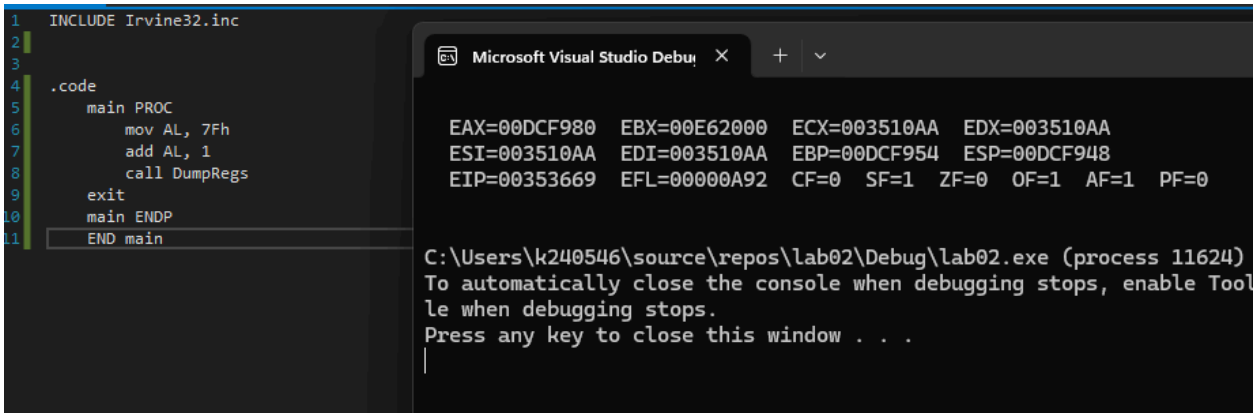
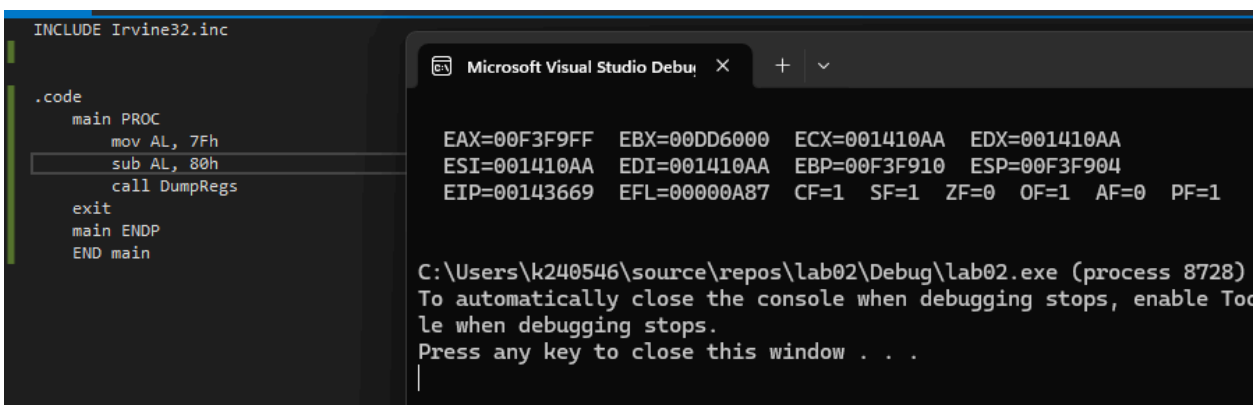
COAL LAB 5 - 24K0546

Task 1



ZF = 0, SF = 1, CF = 0, and OF = 1

```
INCLUDE Irvine32.inc
.code
        main PROC
                mov AL, 7Fh
                add AL, 1
                call DumpRegs
        exit
        main ENDP
        END main
```



ZF = 0, SF = 1, CF = 1, and OF = 1

```
INCLUDE Irvine32.inc
.code
        main PROC
                mov AL, 7Fh
                sub AL, 80h
                call DumpRegs
        exit
```

main ENDP
END main

Task 2

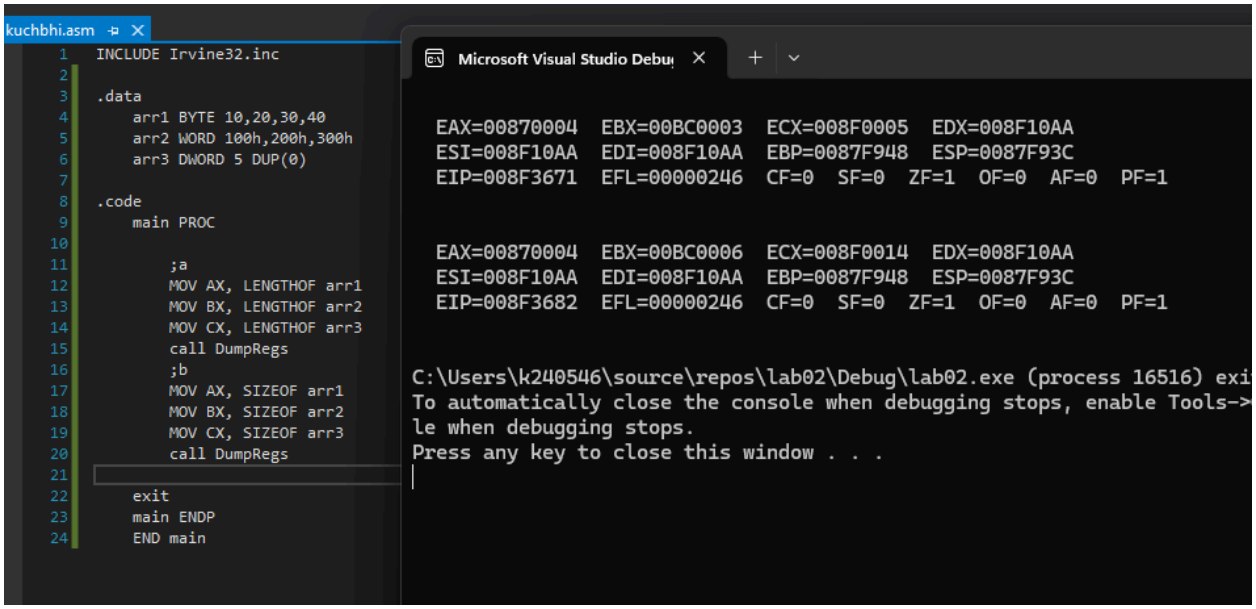
```
EAX=012FFF94  EBX=01001000  ECX=00CC10AA  EDX=00CC10AA
ESI=00CC6000  EDI=00CC10AA  EBP=012FFF48  ESP=012FFF3C
EIP=00CC366A  EFL=00000246  CF=0  SF=0  ZF=1  OF=0  AF=0  PF=1

EAX=012FFF94  EBX=01001000  ECX=00CC10AA  EDX=00CC10AA
ESI=00CC6001  EDI=00CC10AA  EBP=012FFF48  ESP=012FFF3C
EIP=00CC3674  EFL=00000246  CF=0  SF=0  ZF=1  OF=0  AF=0  PF=1

EAX=012FFF94  EBX=01001000  ECX=00CC10AA  EDX=00CC10AA
ESI=00CC6003  EDI=00CC10AA  EBP=012FFF48  ESP=012FFF3C
EIP=00CC367E  EFL=00000246  CF=0  SF=0  ZF=1  OF=0  AF=0  PF=1

EAX=012F1234  EBX=01001000  ECX=00CC10AA  EDX=00CC10AA
ESI=00CC6003  EDI=00CC10AA  EBP=012FFF48  ESP=012FFF3C
EIP=00CC3689  EFL=00000246  CF=0  SF=0  ZF=1  OF=0  AF=0  PF=1

EAX=012F1234  EBX=01000001  ECX=00CC10AA  EDX=00CC10AA
ESI=00CC6003  EDI=00CC10AA  EBP=012FFF48  ESP=012FFF3C
EIP=00CC3692  EFL=00000246  CF=0  SF=0  ZF=1  OF=0  AF=0  PF=1

EAX=012F1234  EBX=01000002  ECX=00CC10AA  EDX=00CC10AA
ESI=00CC6003  EDI=00CC10AA  EBP=012FFF48  ESP=012FFF3C
EIP=00CC369B  EFL=00000246  CF=0  SF=0  ZF=1  OF=0  AF=0  PF=1

EAX=012F1234  EBX=01000004  ECX=00CC10AA  EDX=00CC10AA
ESI=00CC6003  EDI=00CC10AA  EBP=012FFF48  ESP=012FFF3C
EIP=00CC36A4  EFL=00000246  CF=0  SF=0  ZF=1  OF=0  AF=0  PF=1

C:\Users\user\source\repos\Lab5\Debug\Lab5.exe (process 14668) exited with code 0 (0x0).
To automatically close the console when debugging stops, enable Tools->Options->Debuggin
Press any key to close this window . . .
```

```
INCLUDE Irvine32.inc
.data
    myByte  BYTE 12h
    myWord  WORD 1234h
    myDword DWORD 12345678h

.code
main PROC
    ;a
    MOV ESI, OFFSET myByte
    call DumpRegs
    MOV ESI, OFFSET myWord
    call DumpRegs
    MOV ESI, OFFSET myDword
    call DumpRegs

    ;b
    MOV AX, WORD PTR myDword + 2
    call DumpRegs

    ;c
    MOV BX, TYPE myByte
    call DumpRegs
    MOV BX, TYPE myWord
    call DumpRegs
    MOV BX, TYPE myDword
    call DumpRegs

    exit
main ENDP
END main
```

No issues found

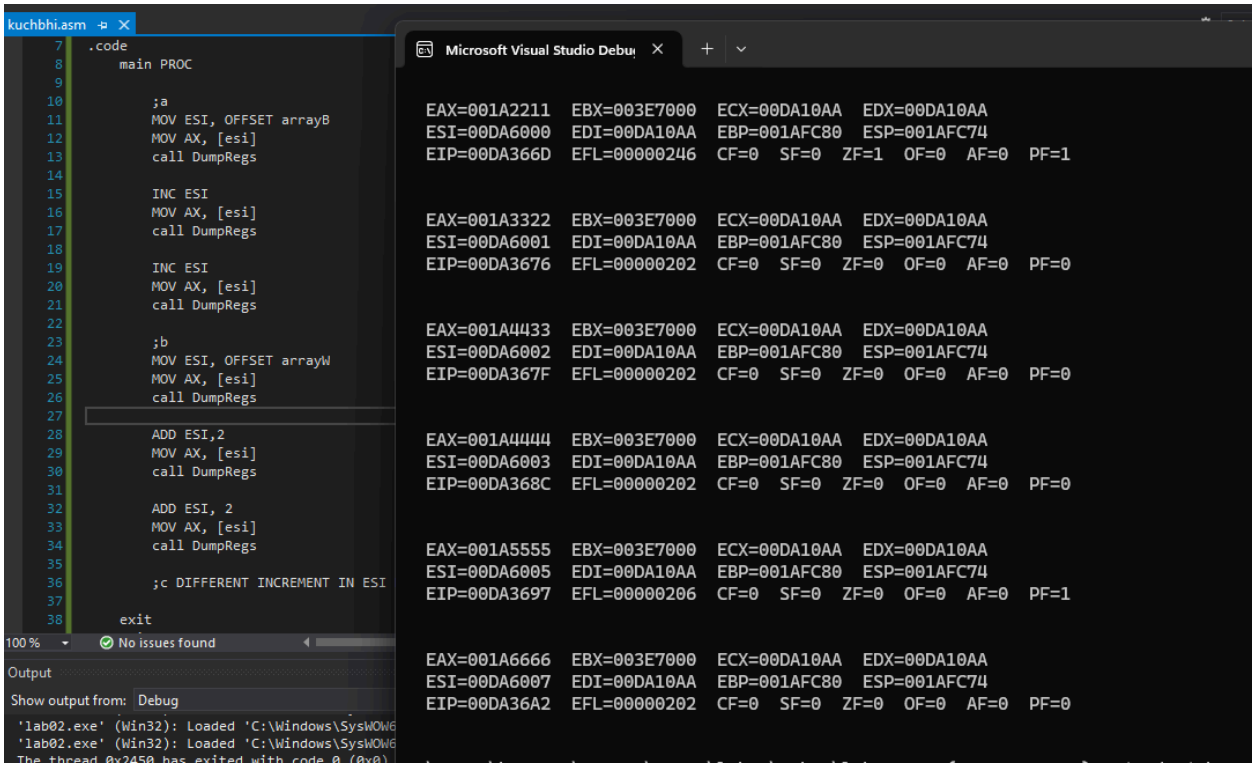
```
INCLUDE Irvine32.inc

.data
	myByte BYTE 12h
	myWord WORD 1234h
	myDword DWORD 12345678h

.code
	main PROC

		;a)
		MOV ESI, OFFSET myByte
		MOV ESI, OFFSET myWord
		MOV ESI, OFFSET myWord

		;b)
		MOV AX, WORD PTR myDword + 2

		;c)
		MOV BX, TYPE myByte
		MOV BX, TYPE myWord
		MOV BX, TYPE myDword

		call DumpRegs

	exit
	main ENDP
	END main
```

Task 3




```
INCLUDE Irvine32.inc

.data
        arr1 BYTE 10,20,30,40
        arr2 WORD 100h,200h,300h
        arr3 DWORD 5 DUP(0)

.code
        main PROC

                ;a
                MOV AX, LENGTHOF arr1
                MOV BX, LENGTHOF arr2
                MOV CX, LENGTHOF arr3
                call DumpRegs
                ;b
                MOV AX, SIZEOF arr1
                MOV BX, SIZEOF arr2
                MOV CX, SIZEOF arr3
                call DumpRegs

        exit
        main ENDP
        END main
```

Task 4

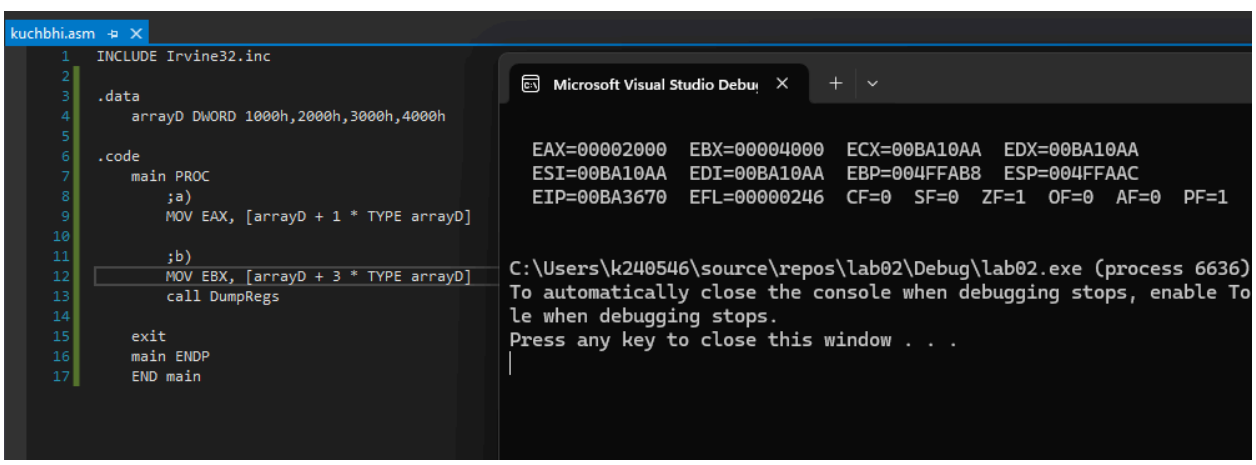


C. DIFFERENT INCREMENT IN ESI BECAUSE SIZE OF BOTH ARRAYS IS DIFFERENT (ArrayB is 1 byte only, while ArrayW is 2 bytes)

```
INCLUDE Irvine32.inc

.data
        arrayB BYTE 11h,22h,33h
        arrayW WORD 4444h,5555h,6666h

.code
        main PROC

                ;a
                MOV ESI, OFFSET arrayB
                MOV AX, [esi]
                call DumpRegs

                INC ESI
                MOV AX, [esi]
                call DumpRegs

                INC ESI
                MOV AX, [esi]
```

```
call DumpRegs

;b
MOV ESI, OFFSET arrayW
MOV AX, [esi]
call DumpRegs

ADD ESI,2
MOV AX, [esi]
call DumpRegs

ADD ESI, 2
MOV AX, [esi]
call DumpRegs

;c DIFFERENT INCREMENT IN ESI BECAUSE SIZE OF BOTH ARRAYS IS DIFFERENT (ArrayB is 1 byte only, while ArrayW is 2 bytes)

exit
main ENDP
END main
```

Task 5



c) The type operator returns the size in bytes of one element in the array thus allows correct calculation of the byte offset for indexed addressing. It makes sure that the scaled index points precisely to the desired array element.

```
INCLUDE Irvine32.inc

.data
    arrayD DWORD 1000h,2000h,3000h,4000h
```

```
.code
	main PROC
		;a)
		MOV EAX, [arrayD + 1 * TYPE arrayD]

		;b)
		MOV EBX, [arrayD + 3 * TYPE arrayD]
		call DumpRegs

	exit
	main ENDP
	END main
```