Department of Computer Science
Faculty of Computing
UNIVERSITI TEKNOLOGI MALAYSIA

SUBJECT NAME:	**COMPUTER ORGANIZATION AND ARCHITECTURE**
SUBJECT CODE:	**SECR 1033**
SEMESTER:	**2 - 2023/24**
LAB TITLE:	**Programming 1: Assembly Language Fundamentals**
INSTRUCTION:	**Student is required to have instructor's signature before proceed from each lab work.**
STUDENT INFO :	**Name: Nur Firzana Binti Badrus Hisham(A23CS0156)** **Lubna Al Haani Binti Radzuan(A23CS0107)** **Section: 03** **Email:** *nurfirzana@graduate.utm.my* *lubnaalhaani@graduate.utm.my*
SUBMITTED DATE:	__________________

COMMENTS:

Lab # 1

Simple program to familiarize with Program Code, Rebuild & Start Without Debugging

Execute the programs below:

PART 1:

i. **Part A**: Adding and Subtracting Integers

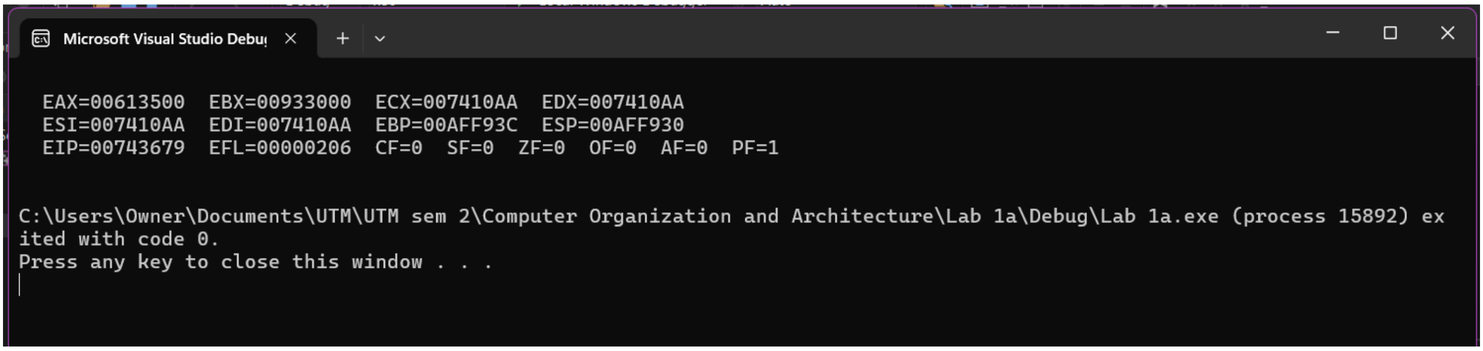
```
TITLE Add and Subtract (AddSub.asm)

; This program adds and subtracts 32-bit integers
; Authors:
; Date:
; Revision:

INCLUDE Irvine32.inc

.data
TOTAL dword 0 ; a variable named TOTAL (declared as DWORD)

.code

main PROC

    mov eax, 123400h ; Set EAX with the value of 123400h
    add eax, 567800h ; Add the content of EAX with 567800h
    sub eax, 77700h ; Subtract content of EAX with 77700h
    mov TOTAL, eax  ; Store content of EAX to TOTAL

    call DumpRegs

    exit
main ENDP

END main
```

Experimental Results:

a) Rebuild & Start Without Debugging

Completed   Fully   Partially   : Checked by: ______________

Screenshot Result:



```
  EAX=00613500  EBX=00933000  ECX=007410AA  EDX=007410AA
  ESI=007410AA  EDI=007410AA  EBP=00AFF93C  ESP=00AFF930
  EIP=00743679  EFL=00000206  CF=0  SF=0  ZF=0  OF=0  AF=0  PF=1


C:\Users\Owner\Documents\UTM\UTM sem 2\Computer Organization and Architecture\Lab 1a\Debug\Lab 1a.exe (process 15892) ex
ited with code 0.
Press any key to close this window . . .
```

ii. **Part B**: Adding Variables

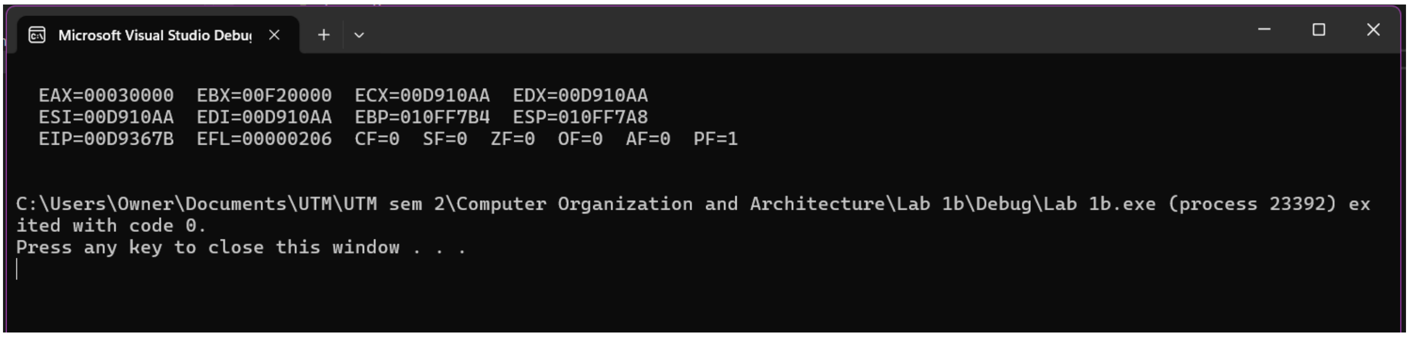
```
TITLE Add and Subtract, Version 2(AddSub2.asm)

; This program adds and subtracts 32-bit unsigned
; integers and stores the sum in a variable.
; Authors:
; Date:
; Revision:

INCLUDE Irvine32.inc
.data
val1 DWORD 10000h
val2 DWORD 40000h
val3 DWORD 20000h
finalVal DWORD ?
.code

main PROC
mov eax,val1     ; start with 10000h
add eax,val2     ; add 40000h
sub eax,val3     ; subtract 20000h
mov finalVal,eax ; store the result (30000h)

call DumpRegs    ; display the registers

exit

main ENDP
END main
```

Experimental Results:

a) Rebuild & Start Without Debugging

Completed   *Fully*   *Partially* *: Checked by:* ______________

Screenshot Result:



```
  EAX=00030000  EBX=00F20000  ECX=00D910AA  EDX=00D910AA
  ESI=00D910AA  EDI=00D910AA  EBP=010FF7B4  ESP=010FF7A8
  EIP=00D9367B  EFL=00000206  CF=0  SF=0  ZF=0  OF=0  AF=0  PF=1


C:\Users\Owner\Documents\UTM\UTM sem 2\Computer Organization and Architecture\Lab 1b\Debug\Lab 1b.exe (process 23392) ex
ited with code 0.
Press any key to close this window . . .
```

iii. **Part C**: Add and Subtract 8 and 16-Bit Version

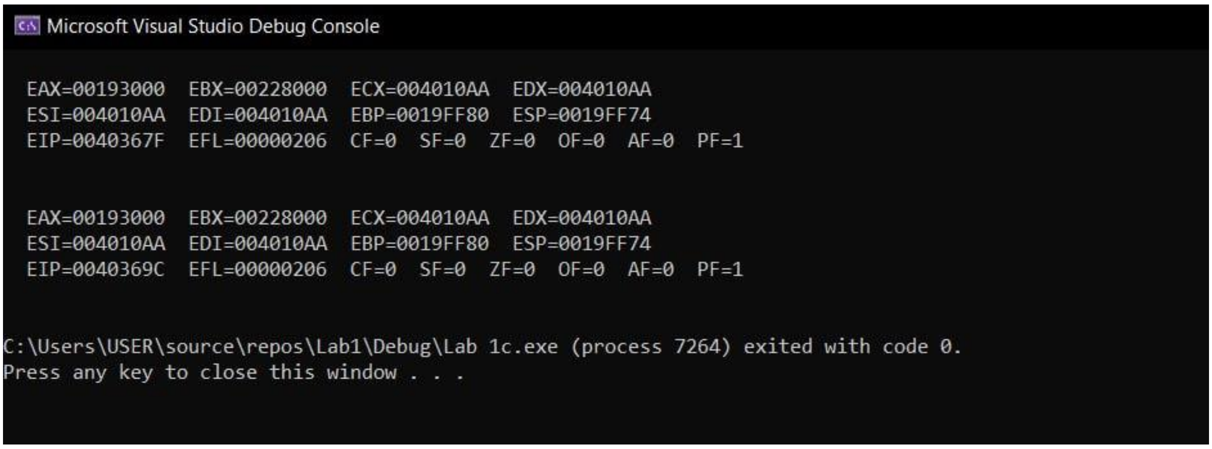
```
TITLE Add and Subtract, Version 3
; This program adds and subtracts 8 and 16 bit
; unsigned integers and stores the sum in a variable.
; Authors:
; Date:
; Revision:

INCLUDE Irvine32.inc
.data
valw1 WORD 1000h
valw2 WORD 4000h
valw3 WORD 2000h
finalValw WORD ?

valb1 BYTE 10h
valb2 BYTE 40h
valb3 BYTE 20h
finalValb BYTE ?

.code
main PROC
mov ax,valw1; start with 10000h
add ax,valw2       ; add 40000h
sub ax,valw3       ; subtract 20000h
mov finalValw,ax   ; store the result (30000h)
call DumpRegs      ; display the registers

mov ah,valb1; start with 10000h
add ah,valb2       ; add 40000h
sub ah,valb3       ; subtract 20000h
mov finalValb,ah   ; store the result (30000h)
call DumpRegs      ; display the registers

exit
main ENDP
END main
```

Experimental Results:

a) Rebuild & Start Without Debugging

Completed   *Fully*   *Partially*   *: Checked by: _____________*

Screenshot Result:



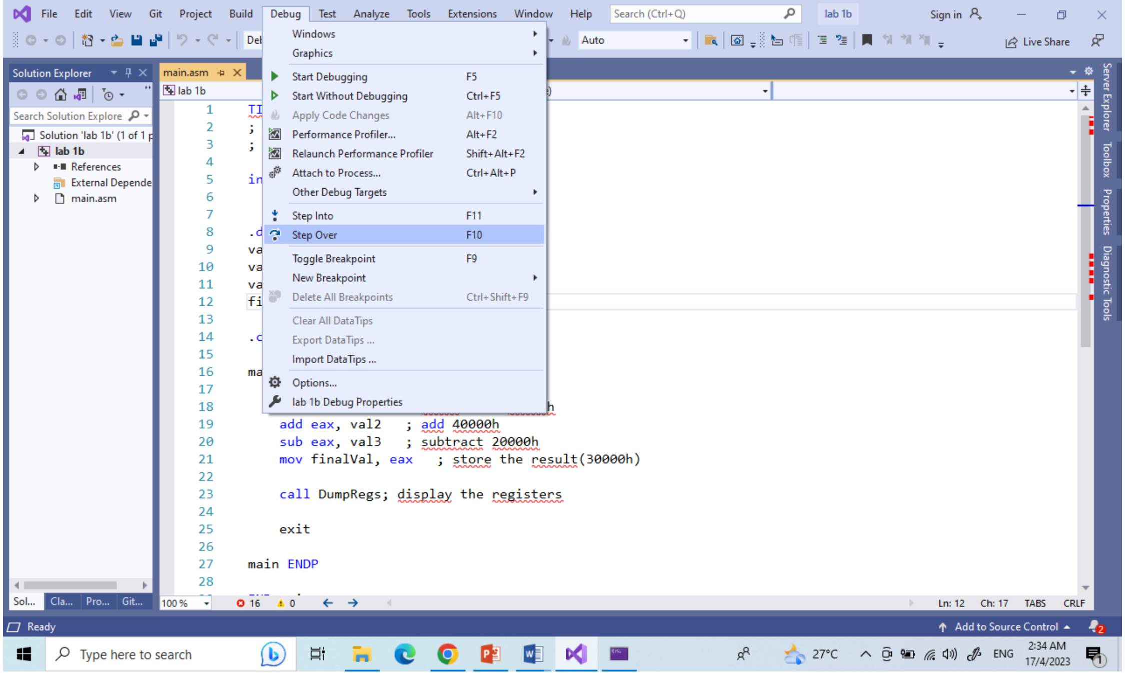
```
 EAX=00193000  EBX=00228000  ECX=004010AA  EDX=004010AA
 ESI=004010AA  EDI=004010AA  EBP=0019FF80  ESP=0019FF74
 EIP=0040367F  EFL=00000206  CF=0  SF=0  ZF=0  OF=0  AF=0  PF=1


 EAX=00193000  EBX=00228000  ECX=004010AA  EDX=004010AA
 ESI=004010AA  EDI=004010AA  EBP=0019FF80  ESP=0019FF74
 EIP=0040369C  EFL=00000206  CF=0  SF=0  ZF=0  OF=0  AF=0  PF=1


C:\Users\USER\source\repos\Lab1\Debug\Lab 1c.exe (process 7264) exited with code 0.
Press any key to close this window . . .
```

Detail Debugging Process

PART 2:

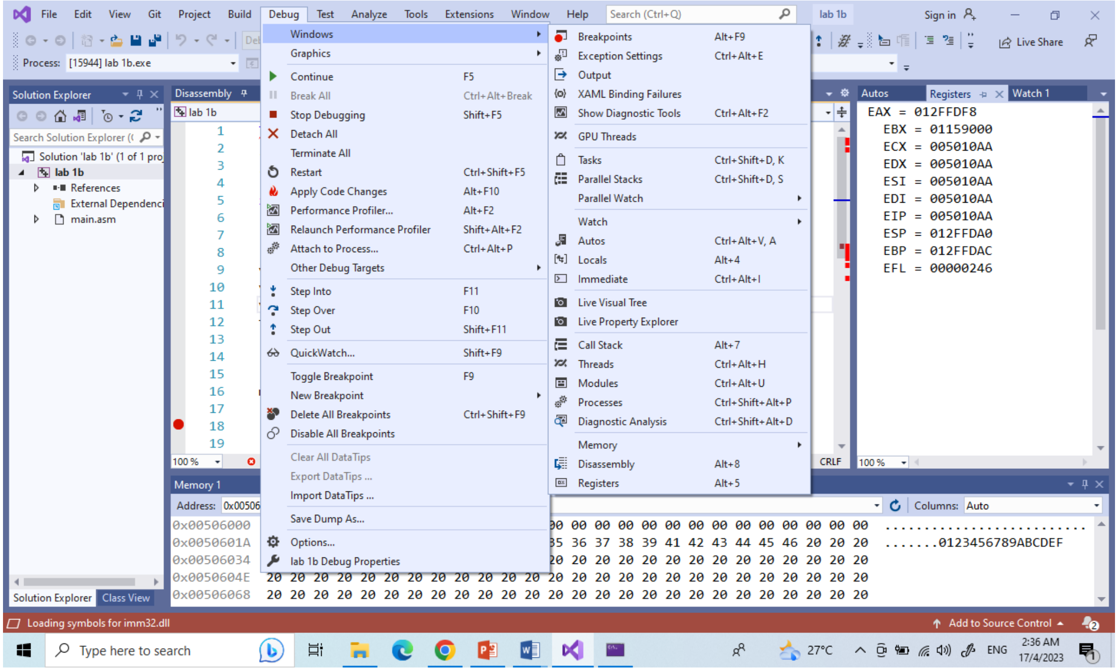
Run Debugging Process for Part B, Part C and Capture Video

1. Press F10 for step by step debugging.

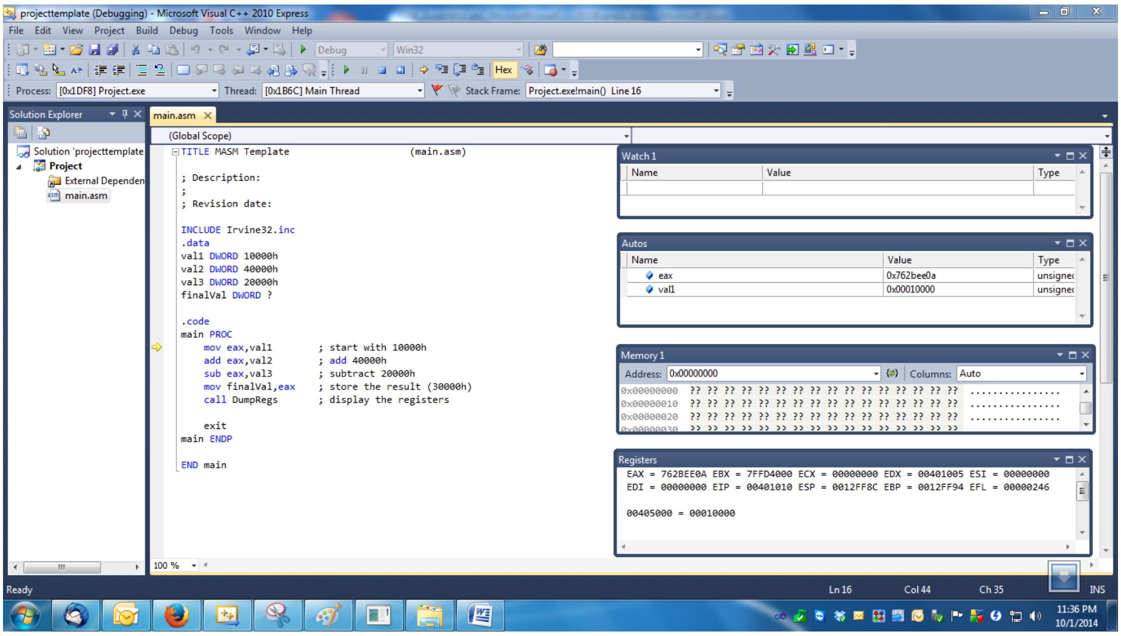


```
    add eax, val2    ; add 40000h
    sub eax, val3    ; subtract 20000h
    mov finalVal, eax    ; store the result(30000h)

    call DumpRegs; display the registers

    exit

main ENDP
```

3. Open 5 windows:
   a. Watch
   b. Autos
   c. Memory
   d. Registers
   e. Disassembly



```
EAX = 012FFDF8
EBX = 01159000
ECX = 005010AA
EDX = 005010AA
ESI = 005010AA
EDI = 005010AA
EIP = 005010AA
ESP = 012FFDA0
EBP = 012FFDAC
EFL = 00000246
```



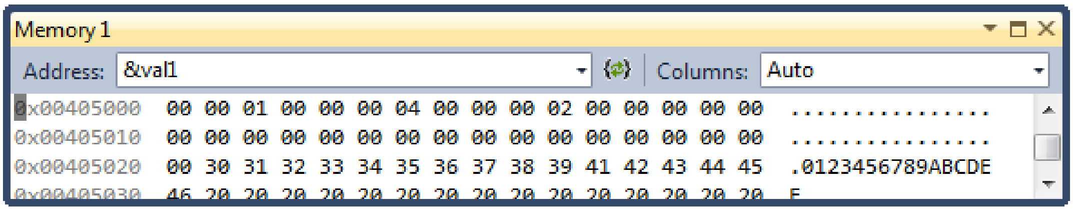
projecttemplate (Debugging) - Microsoft Visual C++ 2010 Express
Thread: [0x1B6C] Main Thread
Stack Frame: Project.exe!main() Line 16
Watch 1
Autos
Memory 1
Registers

```
TITLE MASM Template                     (main.asm)

; Description:
;
; Revision date:

INCLUDE Irvine32.inc
.data
val1 DWORD 10000h
val2 DWORD 40000h
val3 DWORD 20000h
finalVal DWORD ?

.code
main PROC
    mov eax,val1        ; start with 10000h
    add eax,val2        ; add 40000h
    sub eax,val3        ; subtract 20000h
    mov finalVal,eax    ; store the result (30000h)
    call DumpRegs       ; display the registers

    exit
main ENDP

END main
```

Name	Value	Type
eax	0x762bee0a	unsigned
val1	0x00010000	unsigned

```
EAX = 762BEE0A EBX = 7FFD4000 ECX = 00000000 EDX = 00401005 ESI = 00000000
EDI = 00000000 EIP = 00401010 ESP = 0012FF8C EBP = 0012FF94 EFL = 00000246

00405000 = 00010000
```

4. Default memory data segment value is at first variable: &val1 or 0x00405000 (depend on memory used).

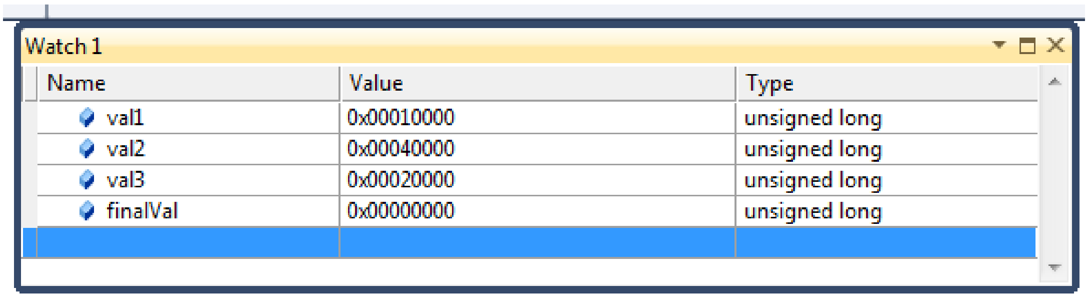


Memory 1

Address: &val1   Columns: Auto

```
0x00405000  00 00 01 00 00 00 04 00 00 00 02 00 00 00 00 00  ................
0x00405010  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0x00405020  00 30 31 32 33 34 35 36 37 38 39 41 42 43 44 45  .0123456789ABCDE
0x00405030  46 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20  F
```

5. Add variables val1, val2, val3 and finalval to Watch



Name	Value	Type
val1	0x00010000	unsigned long
val2	0x00040000	unsigned long
val3	0x00020000	unsigned long
finalVal	0x00000000	unsigned long

6. F10 to trace/step the assembly program line by line.

7. Please debug by looking at value changes at Watch, Memory, Autos and Registers windows.

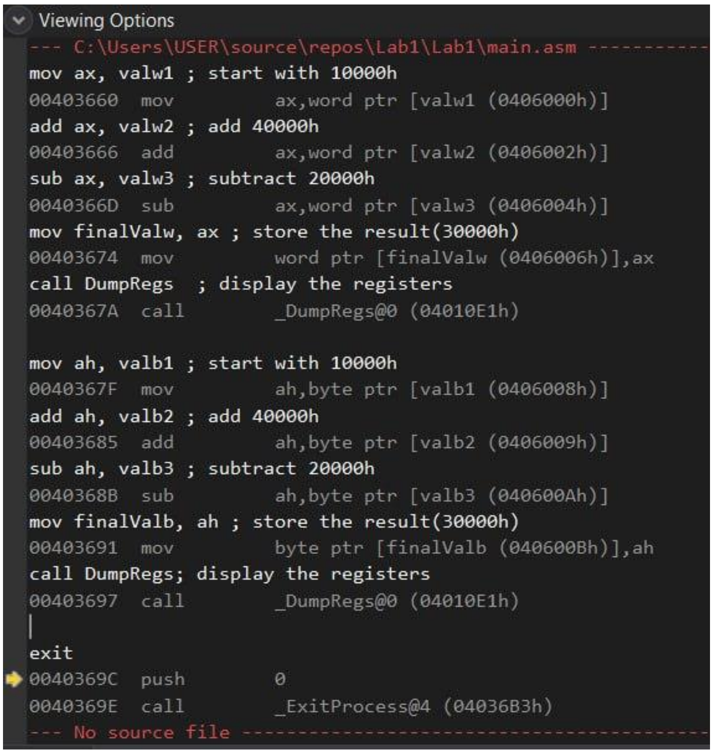
8. Disassembly window can show the following optional information:

- Memory address where each instruction is located. For native applications, it is the actual memory address. For Visual Basic or C#, it's an offset from the beginning of the function.
- Source code from which the assembly code derives.
- Code bytes, that is, the byte representations of the actual machine or MSIL instructions.
- Symbol names for the memory addresses.
- Line numbers corresponding to the source code.

What is Byte Code (or Machine Code) for the following assembly instructions:

```
    mov ax,valw1  ; start with 10000h
    add ax,valw2        ; add 40000h
    sub ax,valw3        ; subtract 20000h
    mov finalValw,ax    ; store the result (30000h)
```

Byte Code:



```
--- C:\Users\USER\source\repos\Lab1\Lab1\main.asm -----------
mov ax, valw1 ; start with 10000h
00403660  mov         ax,word ptr [valw1 (0406000h)]
add ax, valw2 ; add 40000h
00403666  add         ax,word ptr [valw2 (0406002h)]
sub ax, valw3 ; subtract 20000h
0040366D  sub         ax,word ptr [valw3 (0406004h)]
mov finalValw, ax ; store the result(30000h)
00403674  mov         word ptr [finalValw (0406006h)],ax
call DumpRegs  ; display the registers
0040367A  call        _DumpRegs@0 (04010E1h)

mov ah, valb1 ; start with 10000h
0040367F  mov         ah,byte ptr [valb1 (0406008h)]
add ah, valb2 ; add 40000h
00403685  add         ah,byte ptr [valb2 (0406009h)]
sub ah, valb3 ; subtract 20000h
0040368B  sub         ah,byte ptr [valb3 (040600Ah)]
mov finalValb, ah ; store the result(30000h)
00403691  mov         byte ptr [finalValb (040600Bh)],ah
call DumpRegs; display the registers
00403697  call        _DumpRegs@0 (04010E1h)

exit
0040369C  push        0
0040369E  call        _ExitProcess@4 (04036B3h)
--- No source file ------------------------------------------
```

Experimental Results:

a) Rebuild & Start Debugging CamStudio:

https://youtu.be/n0btdlX7Qwk?si=GOtfRTq0VN6_l4Gy

Completed Fully Partially : Checked by: ______________