**Computer Organization and Assembly Language**

**Question 1:**

Write an assembly language code of the following C++ function. Consider all the integers of this procedure as unsigned DWORD in Assembly Language. Write two types of Assembly code one using the CALL instruction and other with INVOKE instruction from the main body of Assembly Language. Use accumulator to return the values from functions. Show the output from main body. Paste the Output Screens of Assembly Language for both procedures.

int subtr (int x, int y) {

int var1=10; int var2=50;

int var4;

var4 = var1 + var2; return var4 – ( x + y);

}

void main () {

:

:

int a = subtr ( 12, 10 )

:

:

}

**Solution:**

**(with CALL instruction)**

include Irvine32.inc

.data

intX DWORD 0

intY DWORD 0

msg0 BYTE "Enter value 1: ", 0

msg1 BYTE "Enter value 2: ", 0

msg2 BYTE "Final value: ", 0

.code

main PROC

mov eax, 0

call crlf

mov edx, OFFSET msg0

call WriteString

call ReadInt

push eax

mov intX, eax

call crlf

mov eax, 0

call crlf

mov edx, OFFSET msg1

call WriteString

call ReadInt

push eax

mov intY, eax

call crlf

call crlf

call crlf

call Subtr

mov edx, OFFSET msg2

call WriteString

call WriteInt

call crlf

exit

main ENDP

Subtr PROC

;----------------------------------------------------------------------------------

;Function subtracts the sum of input arguments from 60 and returns the result

;Receives: two input values from the user via stack

;Requires: nothing

;Returns: resultant value in the EAX register

;----------------------------------------------------------------------------------

push ebp

mov ebp, esp

mov eax, 0

add eax, 10

add eax, 50

mov ebx, 0

mov ebx, [ebp + 12]

add ebx, [ebp + 8]

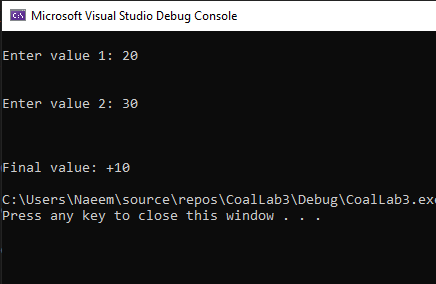
sub eax, ebx

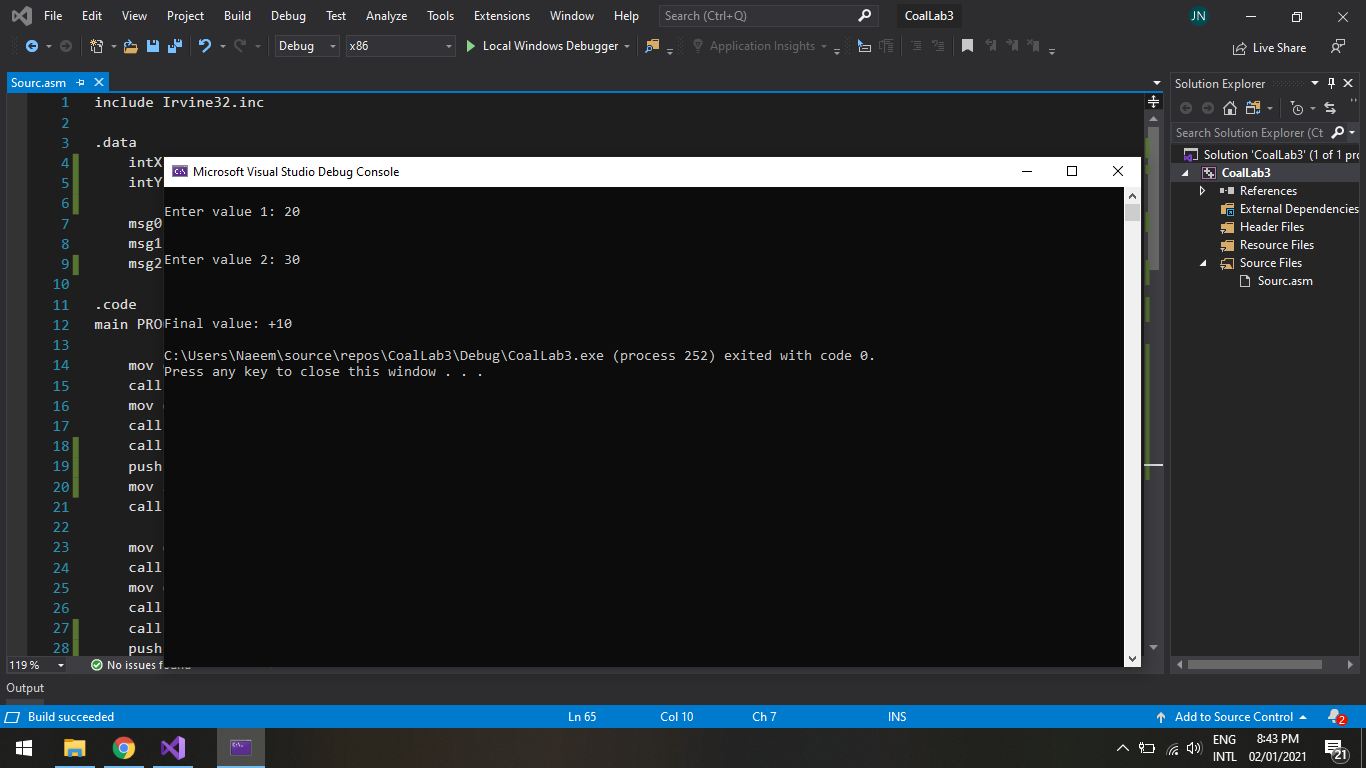
pop ebp

ret 8 ;cleaning up the stack

Subtr ENDP

END main





**(with INVOKE instruction)**

include Irvine32.inc

Subtr PROTO,

v1: PTR DWORD,

v2: PTR DWORD

.data

intX DWORD 0

intY DWORD 0

msg0 BYTE "Enter value 1: ", 0

msg1 BYTE "Enter value 2: ", 0

msg2 BYTE "Final value: ", 0

.code

main PROC

mov eax, 0

call crlf

mov edx, OFFSET msg0

call WriteString

call ReadInt

mov intX, eax

call crlf

mov eax, 0

call crlf

mov edx, OFFSET msg1

call WriteString

call ReadInt

mov intY, eax

call crlf

call crlf

call crlf

INVOKE Subtr, intX, intY ;Assembler was giving wrong answer with ADDR directive

mov edx, OFFSET msg2

call WriteString

call WriteInt

call crlf

exit

main ENDP

Subtr PROC,

v1: PTR DWORD,

v2: PTR DWORD

;----------------------------------------------------------------------------------

;Function subtracts the sum of input arguments from 60 and returns the result

;Receives: two input values from the user via variables

;Requires: nothing

;Returns: resultant value in the EAX register

;----------------------------------------------------------------------------------

mov eax, 0

add eax, 10

add eax, 50

mov ebx, 0

add ebx, v1

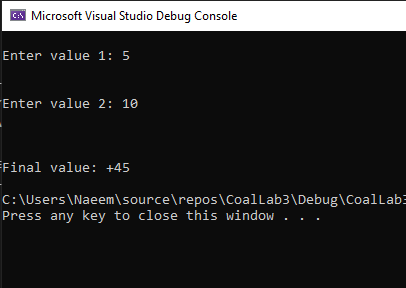
add ebx, v2

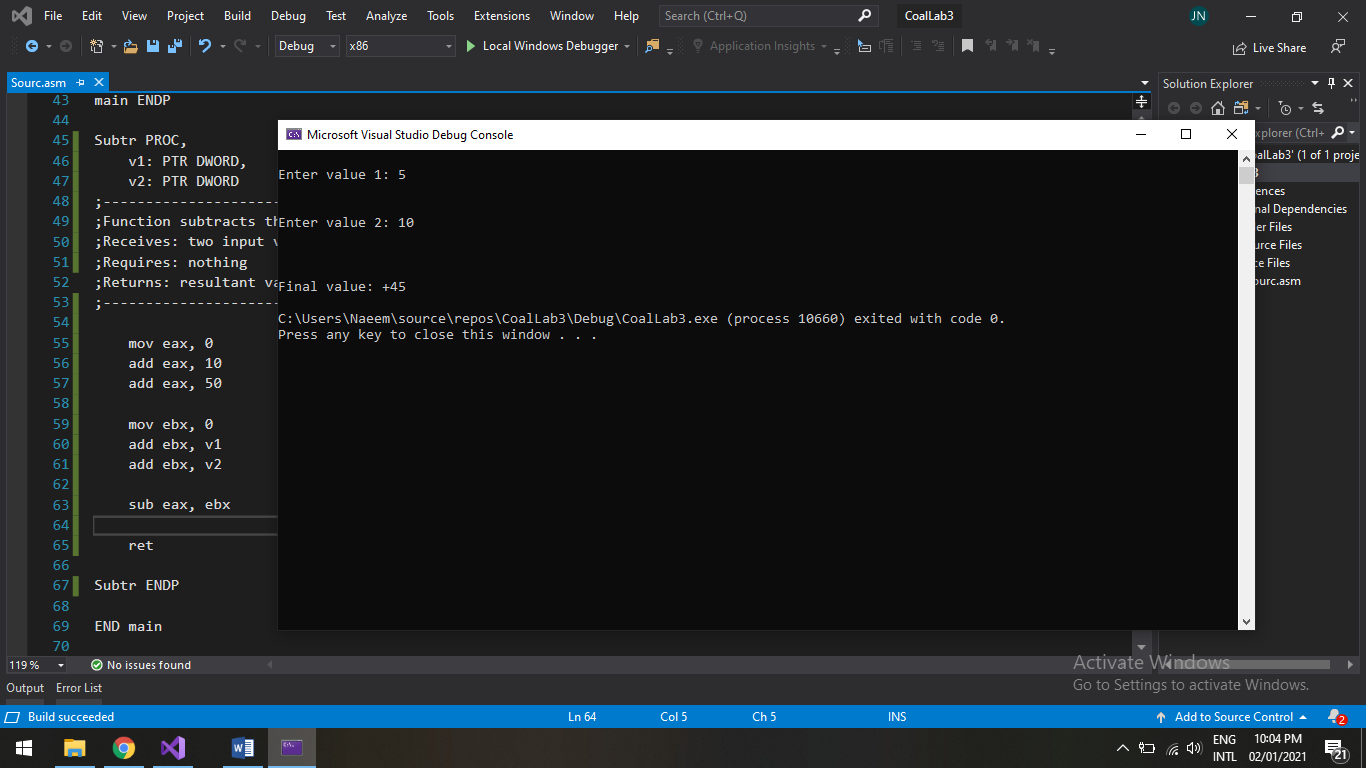
sub eax, ebx

ret

Subtr ENDP

END main





ISSUE WITH USING ADDR:

