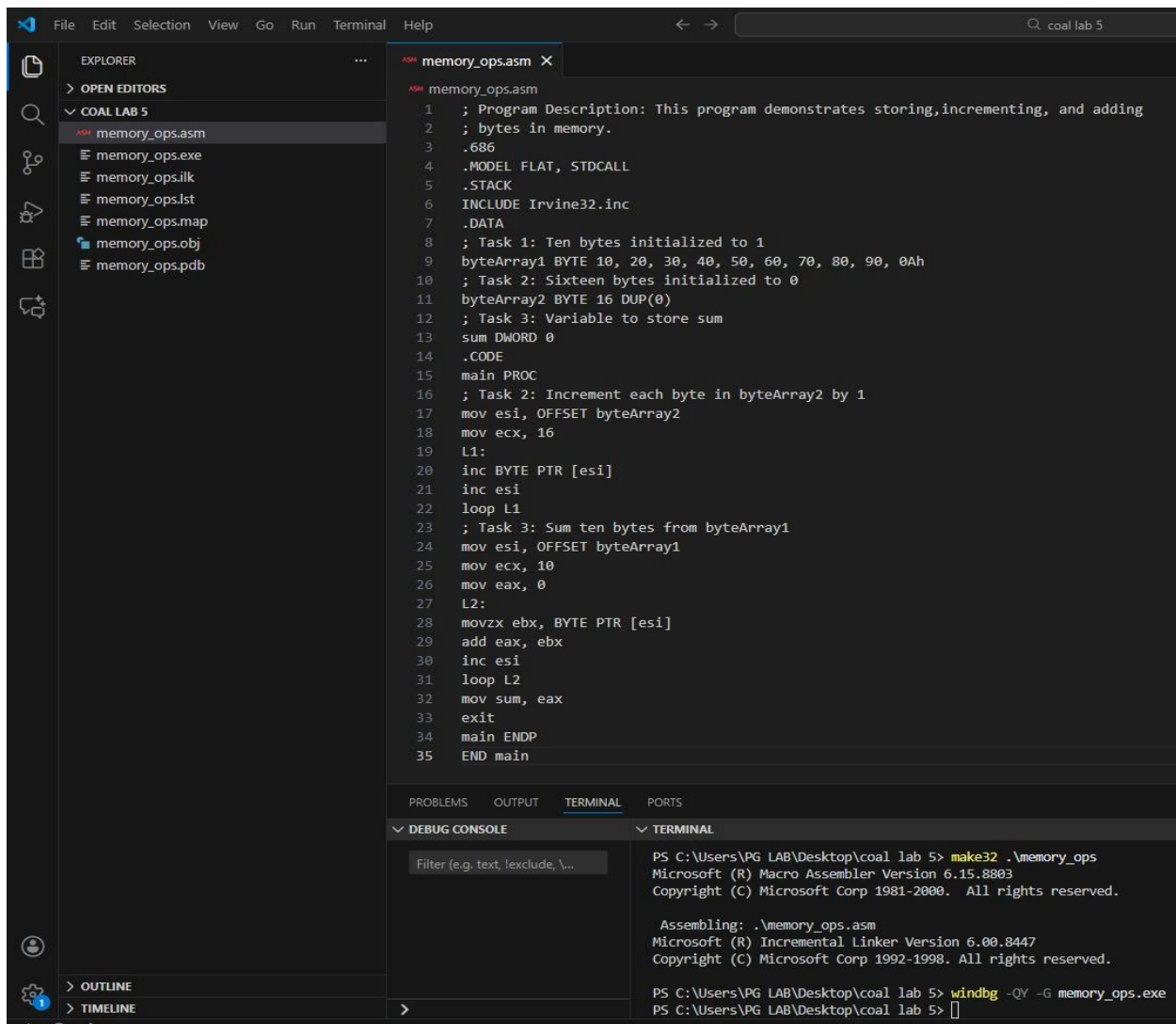
mov sum, eax

exit

main ENDP

END main

## Terminal Output



The screenshot shows the Visual Studio Code interface with the assembly file `memory_ops.asm` open in the editor. The code is as follows:

```
1 ; Program Description: This program demonstrates storing, incrementing, and adding
2 ; bytes in memory.
3 .686
4 .MODEL FLAT, STDCALL
5 .STACK
6 INCLUDE Irvine32.inc
7 .DATA
8 ; Task 1: Ten bytes initialized to 1
9 byteArray1 BYTE 10, 20, 30, 40, 50, 60, 70, 80, 90, 0Ah
10 ; Task 2: Sixteen bytes initialized to 0
11 byteArray2 BYTE 16 DUP(0)
12 ; Task 3: Variable to store sum
13 sum DWORD 0
14 .CODE
15 main PROC
16 ; Task 2: Increment each byte in byteArray2 by 1
17 mov esi, OFFSET byteArray2
18 mov ecx, 16
19 L1:
20 inc BYTE PTR [esi]
21 inc esi
22 loop L1
23 ; Task 3: Sum ten bytes from byteArray1
24 mov esi, OFFSET byteArray1
25 mov ecx, 10
26 mov eax, 0
27 L2:
28 movzx ebx, BYTE PTR [esi]
29 add eax, ebx
30 inc esi
31 loop L2
32 mov sum, eax
33 exit
34 main ENDP
35 END main
```

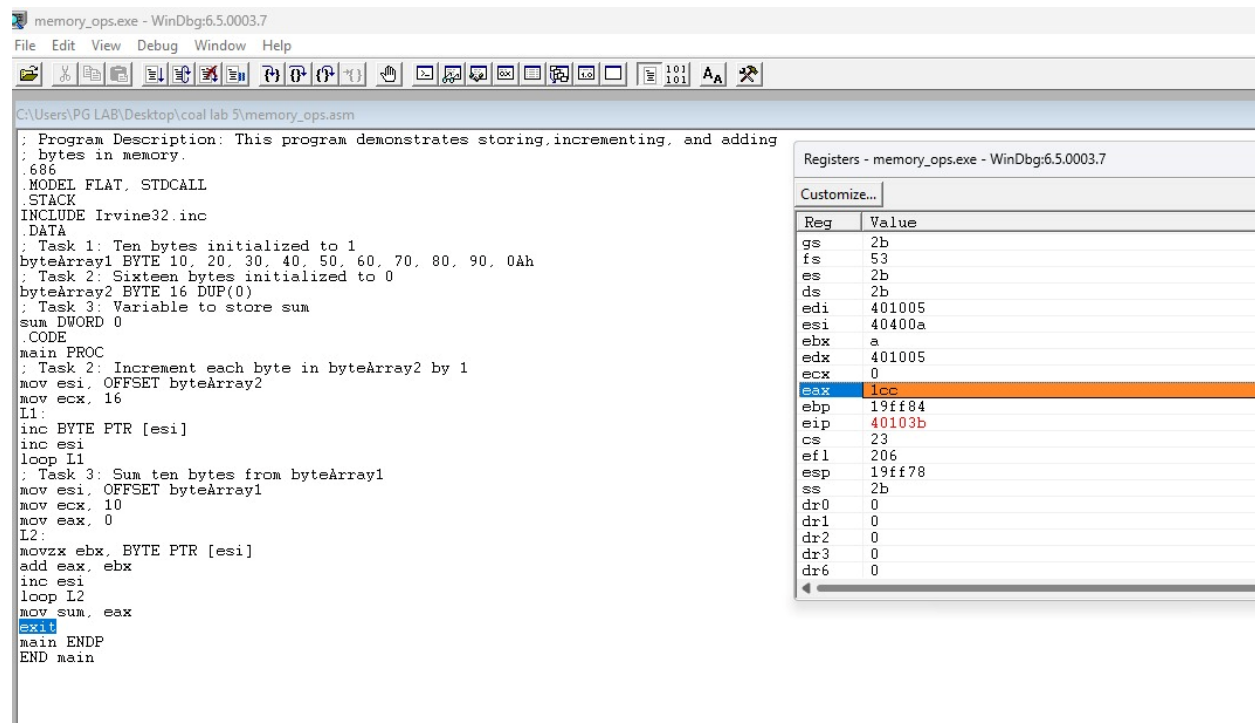
The terminal output shows the following commands and their results:

```
PS C:\Users\PG LAB\Desktop\coal lab 5> make32 .\memory_ops
Microsoft (R) Macro Assembler Version 6.15.8803
Copyright (C) Microsoft Corp 1981-2000. All rights reserved.

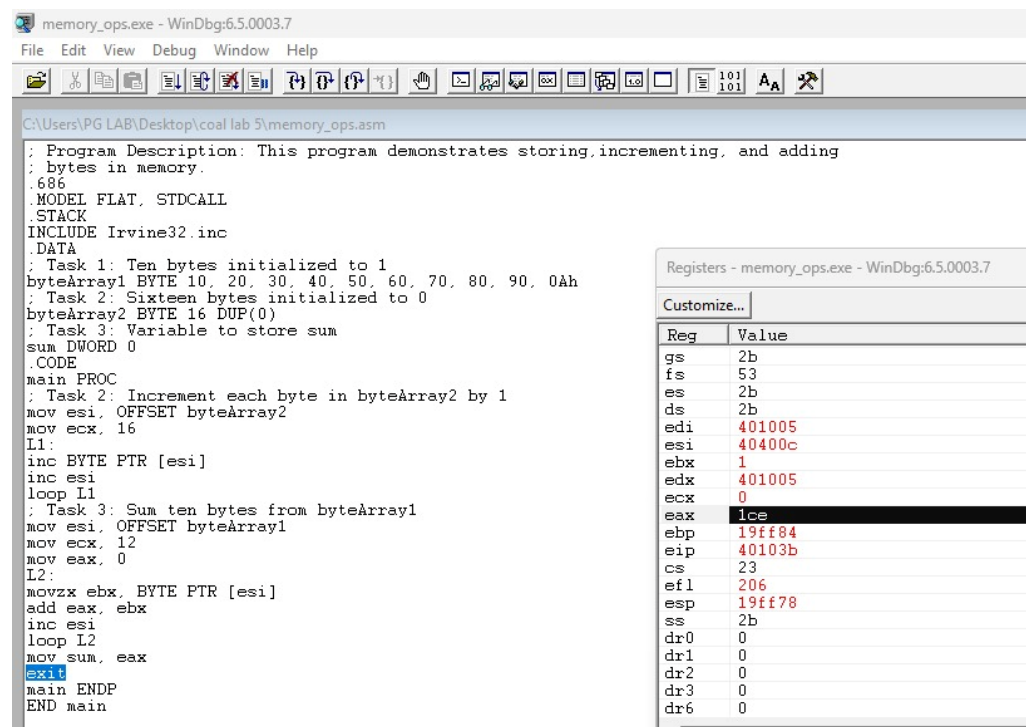
Assembling: .\memory_ops.asm
Microsoft (R) Incremental Linker Version 6.00.8447
Copyright (C) Microsoft Corp 1992-1998. All rights reserved.

PS C:\Users\PG LAB\Desktop\coal lab 5> windbg -QY -G memory_ops.exe
PS C:\Users\PG LAB\Desktop\coal lab 5> 
```

## Debugger output



### Output For Changed Value



[illegible][illegible]

- **ESI:** Used as a *source index* to read bytes from memory (arrays).
- **EDI:** Used as a *destination index* to write bytes to memory (not used in this code).

- Automatically **decrements ECX** and **jumps** if  $ECX \neq 0$ .
- Makes repeating through memory simpler and avoids manual decrement + jump instructions.

- **MOV AL, [esi]** → Reads a byte *from memory into AL*.
- **MOV [edi], AL** → Writes a byte *from AL into memory*.

- An 8-bit register would **overflow** (wrap around after 255).
- To avoid this, use a **32-bit register (EAX)** or check for carry (CF) after each add.

### **Q5. Modified code: sum only even-indexed bytes**

```
mov esi, OFFSET byteArray1
```

```
mov ecx, 5
```

```
mov eax, 0
```

```
SumEven:
```

```
    movzx ebx, BYTE PTR [esi]
```

```
    add eax, ebx
```

```
    add esi, 2      ; go to next even index
```

```
loop SumEven
```

```
mov sum, eax
```