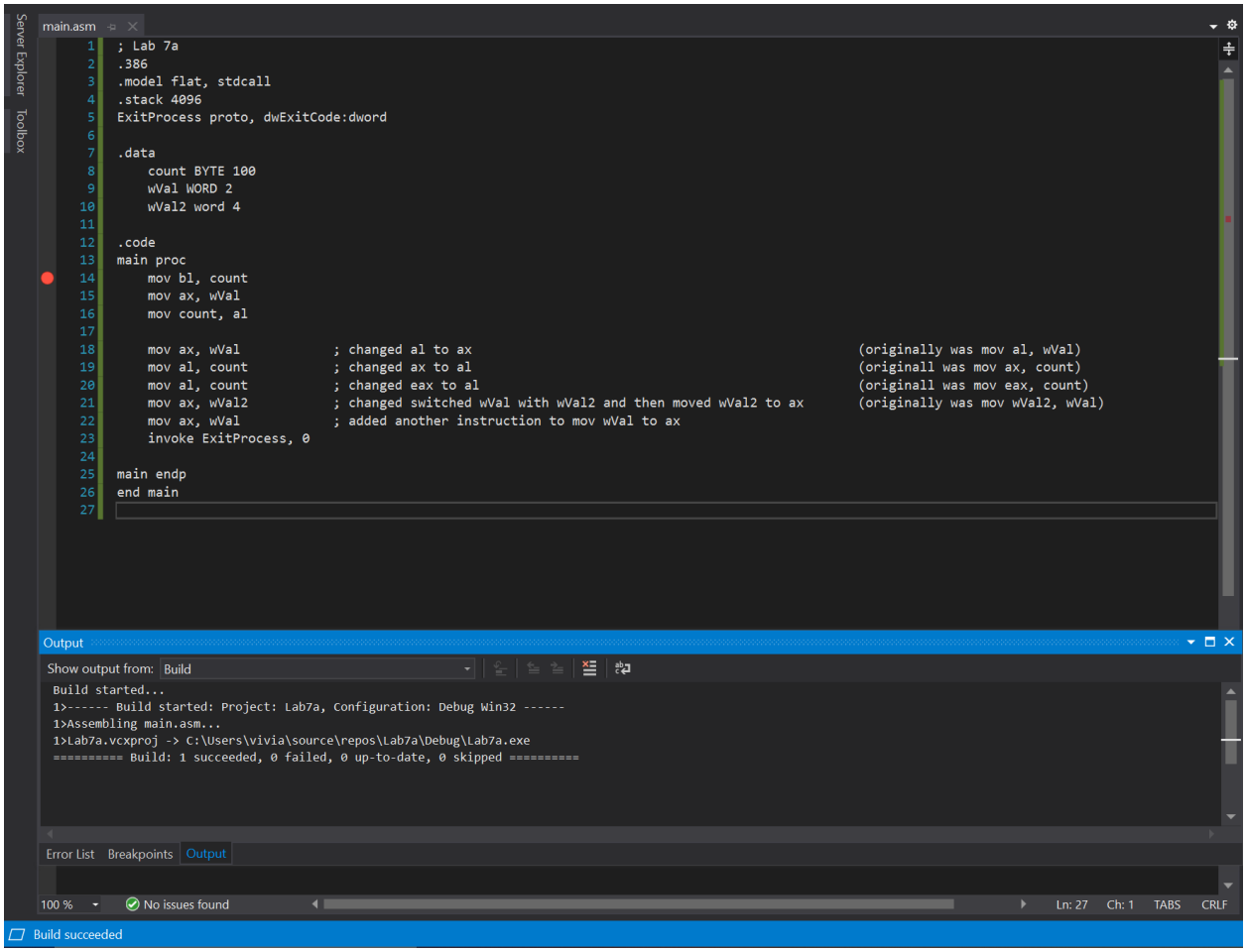
CSC 3210
Computer Organization and Programming
Lab 7
Answer Sheet

Student Name: Vivian Do
Section: CRN 90913; 11:00-12:40
Oct 08 2021

Lab 7(a)

Fix the errors in the provided code.
Build and Attach screenshot showing the code and “build succeeded” message.



Lab 7(b)

Debug through each line of instructions.
Take screenshot that includes code and register window.
Record the register content.

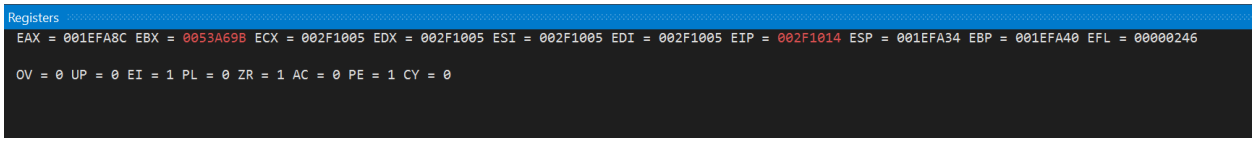
and explain the register contents.

Code:

main.asm

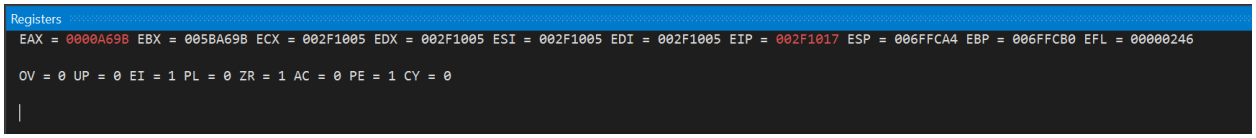
```
1  ; Lab 7b
2  .386
3  .model flat, stdcall
4  .stack 4096
5  ExitProcess proto dwExitCode:dword
6
7  .data
8      myByte1 BYTE 9Bh
9  .code
10 main proc
11     mov bx, 0A69Bh
12     movzx eax, bx
13                          ; EAX = 0000A69B
14     movzx eax, myByte1
15                          ; EAX = 0000009B
16     mov bx, 0A69Bh
17     movsx eax, bx
18                          ; EAX = FFFFA69B
19     invoke ExitProcess, 0
20
21 main endp
22 end main
23
```

Line number: 11
Instruction: mov bx, 0A69Bh
Register Values: EBX = 0000A69B
Screenshot:



Explanation: EBX register is 32-bit long with an unsigned integer variable. This register is updated with 0A69Bh in its contents by *mov*.

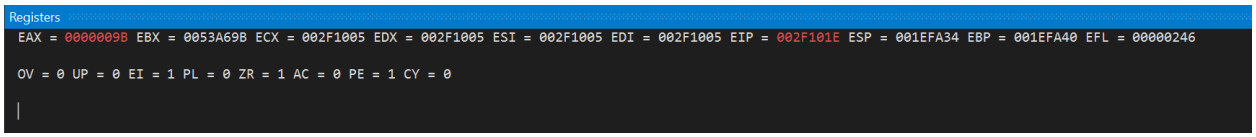
Line number: 12
Instruction: movzx eax, bx
Register Values: EAX = 0000A69B
Screenshot:



Explanation: EAX register is 32-bit long with a signed integer variable. This register is updated with EBX (0A69Bh) while extending the remaining upper half with zeros.

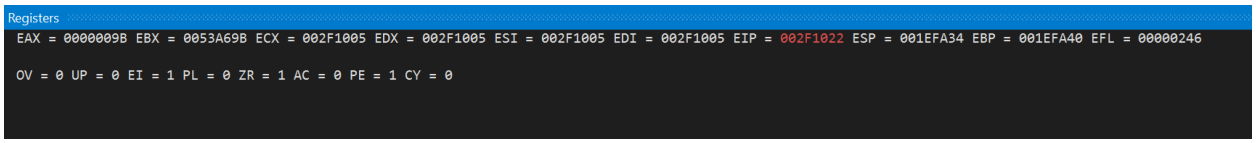
(Copy paste this format if you need more)

Line number: 14
Instruction: movzx eax, myByte1
Register Values: EAX = 0000009B
Screenshot:



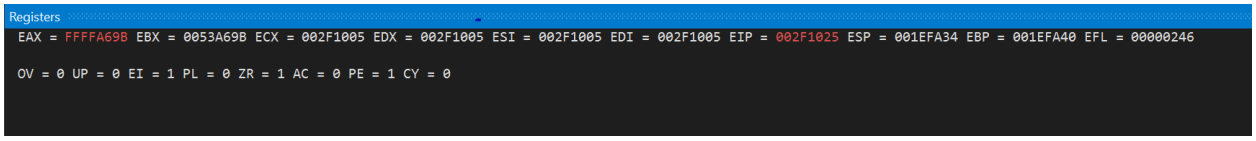
Explanation: The EAX register is updated with myByte1 value (9Bh) while extending the remaining upper half with zeros.

Line number: 16
Instruction: mov bx, 0A69B
Register Values: EAX = 0000A69B
Screenshot:



Explanation: Nothing has changed. Refer back to the explanation for line 11.

Line number: 17
Instruction: movsx eax, bx
Register Values: EAX = FFFFA69B
Screenshot:



Explanation: The EAX register is updated with the contents from the bx register. The leading A tells us that it is the highest bit in set. The rest of the upper half of the register is extended with F's.

Lab 7(c)

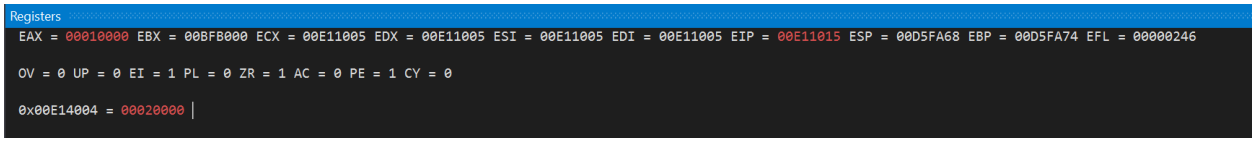
Debug through each line of instructions.
Take screenshot that includes code and register window.
Record the register content.
and explain the register contents.

Code:

main.asm

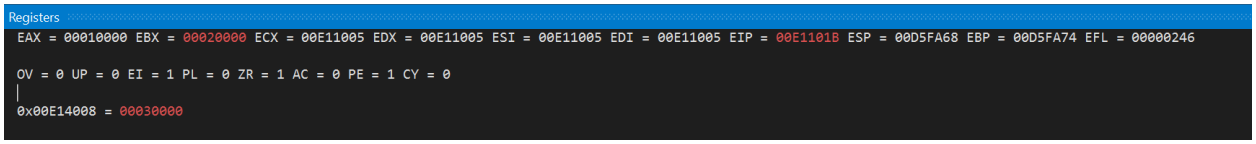
```
1   .386
2   .model flat, stdcall
3   .stack 4096
4   ExitProcess proto dwExitCode:dword
5
6   .data
7       arrayD DWORD 10000h, 20000h, 30000h
8
9   .code
10  main proc
11      ; direct-offset addressig (doubleword array):
12      mov eax, arrayD             ; EAX = 00010000
13      mov ebx, [arrayD + 4]       ; EBX = 00020000
14      mov edx, [arrayD + 8]       ; EDX = 00030000
15      invoke ExitProcess, 0
16
17  main endp
18  end main
19
```

Line number: 12
Instruction: mov eax, arrayD
Register Values: EAX = 00010000
Screenshot:



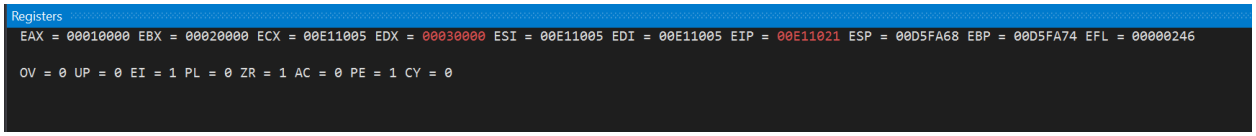
Explanation: EAX register is 32-bit long with a signed integer variable. This register is updated with arrayD (10000h) in its contents by *mov*.

Line number: 13
Instruction: mov ebx, [arrayD + 4]
Register Values: EBX = 00020000
Screenshot:



Explanation: EBX register is 32-bit long with an unsigned integer variable. This register is updated with 200000h. By adding 4 to arrayD, we are accessing the array element that is 4 bytes offset to the first one.

Line number: 14
Instruction: mov edx, [arrayD + 8]
Register Values: EDX = 00030000
Screenshot:

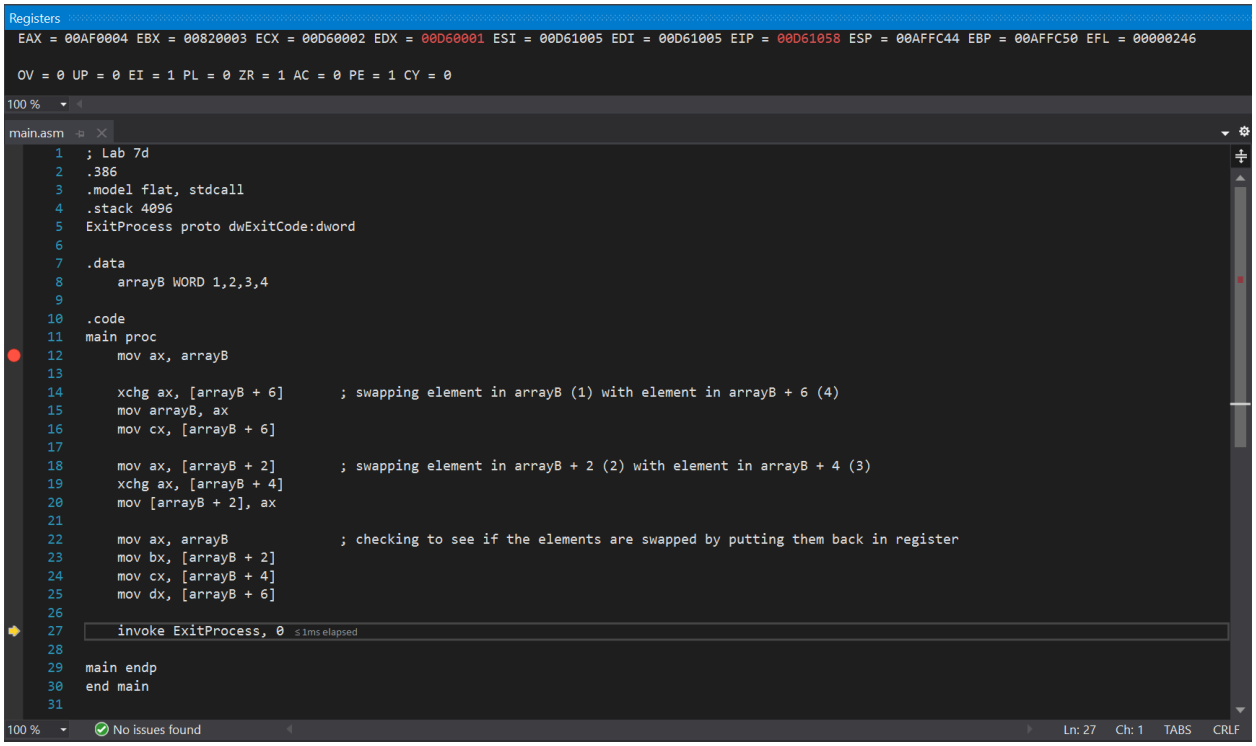


Explanation: EDX register is updated with 300000h. By adding 8 to arrayD, we are accessing the array element that is 8 bytes away from the first byte.

Lab 7(d)

Create a new project to run the following program.
Declare an array in the data segment: arrayB WORD 1,2,3,4
Write code to Rearrange the array as follows: 4,3,1,2

Add the screenshot of your code here.
The registers EAX, EBX, ECX, and EDX at the end show the rearranged array.



Lab 7(e)

Create a new application to run the following program.
The data segment is provided:

.data

Val1 SWORD 23
Val2 SWORD -35
Val3 SDWORD 4

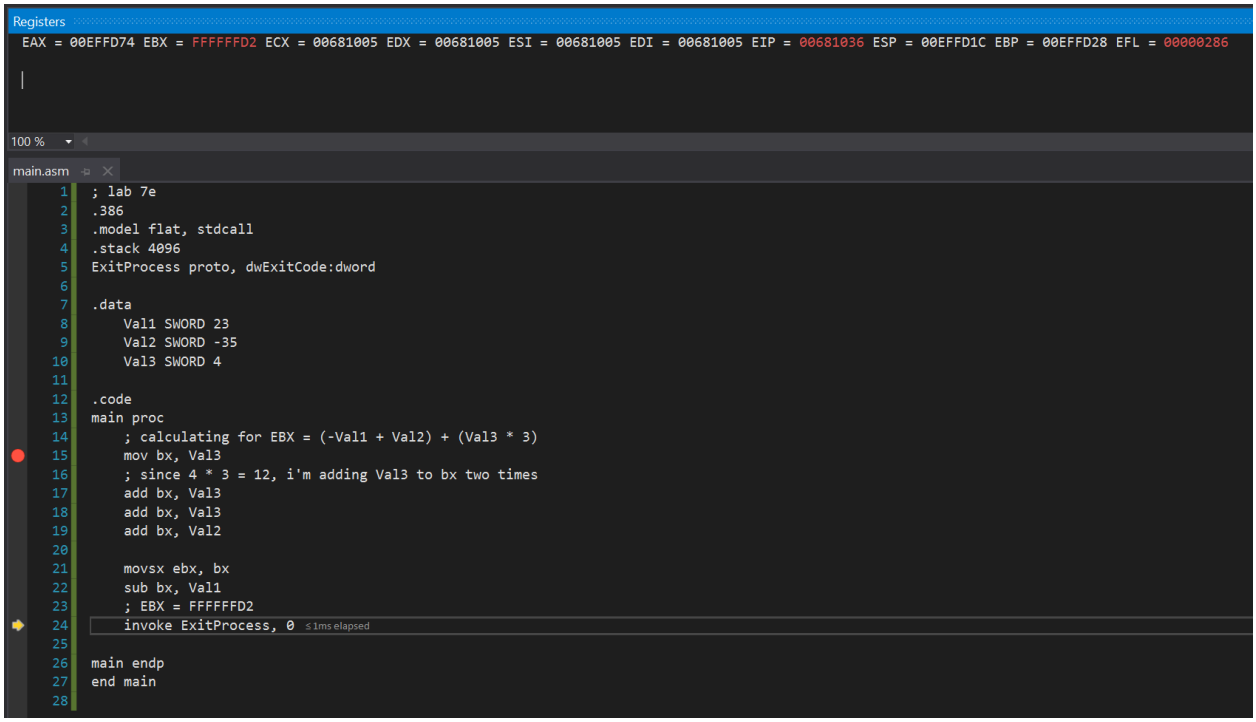
Evaluate the following expression:

EBX = (-Val1 + val2) + (val3*3)

You can only use Mov, Movsz, Movzx, Add, Sub instructions.

Build and run the program using the debugger

Debug the code until you reach “INVOKE ExitProcess, 0” and attach a screenshot of your code and EBX register content.


Registers
EAX = 00EFFD74 EBX = FFFFFFD2 ECX = 00681005 EDX = 00681005 ESI = 00681005 EDI = 00681005 EIP = 00681036 ESP = 00EFFD1C EBP = 00EFFD28 EFL = 00000286
100 %
main.asm
; lab 7e
.386
.model flat, stdcall
.stack 4096
ExitProcess proto, dwExitCode:dword
.data
Val1 SWORD 23
Val2 SWORD -35
Val3 SWORD 4
.code
main proc
; calculating for EBX = (-Val1 + Val2) + (Val3 * 3)
mov bx, Val3
; since 4 * 3 = 12, i'm adding Val3 to bx two times
add bx, Val3
add bx, Val3
add bx, Val2
movsx ebx, bx
sub bx, Val1
; EBX = FFFFFFD2
invoke ExitProcess, 0
≤1ms elapsed
main endp
end main