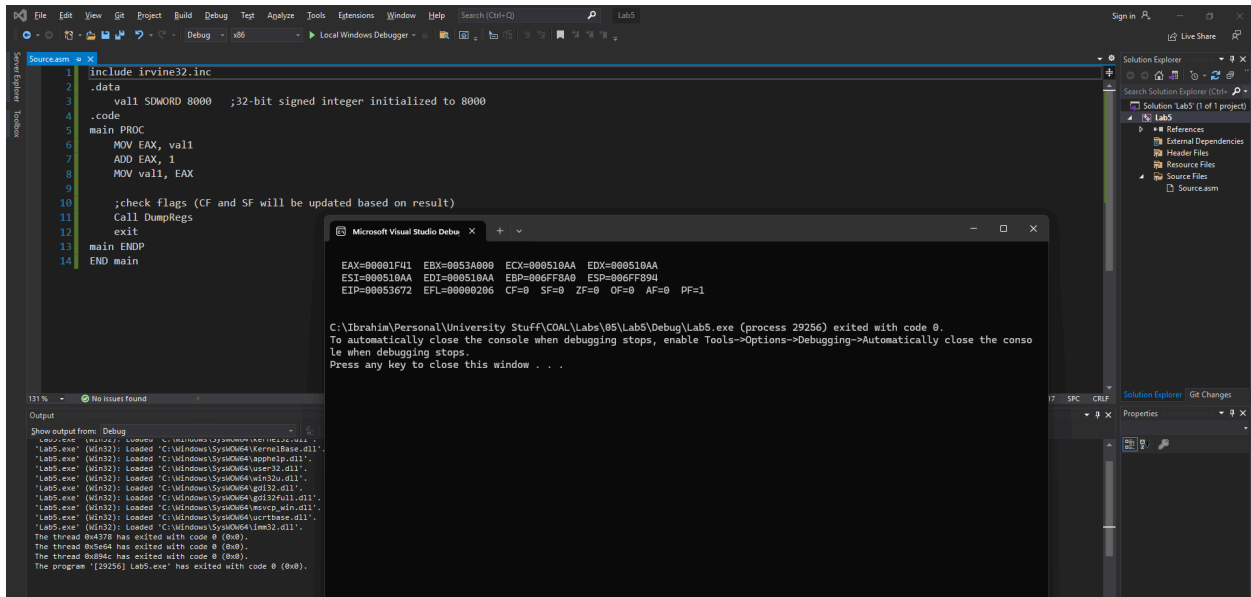


## COAL LAB 05 - 06 Mar 2025

Student Name: Ibrahim Johar Farooqi

Student ID: 23K-0074

### Task 1:



The screenshot shows the Visual Studio Code interface. The main editor displays assembly code for a program named 'Lab5.asm'. The code includes Irvine32.inc, defines a 32-bit signed integer 'val1' initialized to 8000, and contains a 'main' procedure. The 'main' procedure moves 'val1' into EAX, increments it by 1, and moves the result back into EAX. It then checks the Carry Flag (CF) and Sign Flag (SF) based on the result. The debugger window shows the current state of the CPU registers: EAX=00001F41, EBX=0053A000, ECX=000510AA, EDX=000510AA, ESI=000510AA, EDI=000510AA, EBP=006FF8A0, ESP=006FF894, EIP=00053672, EFL=00000206, CF=0, SF=0, ZF=0, OF=0, AF=0, PF=1. The console window shows the program's output, indicating that the program exited with code 0.

```
1 include irvine32.inc
2 .data
3     val1 SDWORD 8000 ;32-bit signed integer initialized to 8000
4 .code
5 main PROC
6     MOV EAX, val1
7     ADD EAX, 1
8     MOV val1, EAX
9
10    ;check flags (CF and SF will be updated based on result)
11    Call DumpRegs
12    exit
13 main ENDP
14 END main
```

Microsoft Visual Studio Debug Console:

```
EAX=00001F41 EBX=0053A000 ECX=000510AA EDX=000510AA
ESI=000510AA EDI=000510AA EBP=006FF8A0 ESP=006FF894
EIP=00053672 EFL=00000206 CF=0 SF=0 ZF=0 OF=0 AF=0 PF=1
```

C:\Ibrahim\Personal\University Stuff\COAL\Lab5\Debug\Lab5.exe (process 29256) exited with code 0. To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops. Press any key to close this window . . .

- Carry Flag (CF) = 0 (no unsigned overflow)
- Sign Flag (SF) = 0 (result is positive)

### Task 2:

MOV AX, 7FF0h

ADD AL, 10h

- a. CF = 1, SF = 0, ZF = 1, OF = 0

ADD AH, 1

- b. CF = 0, SF = 1, ZF = 0, OF = 1

ADD AX, 2

- c. CF = 0, SF = 1, ZF = 0, OF = 0

### Task 3:

```
Source.asm
1  include irvine32.inc
2  .data
3      unsortedArray BYTE 61, 43, 11, 52, 25 ; original unsorted array (BYTE)
4      sortedArray BYTE 5 DUP(?) ; empty array to store sorted values
5  .code
6  main PROC
7      mov sortedArray[0], 11
8      mov sortedArray[1], 25
9      mov sortedArray[2], 43
10     mov sortedArray[3], 52
11     mov sortedArray[4], 61
12     ; print sorted values
13     movzx eax, sortedArray[0]
14     call WriteDec
15     call Crlf
16     movzx eax, sortedArray[1]
17     call WriteDec
18     call Crlf
19     movzx eax, sortedArray[2]
20     call WriteDec
21     call Crlf
22     movzx eax, sortedArray[3]
23     call WriteDec
24     call Crlf
25     movzx eax, sortedArray[4]
26     call WriteDec
27     call Crlf
28
29     exit
30 main ENDP
31 END main
```

### Task 4:

```
File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q) Lab5
Debug x86 Local Windows Debugger
Source.asm
1  include irvine32.inc
2  .data
3      arrayB BYTE 10, 20, 30
4      arrayW WORD 150, 250, 350
5      arrayD DWORD 600, 1200, 1800
6
7      ; double word variables to store results
8      SUM1 DWORD ?
9      SUM2 DWORD ?
10     SUM3 DWORD ?
11 .code
12 main PROC
13     ; SUM1 = arrayB[0] + arrayW[0] + arrayD[0]
14     movzx eax, arrayB[0]
15     movzx ebx, arrayW[0]
16     mov ecx, arrayD[0]
17     add eax, ebx
18     add eax, ecx
19     mov SUM1, eax
20
21     ; SUM2 = arrayB[1] + arrayW[1] + arrayD[1]
22     movzx eax, arrayB[1]
23     movzx ebx, arrayW[1]
24     mov ecx, arrayD[1]
25     add eax, ebx
26     add eax, ecx
27     mov SUM2, eax
28
29     ; SUM3 = arrayB[2] + arrayW[2] + arrayD[2]
30     movzx eax, arrayB[2]
31     movzx ebx, arrayW[2]
32     mov ecx, arrayD[2]
33     add eax, ebx
34     add eax, ecx
35     mov SUM3, eax
36
37     exit
38 main ENDP
39 END main
40
```

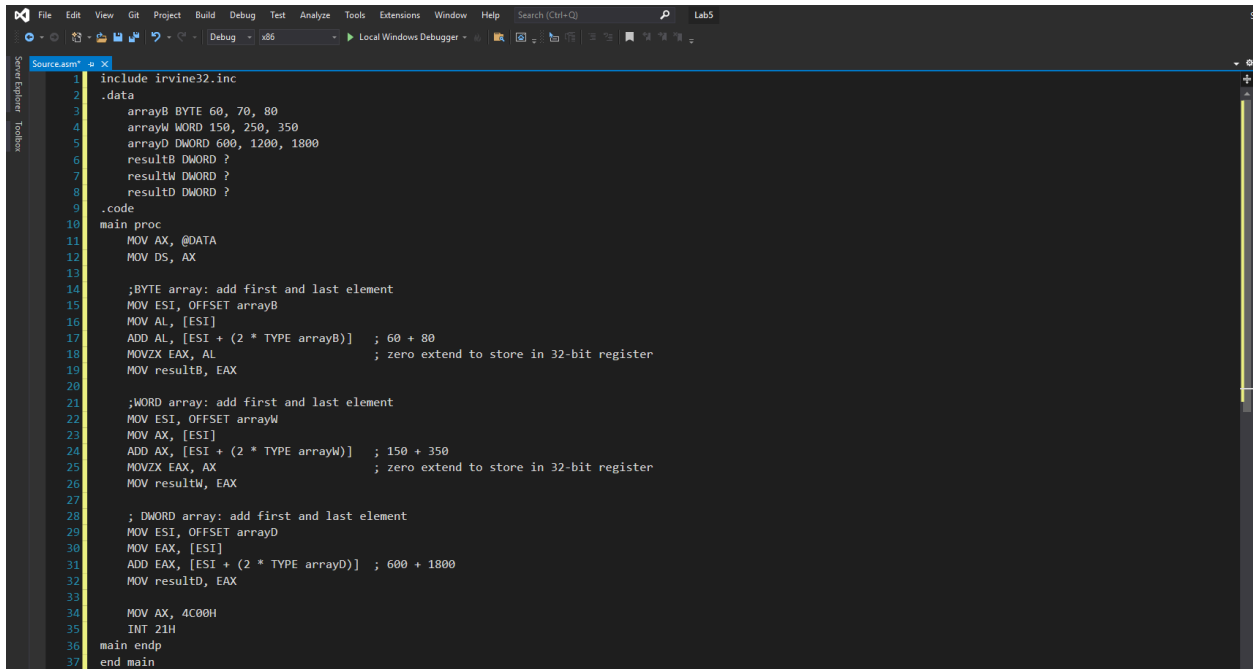
## Task 5:

```
Source.asm  X
1  include irvine32.inc
2  .data
3      array1  BYTE 10, 20, 30, 40
4      array2  BYTE 4 DUP(?)
5  .code
6  main PROC
7      mov esi, OFFSET array1
8      mov edi, OFFSET array2 + 3
9
10     mov al, [esi]
11     mov [edi], al
12     dec edi
13     inc esi
14
15     mov al, [esi]
16     mov [edi], al
17     dec edi
18     inc esi
19
20     mov al, [esi]
21     mov [edi], al
22     dec edi
23     inc esi
24
25     mov al, [esi]
26     mov [edi], al
27     exit
28 main ENDP
29 END main
30
```

## Task 6:

```
Source.asm  X
1  include irvine32.inc
2  .data
3      array  DWORD 100, 25, 10, 5, 2
4      result  DWORD ?
5
6  .code
7  main PROC
8      mov ESI, OFFSET array
9      mov EAX, [ESI]
10
11     add ESI, 4
12     sub EAX, [ESI]
13
14     add ESI, 4
15     sub EAX, [ESI]
16
17     add ESI, 4
18     sub EAX, [ESI]
19
20     add ESI, 4
21     sub EAX, [ESI]
22
23     mov result, EAX
24
25     exit
26 main ENDP
27 END main
```

## Task 7:



```
1 include irvine32.inc
2 .data
3     arrayB BYTE 60, 70, 80
4     arrayW WORD 150, 250, 350
5     arrayD DWORD 600, 1200, 1800
6     resultB DWORD ?
7     resultW DWORD ?
8     resultD DWORD ?
9 .code
10 main proc
11     MOV AX, @DATA
12     MOV DS, AX
13
14     ;BYTE array: add first and last element
15     MOV ESI, OFFSET arrayB
16     MOV AL, [ESI]
17     ADD AL, [ESI + (2 * TYPE arrayB)] ; 60 + 80
18     MOVZX EAX, AL ; zero extend to store in 32-bit register
19     MOV resultB, EAX
20
21     ;WORD array: add first and last element
22     MOV ESI, OFFSET arrayW
23     MOV AX, [ESI]
24     ADD AX, [ESI + (2 * TYPE arrayW)] ; 150 + 350
25     MOVZX EAX, AX ; zero extend to store in 32-bit register
26     MOV resultW, EAX
27
28     ;DWORD array: add first and last element
29     MOV ESI, OFFSET arrayD
30     MOV EAX, [ESI]
31     ADD EAX, [ESI + (2 * TYPE arrayD)] ; 600 + 1800
32     MOV resultD, EAX
33
34     MOV AX, 4C00H
35     INT 21H
36 main endp
37 end main
```