

CSC 3210

Computer Organization and Programming

Lab Work 4

Dr. Zulkar Nine

mnine@gsu.edu

Georgia State University

Fall 2021

Learning Objective

- Assembly Language Template
- Assembler Directives
- Writing and Debugging a program
- Watch registers and flag values

Lab Work 4 Instructions

- Lab 4: Write and run a program that can evaluate an expression (10 points)
- More details on Slide – 44 and 45

Due Date: Posted on iCollege

Disclaimer

- The process shown in these slides might not work in every single computer due to Operating system version, Microsoft Visual Studio versions and everything.
- If you find any unusual error, you can inform the instructor.
- Instructor will help you resolve the issue.

Attendance!

A review on assembly language

Program Template

in Microsoft Visual Studio

Program Template

Program title, optional

`; Program template (Template.asm)`

32-bit program directives

```
.386
.model flat,stdcall
.stack 4096
ExitProcess PROTO, dwExitCode:DWORD
```

Data section, not always needed

```
.data
    ; declare variables here
```

Code section, always needed

```
.code
main PROC
    ; write your code here

    INVOKE ExitProcess,0
main ENDP
END main
```

Assembler Directives

; AddTwo.asm - adds two 32-bit integers

.386

.model flat, stdcall

.stack 4096

ExitProcess PROTO, dwExitCode:DWORD

.code

main PROC

mov eax, 5 ; move 5 to the EAX register

add eax, 6 ; add 6 to the EAX register

INVOKE ExitProcess, 0

main ENDP

END main

.386 directive, identifies this as a 32-bit program that can access 32-bit registers and addresses.

.model selects the program's memory **model (flat)**, and identifies the **calling convention (named stdcall)** for procedures. Ex. Windows API

.stack aside **4096** bytes of storage for the **runtime stack**, which every program must have. Size of a memory page.

Assembler Directives

; AddTwo.asm - adds two 32-bit integers

.386

.model flat,stdcall

.stack 4096

~~ExitProcess~~ ~~PROTO, dwExitCode:DWORD~~

.code

main PROC

mov eax,5 ; move 5 to the EAX register

add eax,6 ; add 6 to the EAX register

INVOKE ExitProcess,0

main ENDP

END main

ExitProcess, declares a **prototype** for the ExitProcess function.

- A prototype consists of the function name, the **PROTO** keyword, a **comma**, and a **list of input parameters**.

- The input parameter for ExitProcess is named dwExitCode.

.CODE directive marks the beginning of the code area of a program, the area that contains executable instructions

The label main identifies **the program entry point** (main) and marks the address at which the program will begin to execute.

Assembler Directives

; AddTwo.asm - adds two 32-bit integers

.386

.model flat,stdcall

.stack 4096

ExitProcess PROTO, dwExitCode:DWORD

.code

main PROC

mov eax,5 ; move 5 to the EAX register

add eax,6 ; add 6 to the EAX register

INVOKE ExitProcess,0

main ENDP

END main

INVOKE Calls on, the procedure ExitProcess, passing the arguments on the stack or in registers.

ENDP directive marks the end of a procedure. Our program had a procedure named main,

end directive marks the last line to be assembled, and it identifies the program entry point (main).

Writing and Debugging

A program

Writing & Debugging a Program

Exercise 1: Write and run a program to solve the following problem:

$$\mathbf{EAX} = (\mathbf{ECX} + \mathbf{EBX}) - \mathbf{EDX}$$

- Assume that all the values are 32 bit and all the registers are 32 bit as well
- Store the following **decimal values** into the registers -
 - 15 to ECX, 15 to EBX and 31 to EDX
 - Evaluate the above expression
 - Store the result in EAX register

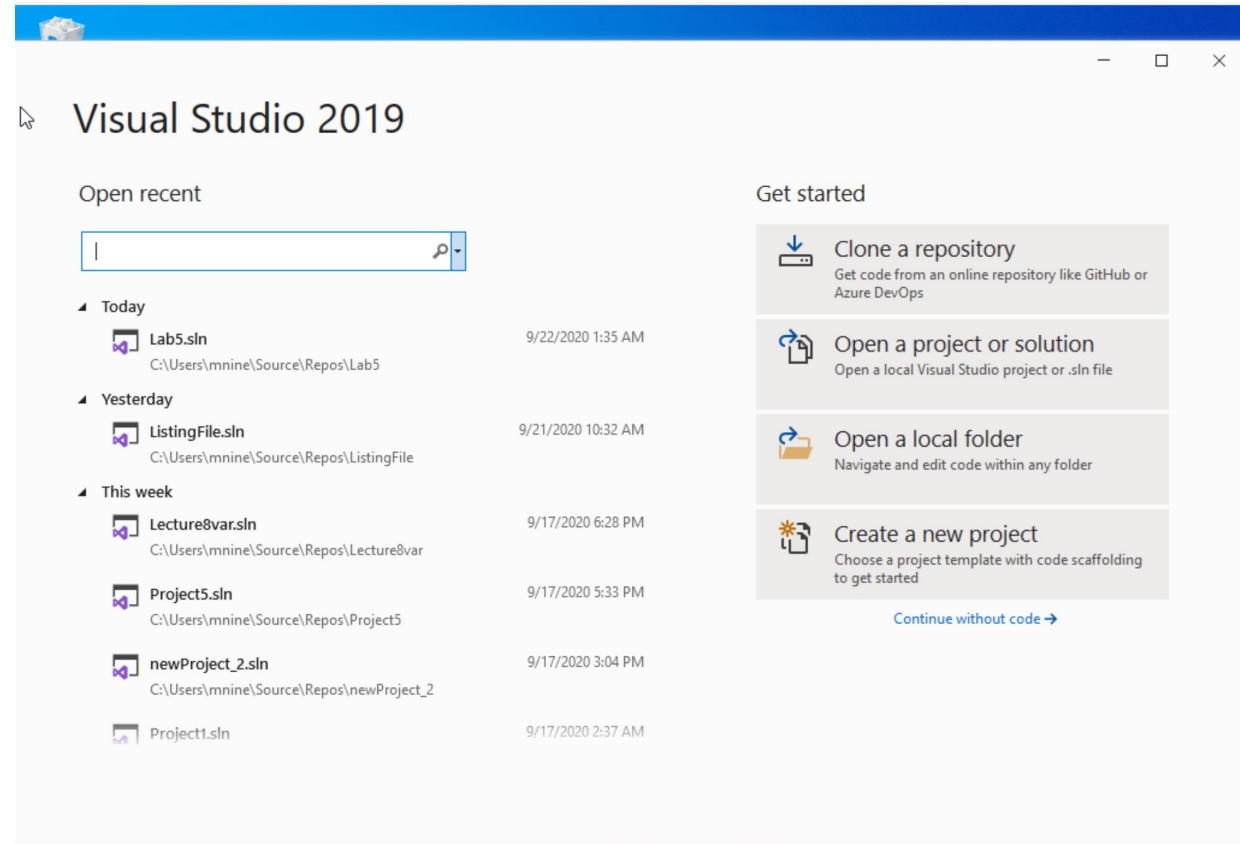
- To create a project for this exercise, follow the steps provided in next slides (similar to lab 2)
- To verify the result in EAX, follow the steps in the slides titled : “Showing **registers** and **flags**”

Steps to follow

- Follow the steps :
 - **Step 1:** Create a project
 - **Step 2:** Write your code to evaluate, $EAX = (ECX + EBX) - EDX$
 - **Step 3:** Build the project
 - **Step 4:** Debug the project

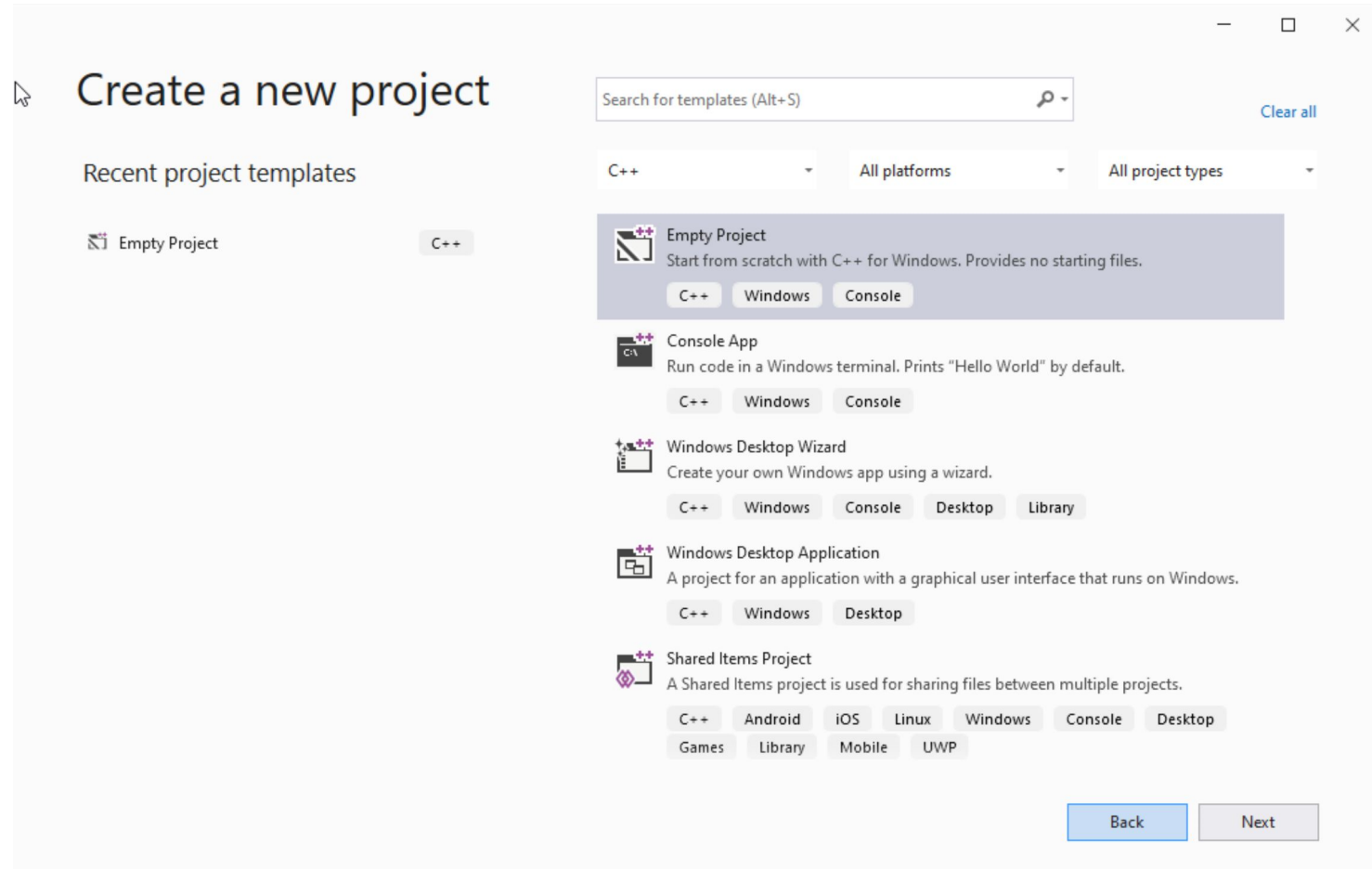
Step 1: Create a project (1)

- (1) Start Visual Studio
- (2) Click Create a new Project



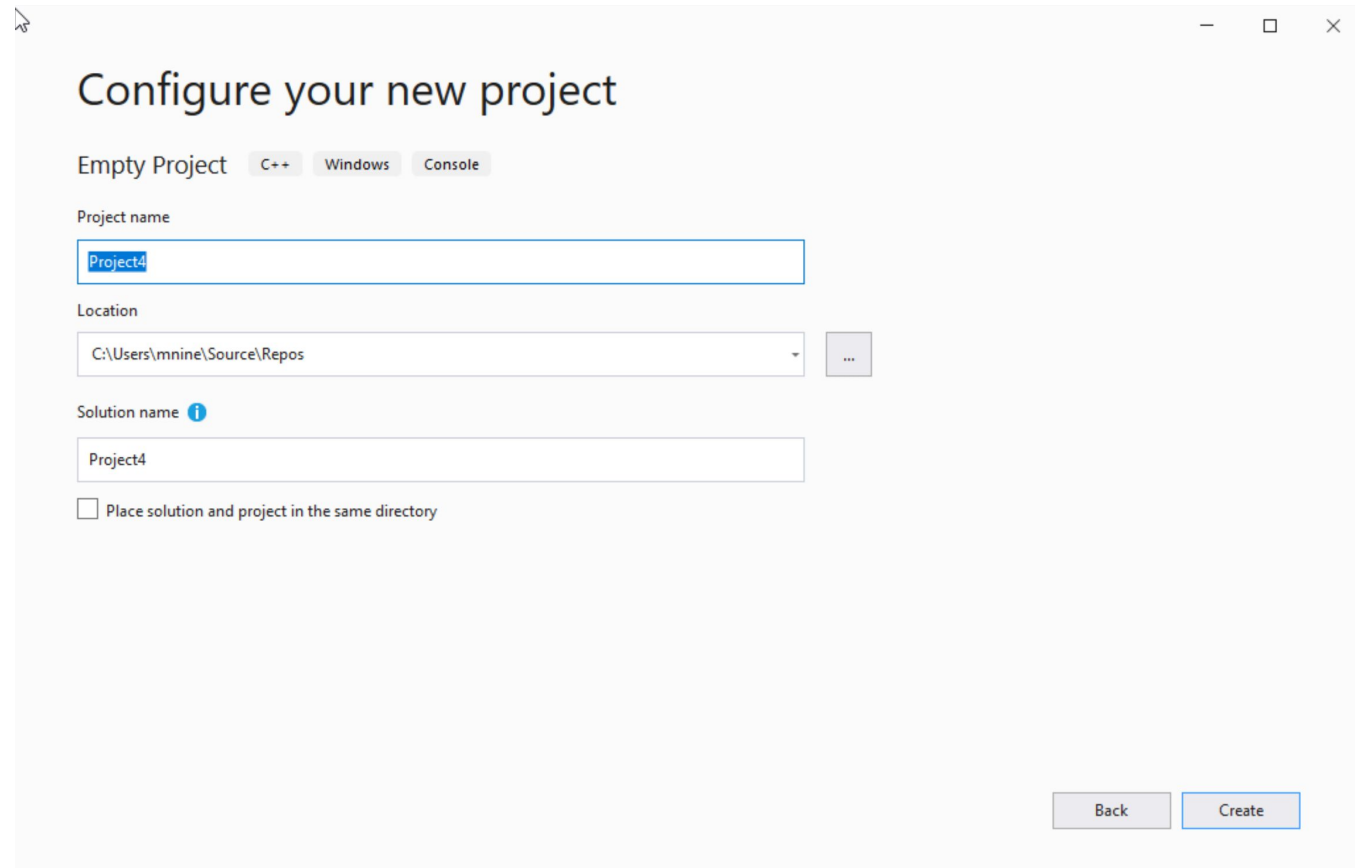
Step 1: Create a project (2)

- (1) Select C++ as language
- (2) Select Empty Project
- (3) Click Next



Step 1: Create a project (3)

- (1) You can change the project name as you like
- (1) Also you can change the project location
- (2) Click Next



The screenshot shows the 'Configure your new project' dialog box in Visual Studio. The title bar includes standard window controls. The main heading is 'Configure your new project'. Below it, there are tabs for 'Empty Project', 'C++', 'Windows', and 'Console'. The 'Empty Project' tab is selected. The 'Project name' field contains 'Project4'. The 'Location' field shows the path 'C:\Users\mnine\Source\Repos' with a browse button ('...') to its right. The 'Solution name' field, which has an information icon ('i') to its left, also contains 'Project4'. At the bottom, there is a checkbox labeled 'Place solution and project in the same directory' which is currently unchecked. In the bottom right corner, there are 'Back' and 'Create' buttons.

Step 1: Create a project (4)

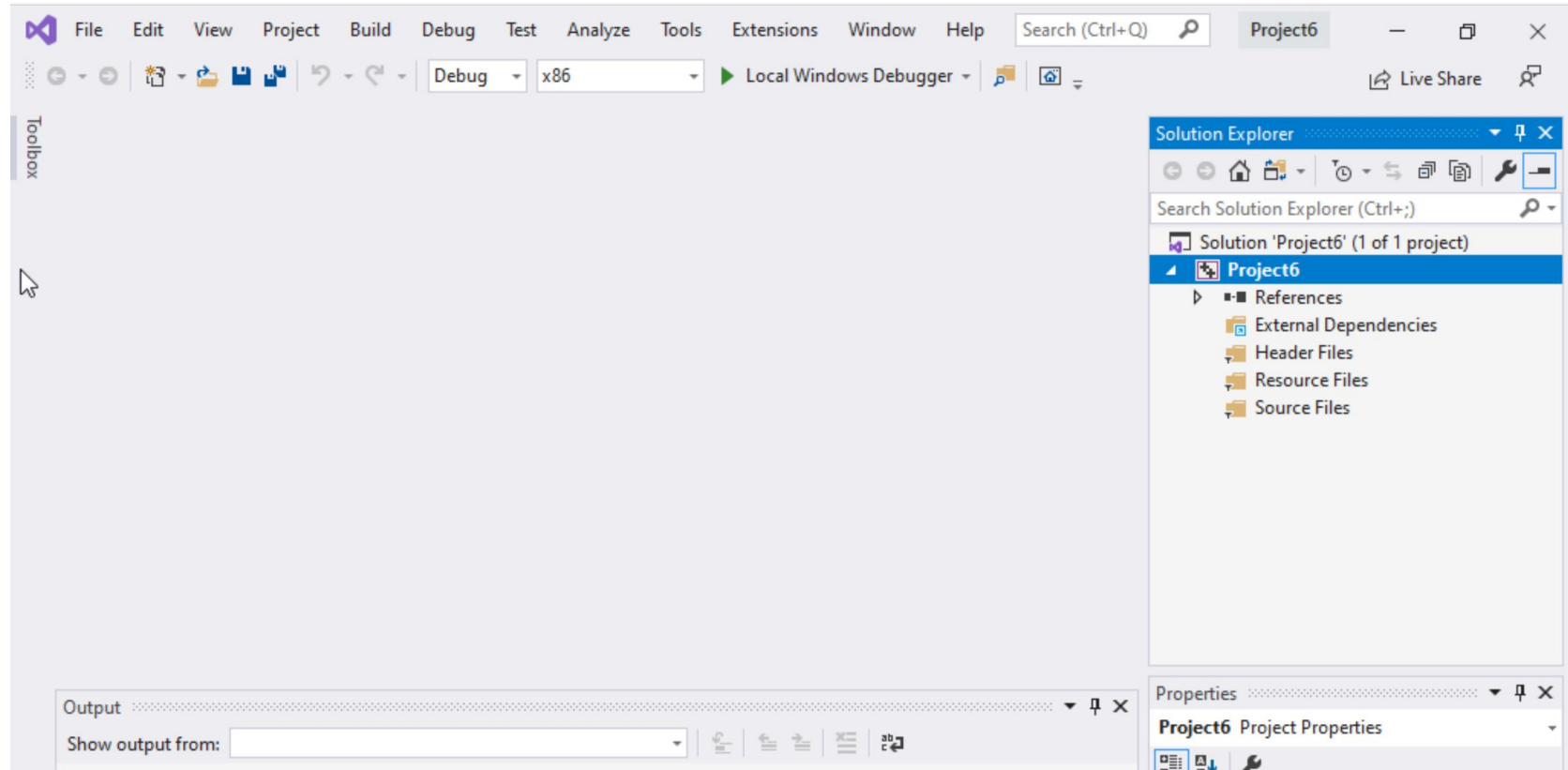
Delete the

Following folders:

- Header files

- Resources Files, and

- Source Files



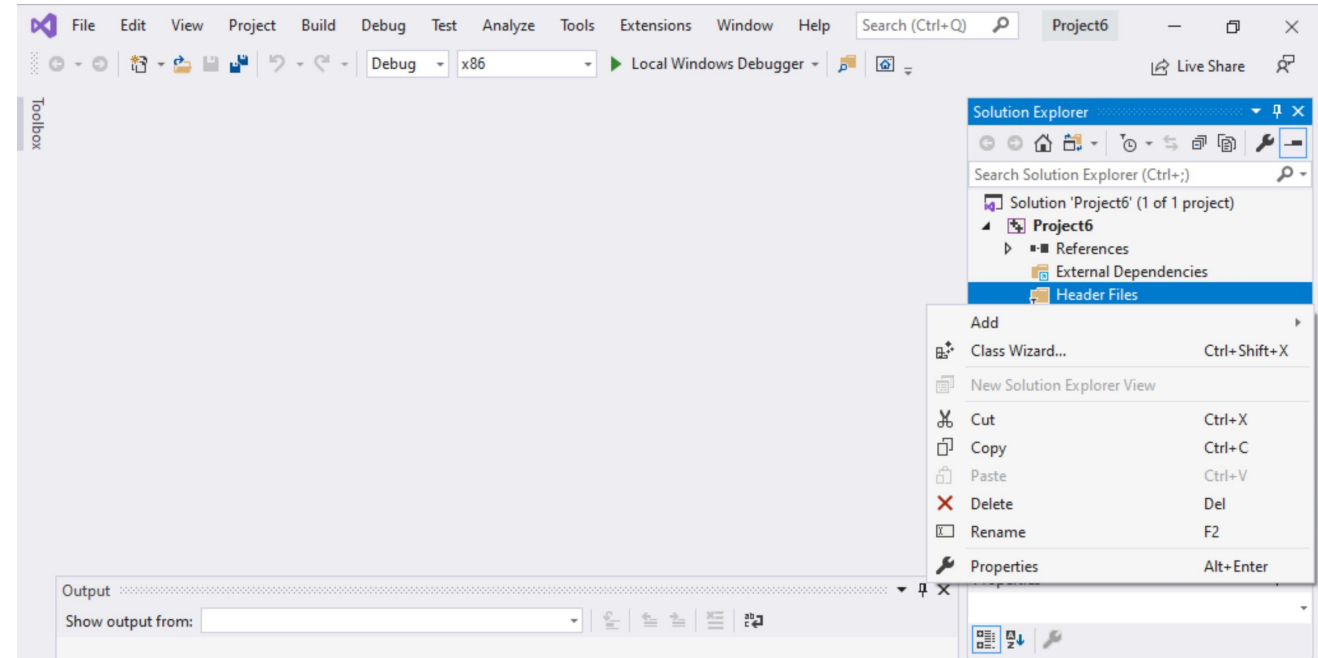
Step 1: Create a project (5)

To delete :

Select the folders

Right click on it

Select delete



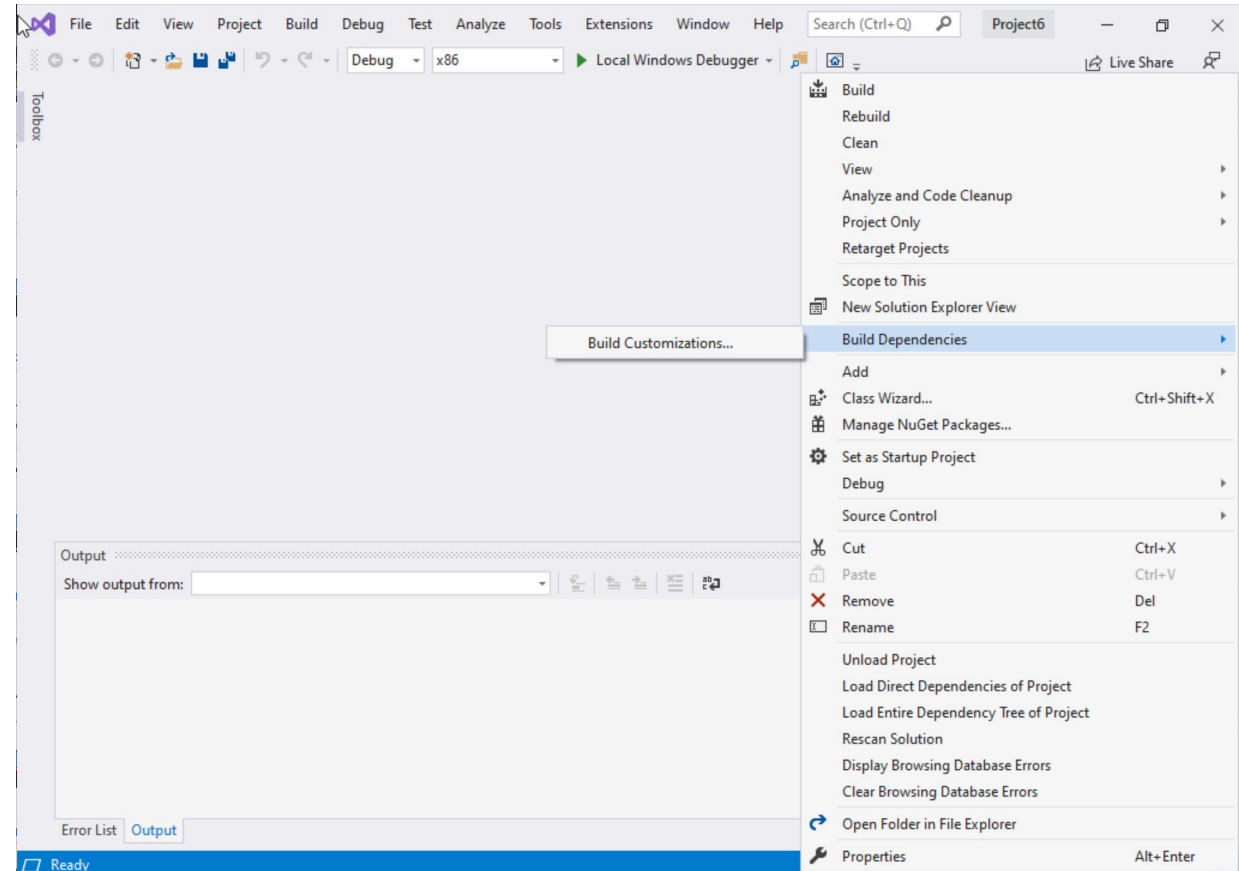
Step 1: Create a project (6)

Select Project Name on solution explorer

Right click on it

Go to Build Dependencies

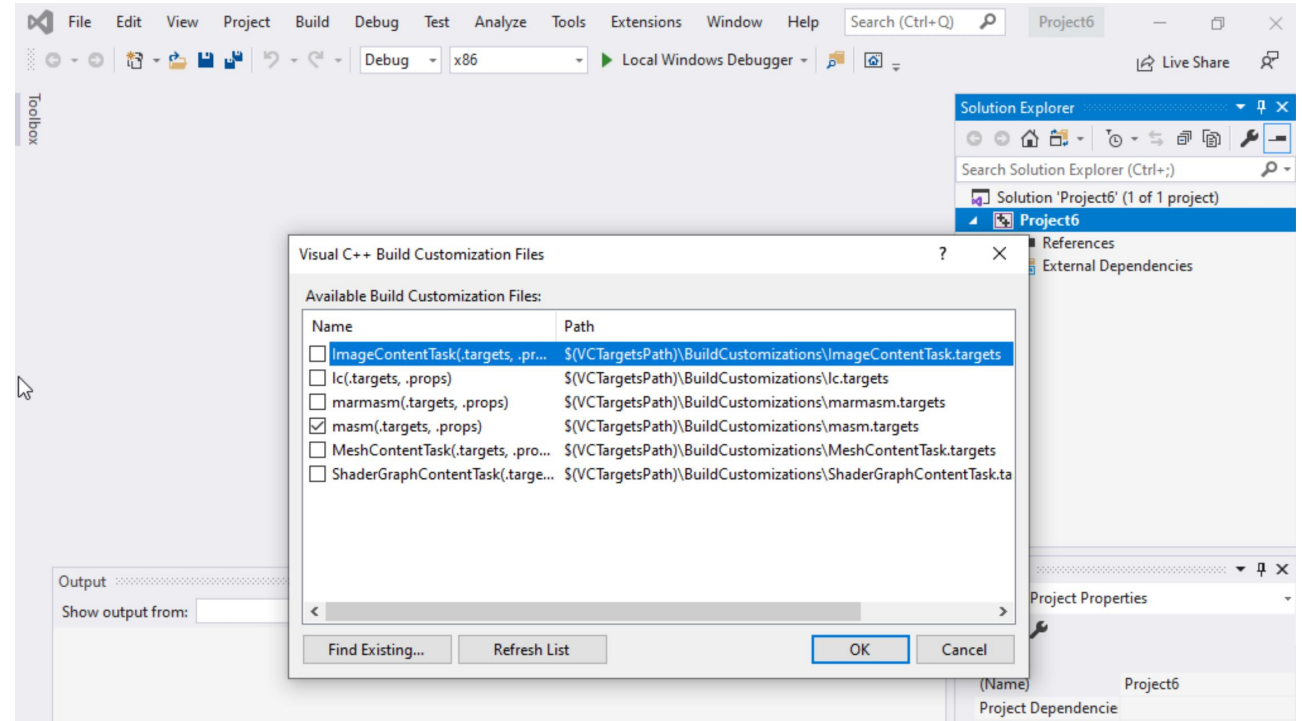
Click on Build Customizations



Step 1: Create a project (7)

Select mash(.target, .props)

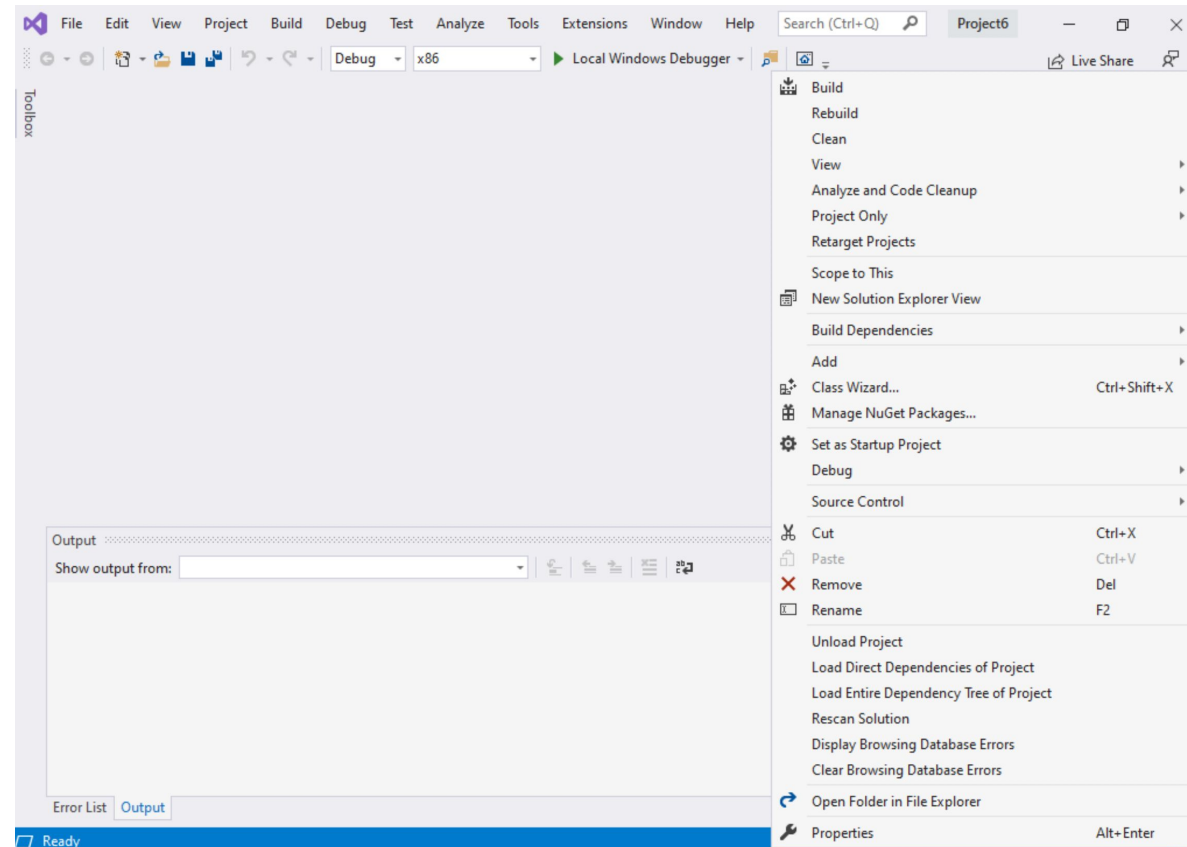
Click ok



Step 1: Create a project (8)

Right click on the Project name in the solution explorer

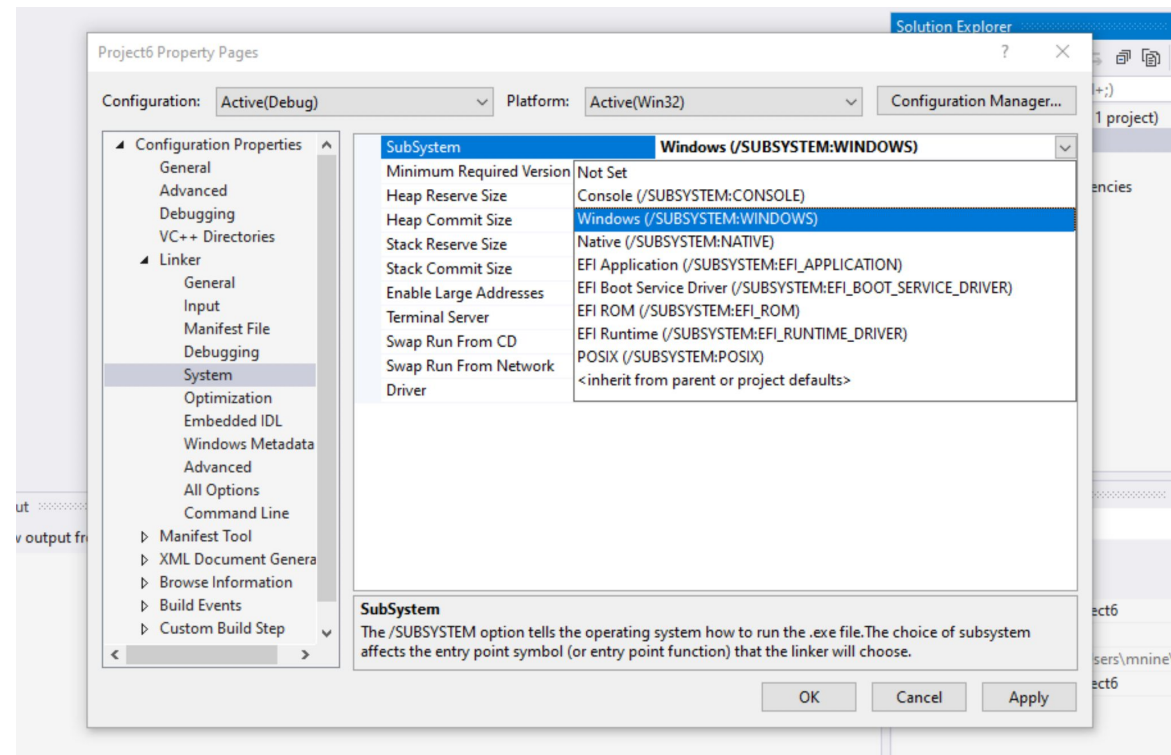
Click properties



Step 1: Create a project (9)

Select Windows(/SUBSYSTEM:WINDOWS)

Click OK



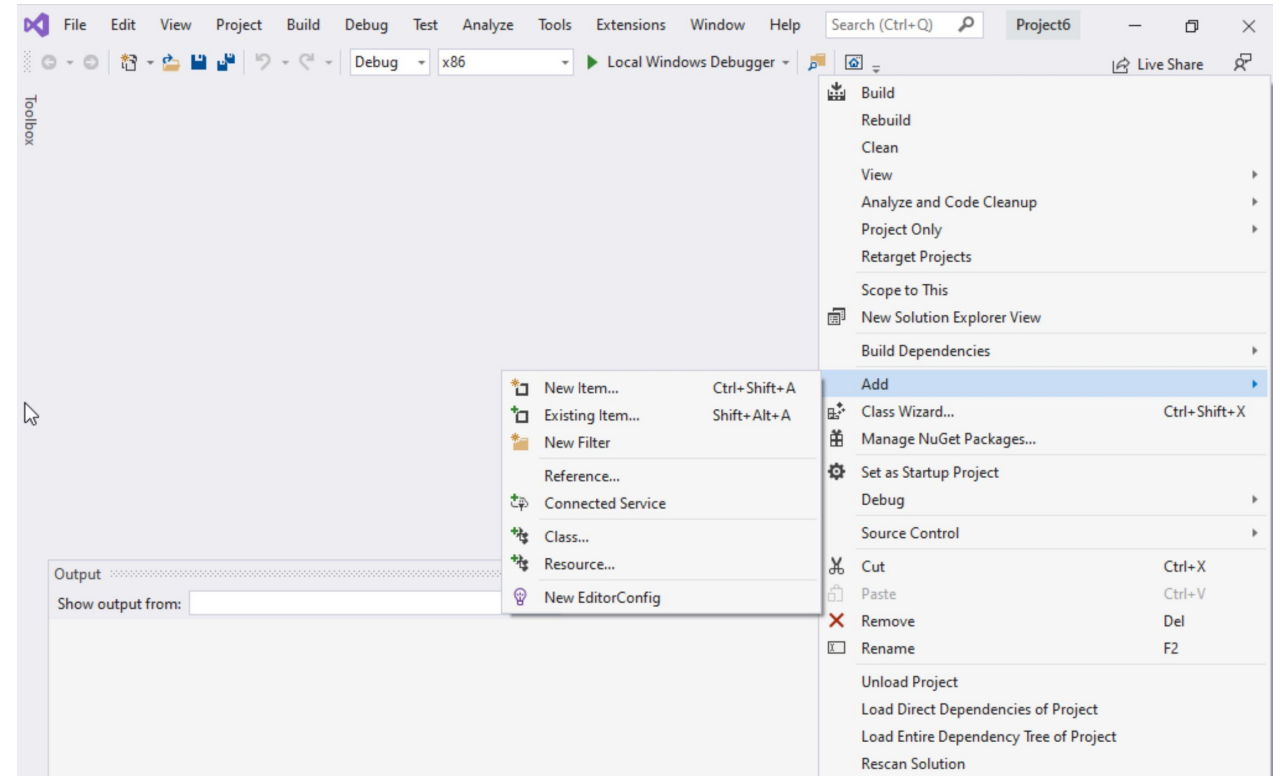
Step 1: Create a project (10)

Select Project name on solution explorer

Right click on it

Expand Add

Choose New Item

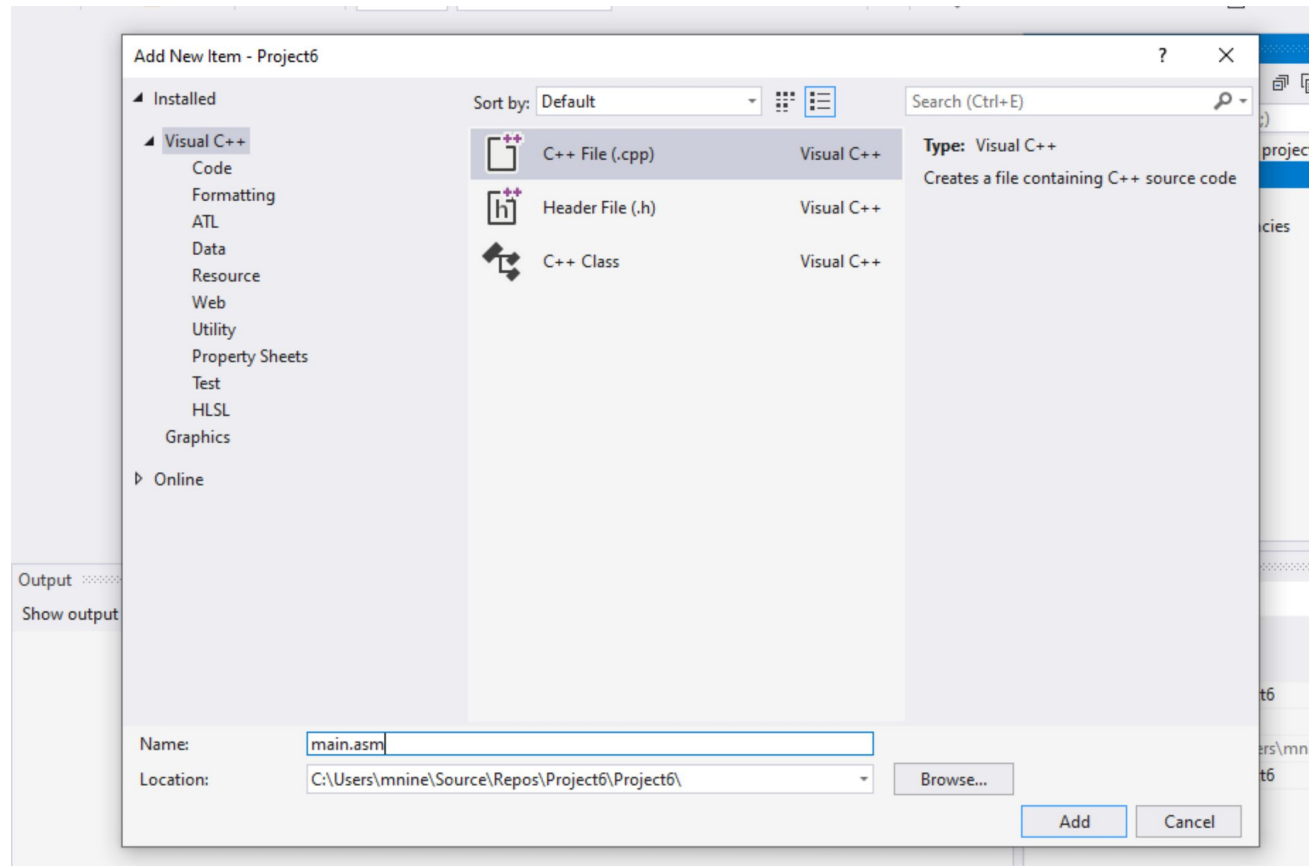


Step 1: Create a project (11)

Select C++ File(.cpp)

Name: main.asm

Click Add

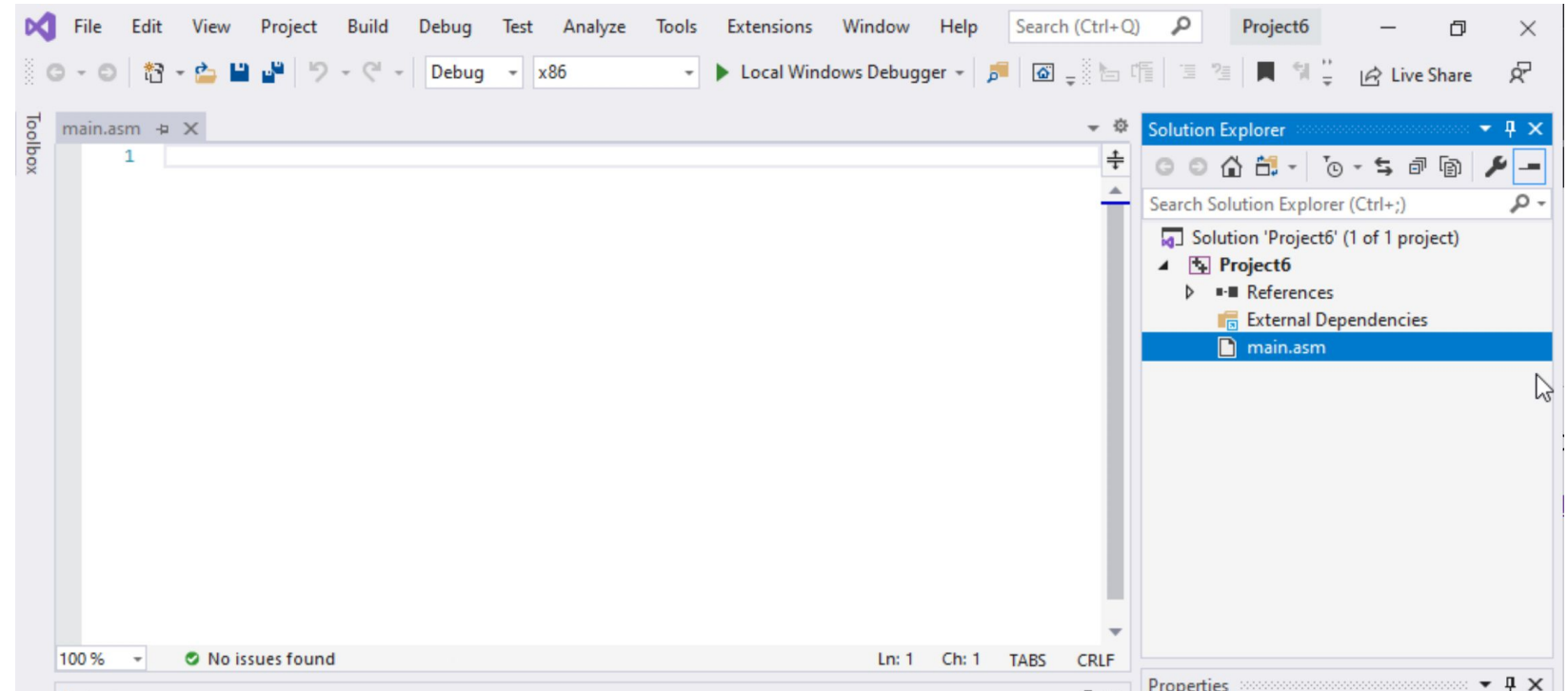


Step 1: Create a project (12)

Select main.asm

Add your code

In the main.asm File.



Step 2: Write your code to evaluate, $EAX = (ECX + EBX) - EDX$

4. Make the required changes:

$$EAX = (ECX + EBX) - EDX$$

- Move to ECX the value 15
- Move to EBX the value 15
- Move to EDX the value 31
- Add EBX to ECX
- Subtract EDX from ECX (😬)
- Move the result in ECX to EAX
- Done

Replace this

```
; AddTwo.asm - adds two 32-bit integers.
; Chapter 3 example

.386
.model flat,stdcall
.stack 4096
ExitProcess proto,dwExitCode:dword

.code
main proc
    mov eax,5
    add eax,6
    invoke ExitProcess,0
main endp
end main
```

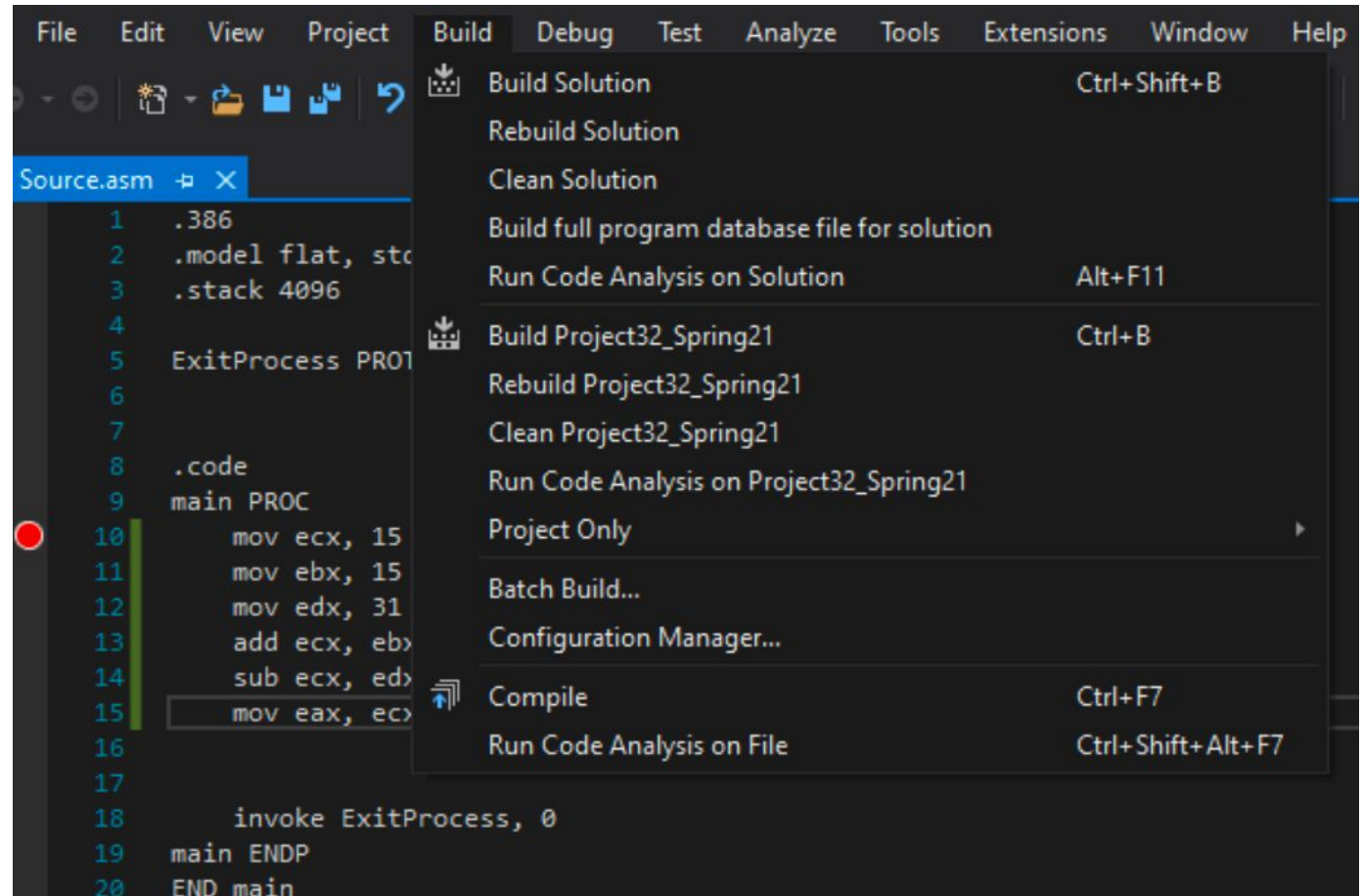
Step 2: The code

```
; AddTwo.asm - adds two 32-bit integers.
; Chapter 3 example
.386
.model flat,stdcall
.stack 4096
ExitProcess proto,dwExitCode:dword

.code
main proc
    mov ecx,15
    mov ebx,15
    mov edx,31
    add ecx,ebx
    sub ecx,edx
    mov eax,ecx
    invoke ExitProcess,0
main endp
end main
```

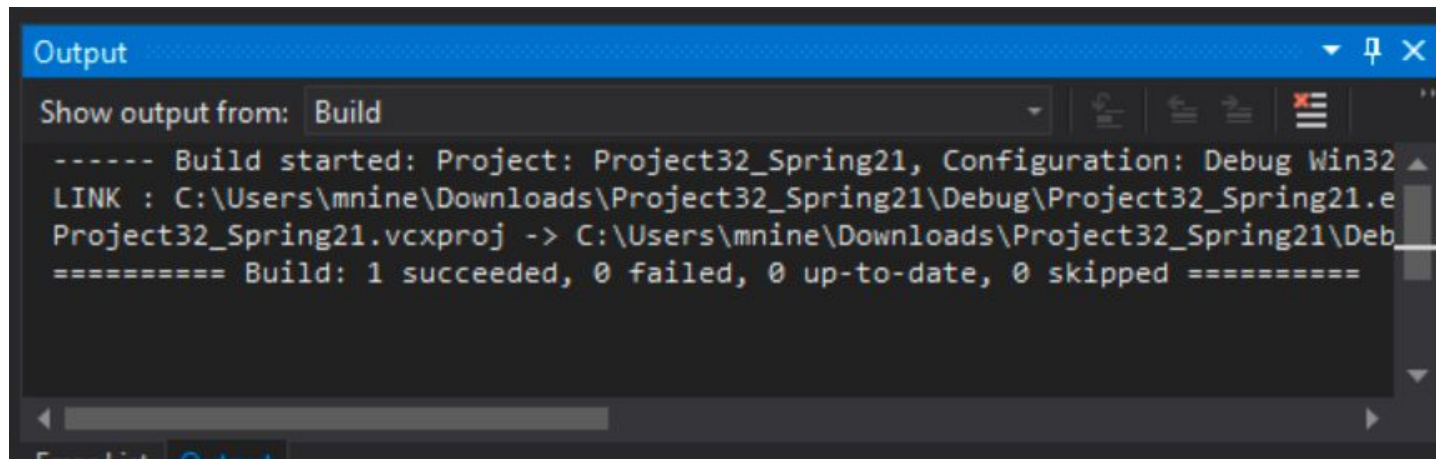
Step 3: Build the Project (1)

- Select **Build Project** from the **Build** menu.
- This will **assemble** and **link** your program and create an executable file.



Step 3: Build the Project (2)

- You should see messages like the following in your output window, indicating the build progress:

A screenshot of the Visual Studio Output window. The title bar is blue and says "Output". Below it, there's a dropdown menu set to "Build" and some icons. The main area is a dark gray text box with white text. The text shows the build process for "Project32_Spring21" in "Debug Win32" configuration. It includes the linker command and the final status message: "Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped".

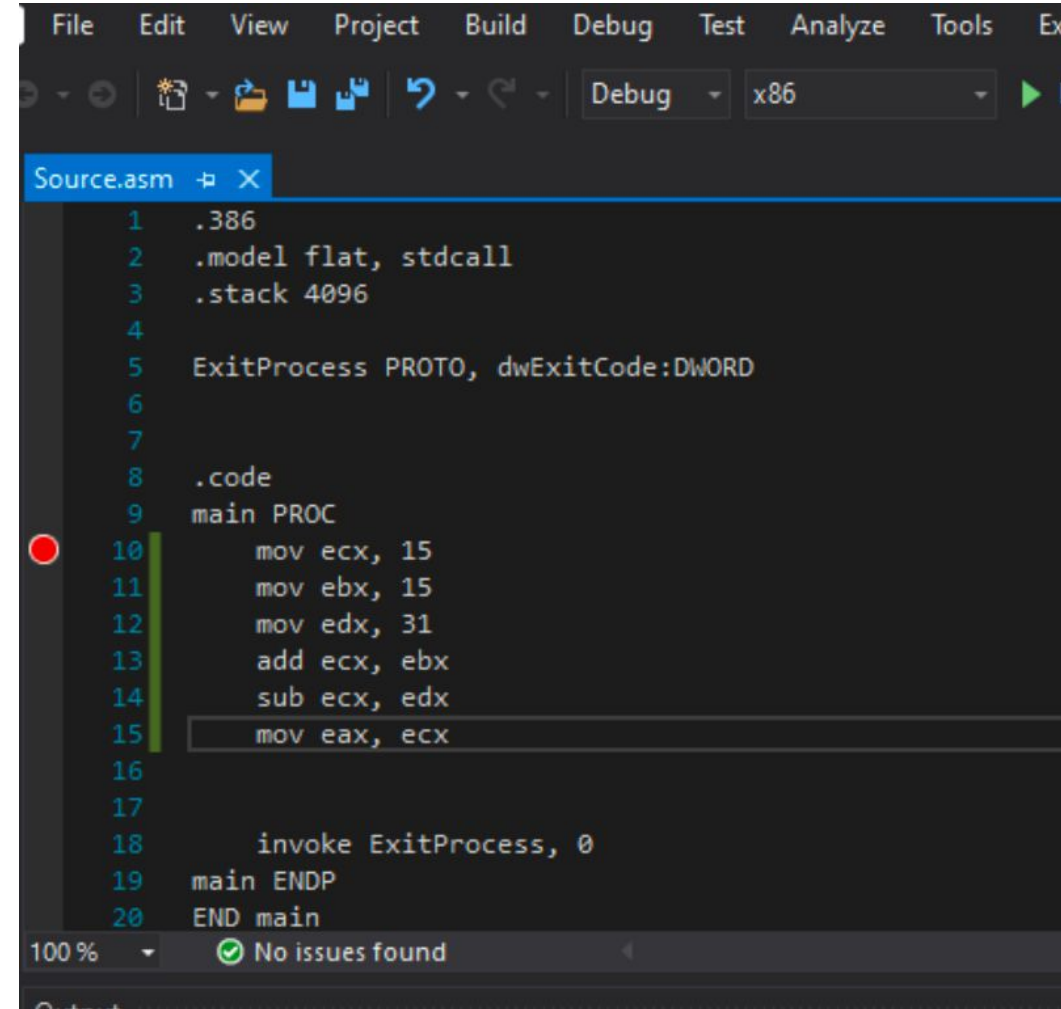
```
----- Build started: Project: Project32_Spring21, Configuration: Debug Win32
LINK : C:\Users\mnine\Downloads\Project32_Spring21\Debug\Project32_Spring21.e
Project32_Spring21.vcxproj -> C:\Users\mnine\Downloads\Project32_Spring21\Deb
===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped =====
```

- You should see the message in the last line:

=====Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped =====

Step 4: Debug the project (1)

- Set a breakpoint first.
- **Set a breakpoint** on a program statement by clicking the mouse in the vertical gray bar just to the left of the code window.
- A **large red dot** will mark the breakpoint location.
- In this case, set a breakpoint at **Line 10**.



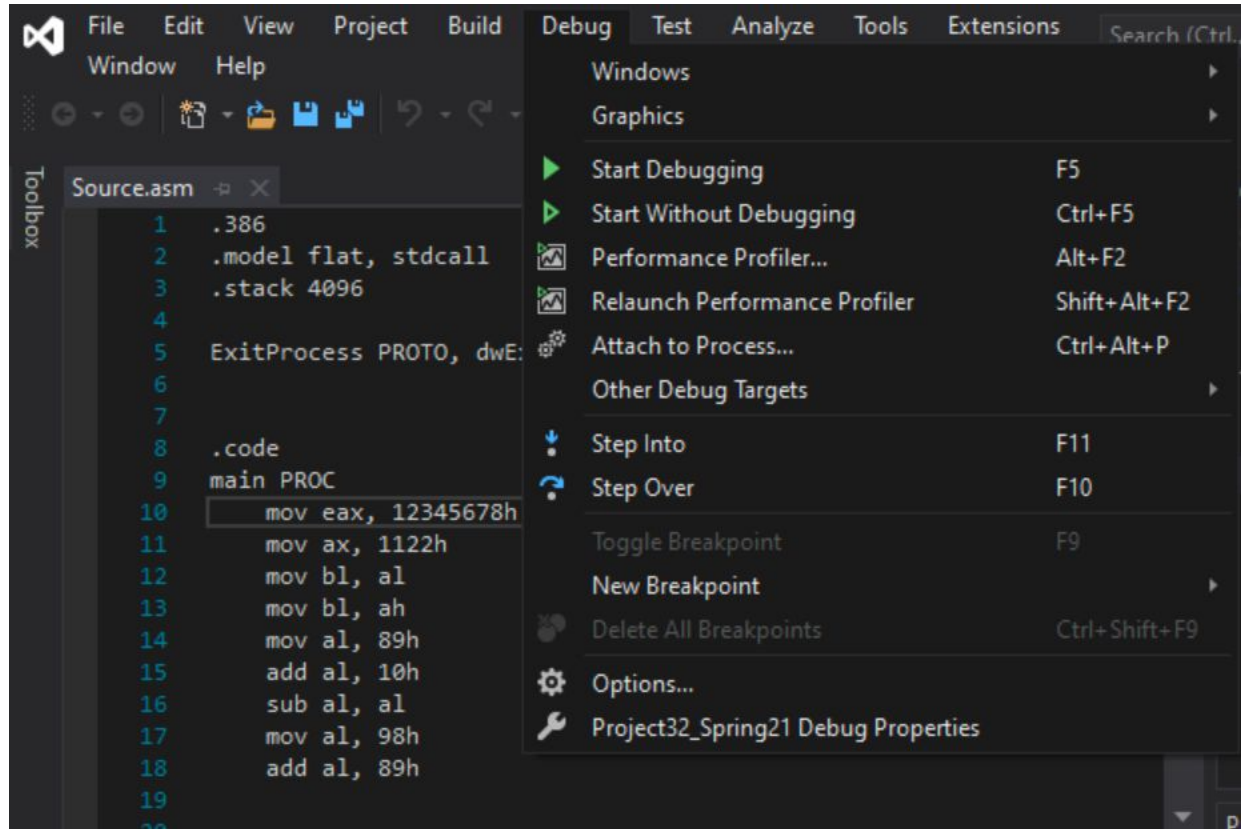
The screenshot shows a debugger interface with a menu bar (File, Edit, View, Project, Build, Debug, Test, Analyze, Tools, Ex) and a toolbar. The 'Source.asm' window is active, displaying assembly code. A red dot, representing a breakpoint, is placed in the vertical gray bar to the left of line 10. The code is as follows:

```
1 .386
2 .model flat, stdcall
3 .stack 4096
4
5 ExitProcess PROTO, dwExitCode:DWORD
6
7
8 .code
9 main PROC
10     mov ecx, 15
11     mov ebx, 15
12     mov edx, 31
13     add ecx, ebx
14     sub ecx, edx
15     mov eax, ecx
16
17
18     invoke ExitProcess, 0
19 main ENDP
20 END main
```

The status bar at the bottom indicates '100 %' zoom and 'No issues found'.

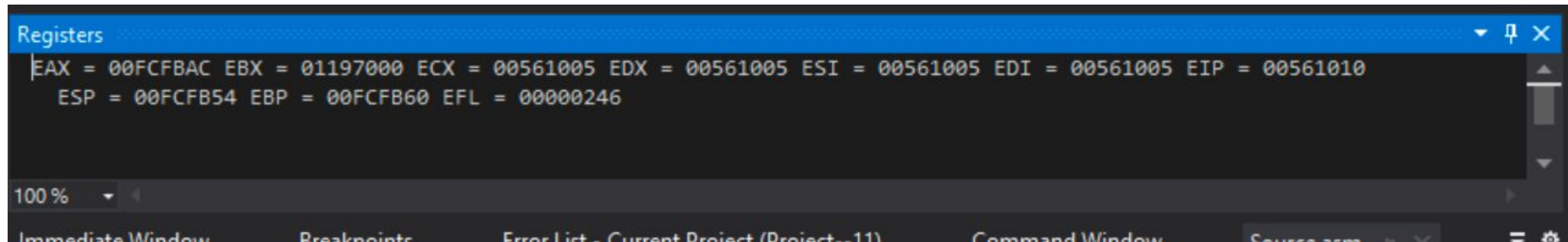
Step 4: Debug the project (2)

- **Run** the Program by selecting **Start Debugging** from the **Debug** menu.



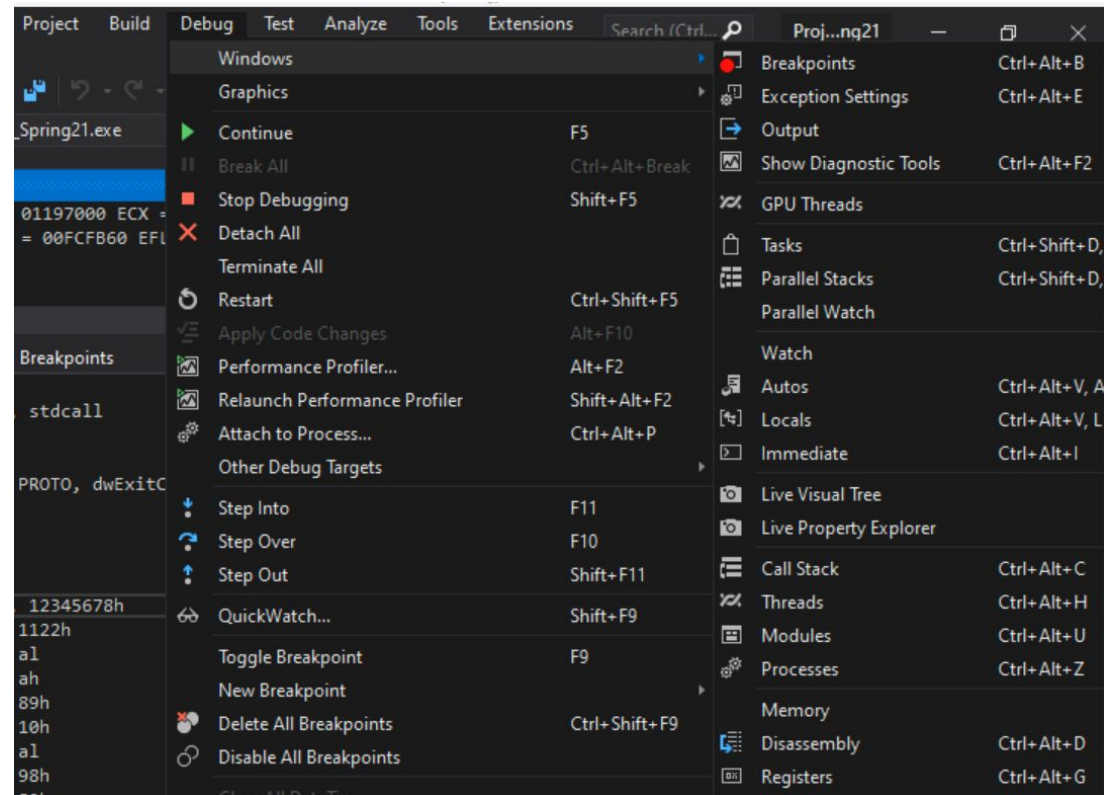
Step 4: Debug the project (3)

- You should be able to see the register window :



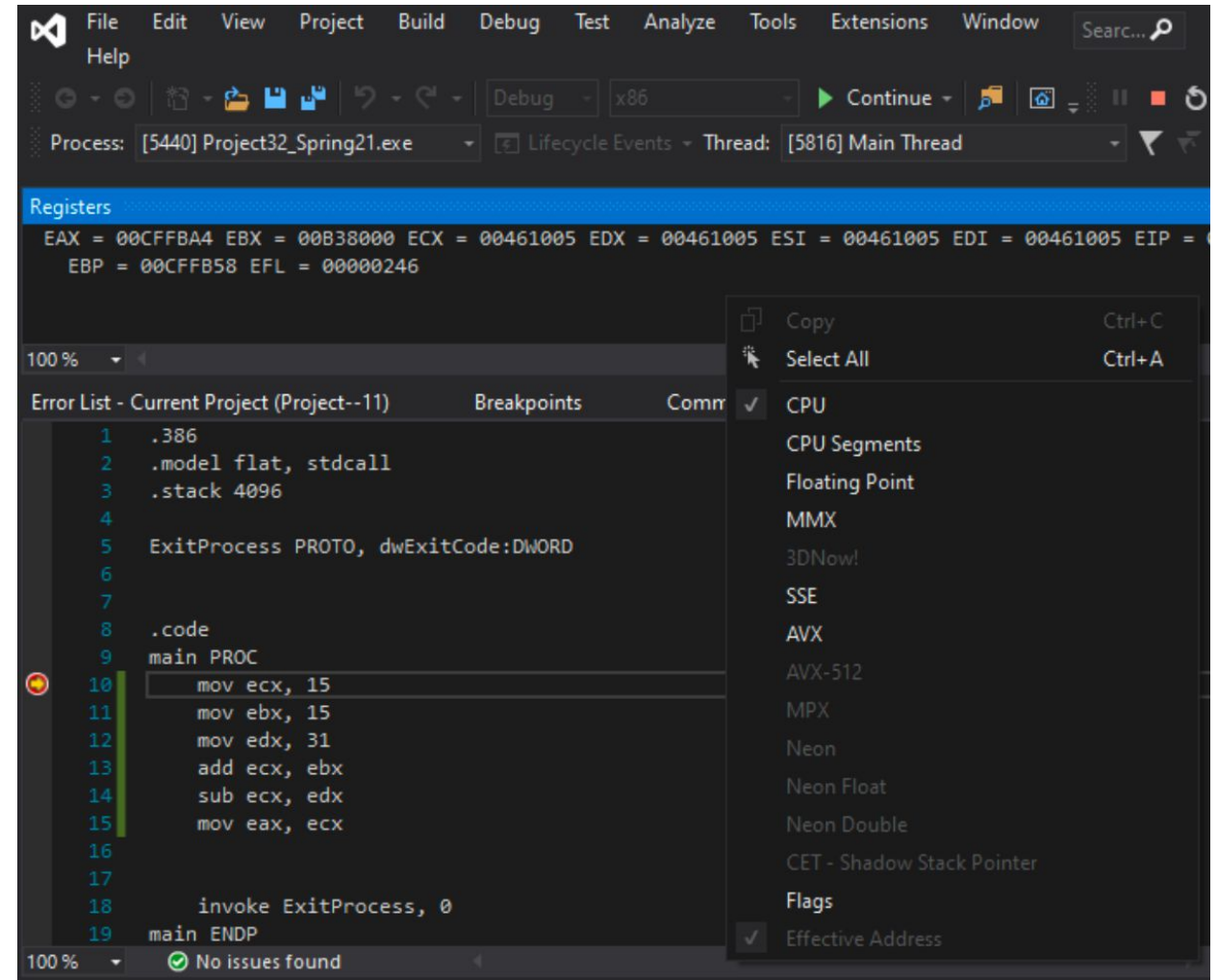
Step 4: Debug the project (4)

- If you don't see the register window: Go to : Debug -> Windows->registers.



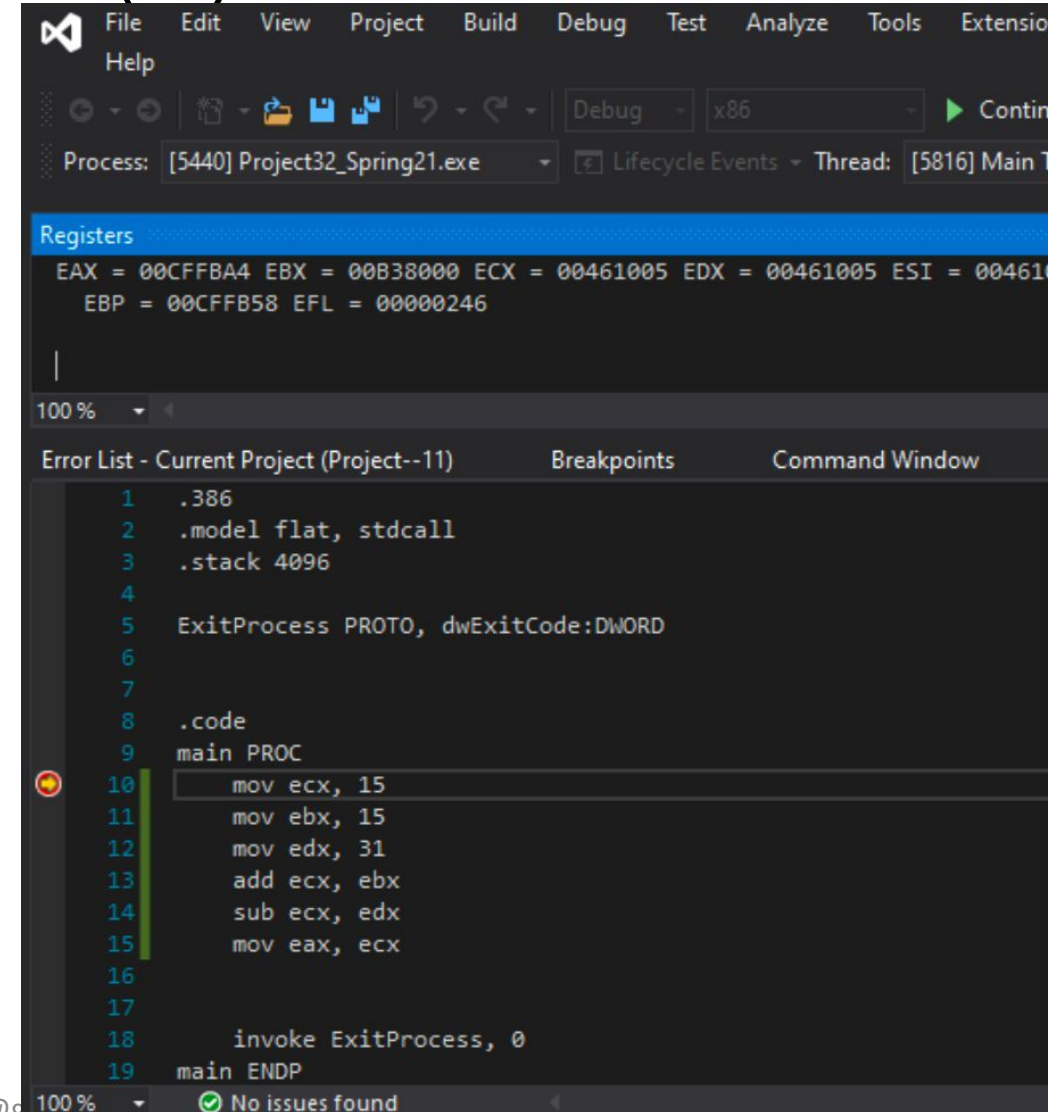
Step 4: Debug the project (5)

- Right click on the register window
- Turn on the EFLAGS in the register window.



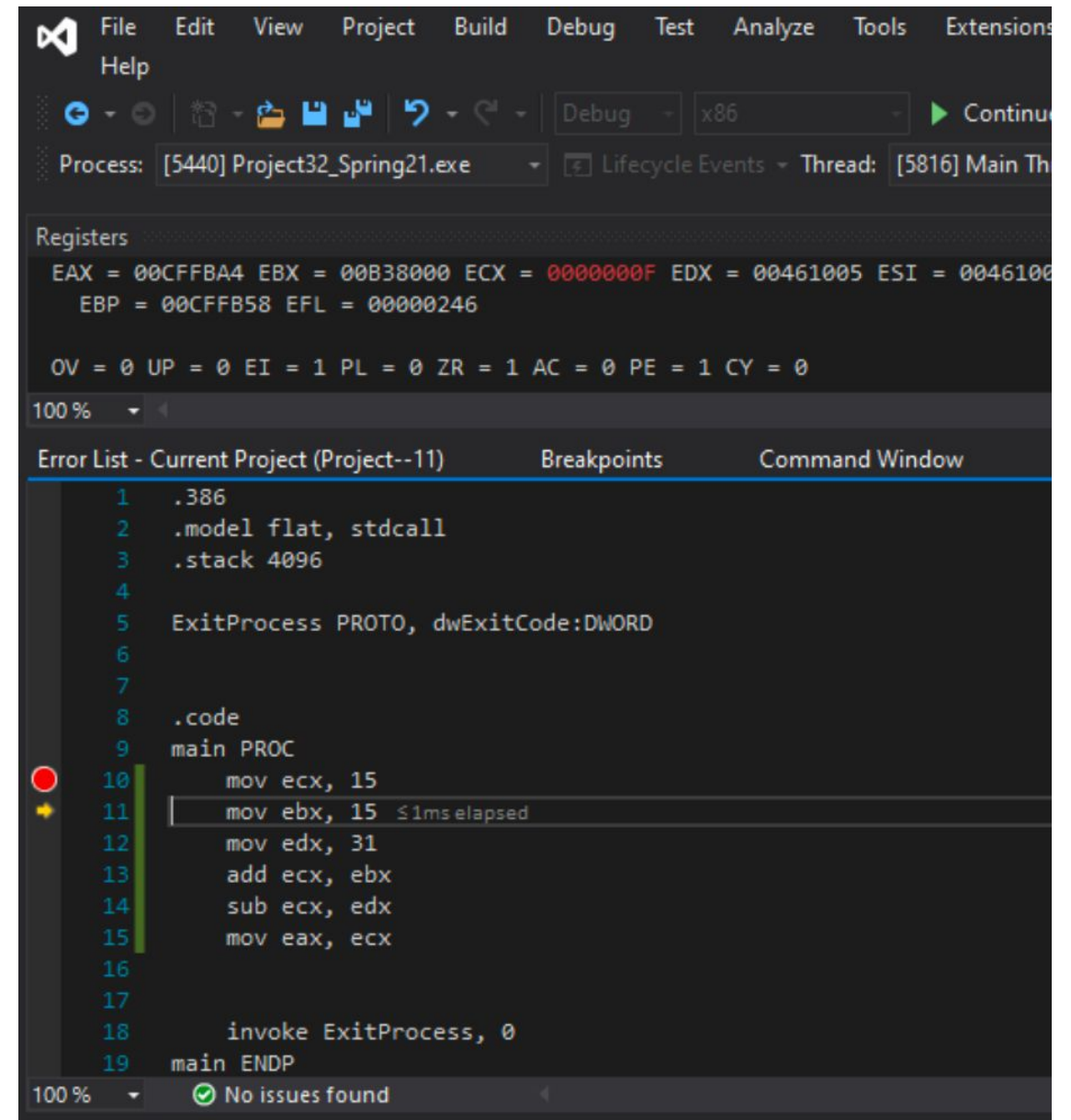
Step 4: Debug the project (6)

- Now the red dot (breakpoint) has a yellow pointer inside of it now.
- That means code execution halts at line 10 now.



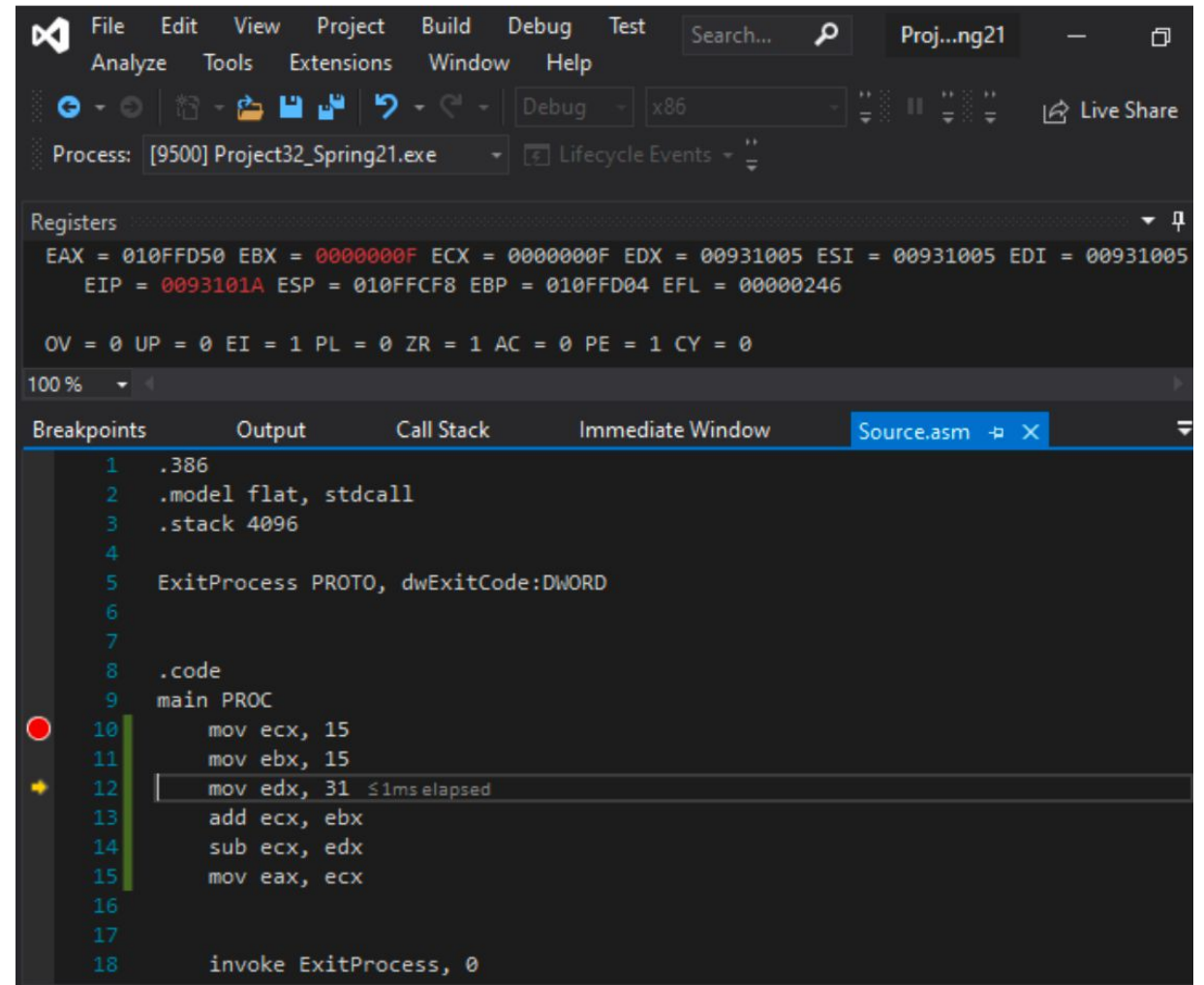
Debug the Project

- To execute line 10, Select 'Step over' from debug menu
 - **OR** You can also use shortcut :
Fn+F10
- Now the yellow pointer moved to line 11. That means line 10 is executed.
- Check the ECX register content.
- You can see the hexadecimal representation of decimal 15.



Debug the Project

- To execute line 11, Select 'Step over' from debug menu
 - **OR** You can also use shortcut :
Fn+F10
- Now the yellow pointer moved to line 12. That means line 11 is executed.
- Check the ECX register content.
- You can see the hexadecimal representation of decimal 15.



Visual Studio Code interface showing the debugger. The 'Registers' window displays the following values:

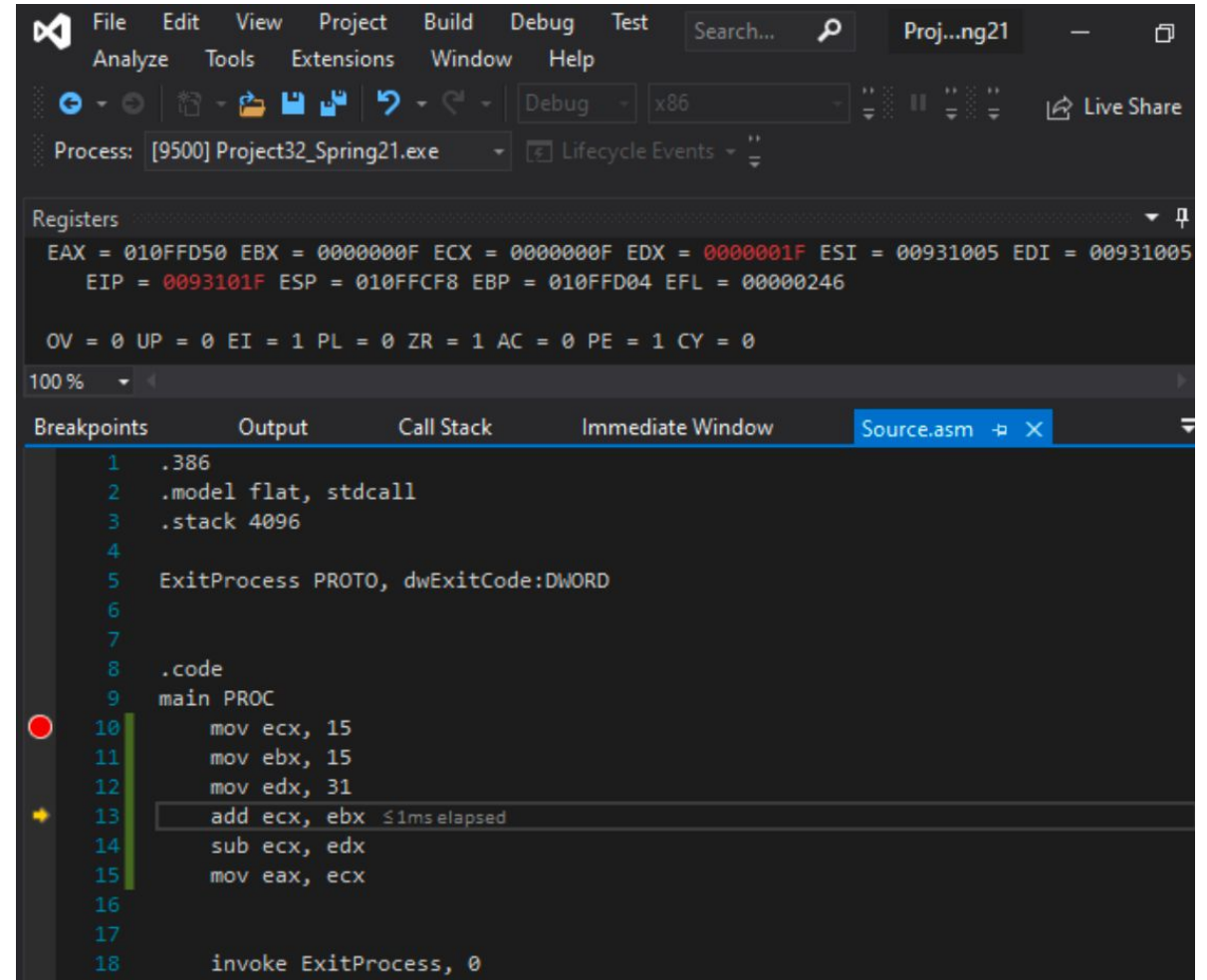
Register	Value
EAX	010FFD50
EBX	0000000F
ECX	0000000F
EDX	00931005
ESI	00931005
EDI	00931005
EIP	0093101A
ESP	010FFCF8
EBP	010FFD04
EFL	00000246

The 'Source.asm' window shows the following assembly code:

```
1 .386
2 .model flat, stdcall
3 .stack 4096
4
5 ExitProcess PROTO, dwExitCode:DWORD
6
7
8 .code
9 main PROC
10     mov ecx, 15
11     mov ebx, 15
12     mov edx, 31
13     add ecx, ebx
14     sub ecx, edx
15     mov eax, ecx
16
17
18     invoke ExitProcess, 0
```


Debug the Project

- To execute line 12, Select 'Step over' from debug menu
 - **OR** You can also use shortcut : Fn+F10
- Now the yellow pointer moved to line 13. That means line 12 is executed.
- Check the EDX register content.
- You can see the hexadecimal representation of decimal 31.

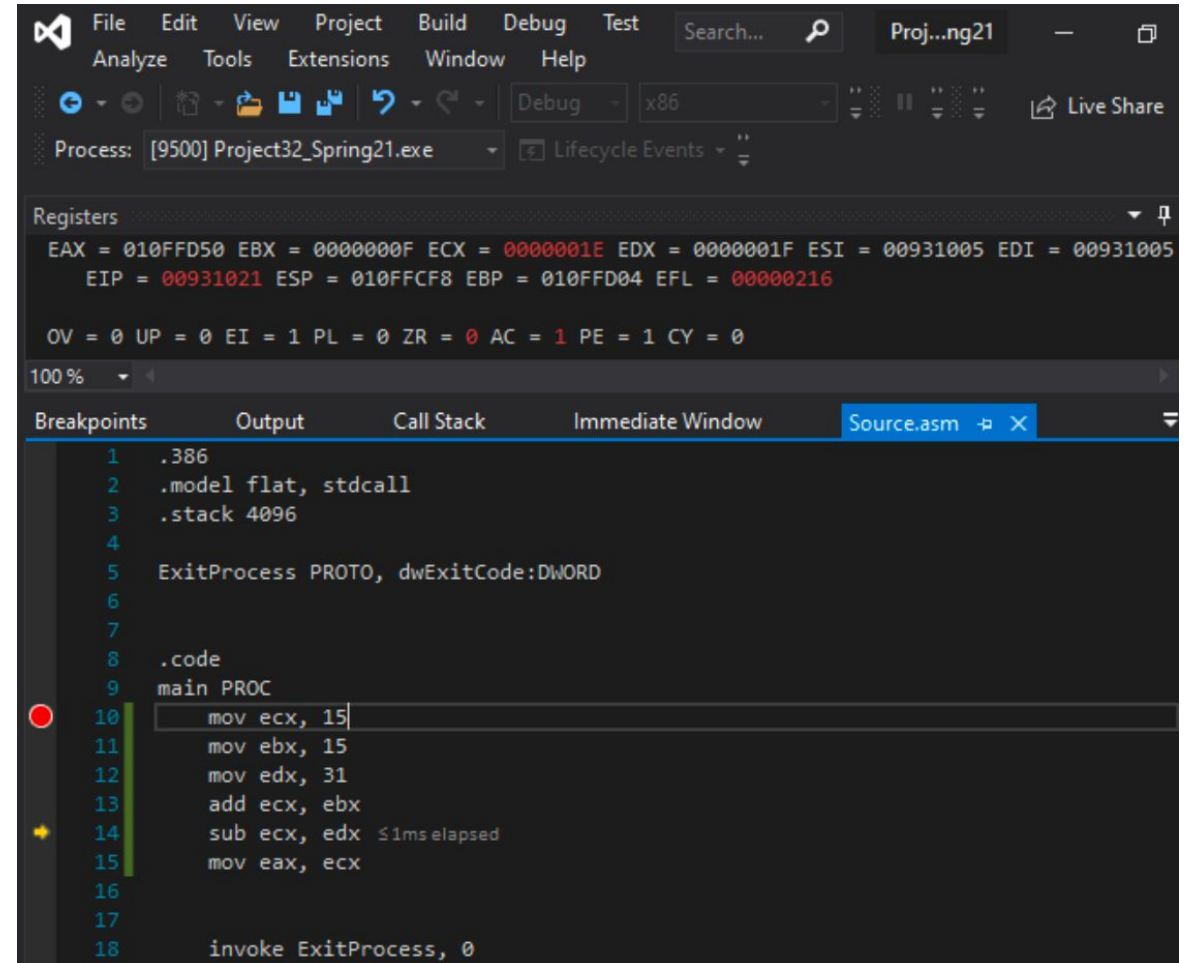


The screenshot shows the Visual Studio Code interface with the debugger active. The top menu bar includes File, Edit, View, Project, Build, Debug, and Test. The toolbar shows various debugging icons, including a search icon and a 'Live Share' button. The 'Process' dropdown is set to '[9500] Project32_Spring21.exe'. The 'Registers' window displays the following values: EAX = 010FFD50, EBX = 0000000F, ECX = 0000000F, EDX = 0000001F, ESI = 00931005, EDI = 00931005, EIP = 0093101F, ESP = 010FFCF8, EBP = 010FFD04, EFL = 00000246. Below the registers, the status bar shows 'OV = 0 UP = 0 EI = 1 PL = 0 ZR = 1 AC = 0 PE = 1 CY = 0'. The 'Source.asm' window is open, showing assembly code. Line 12 is highlighted, and the yellow pointer is on line 13. The code includes comments like '.386', '.model flat, stdcall', '.stack 4096', and 'ExitProcess PROTO, dwExitCode:DWORD'. The main procedure starts with 'main PROC' and contains several instructions: 'mov ecx, 15', 'mov ebx, 15', 'mov edx, 31', 'add ecx, ebx', 'sub ecx, edx', 'mov eax, ecx', and 'invoke ExitProcess, 0'.

```
File Edit View Project Build Debug Test Search... Proj...ng21
Analyze Tools Extensions Window Help
Debug x86
Process: [9500] Project32_Spring21.exe Lifecycle Events
Registers
EAX = 010FFD50 EBX = 0000000F ECX = 0000000F EDX = 0000001F ESI = 00931005 EDI = 00931005
EIP = 0093101F ESP = 010FFCF8 EBP = 010FFD04 EFL = 00000246
OV = 0 UP = 0 EI = 1 PL = 0 ZR = 1 AC = 0 PE = 1 CY = 0
100 %
Breakpoints Output Call Stack Immediate Window Source.asm
1 .386
2 .model flat, stdcall
3 .stack 4096
4
5 ExitProcess PROTO, dwExitCode:DWORD
6
7
8 .code
9 main PROC
10 mov ecx, 15
11 mov ebx, 15
12 mov edx, 31
13 add ecx, ebx 51ms elapsed
14 sub ecx, edx
15 mov eax, ecx
16
17
18 invoke ExitProcess, 0
```

Debug the Project

- To execute line 13, Select 'Step over' from debug menu
 - **OR** You can also use shortcut : Fn+F10
- Now the yellow pointer moved to line 14. That means line 13 is executed.
- Check the ECX register content.
- Add instruction performs (15+15) and stores the result in ECX in hexadecimal, 1E.

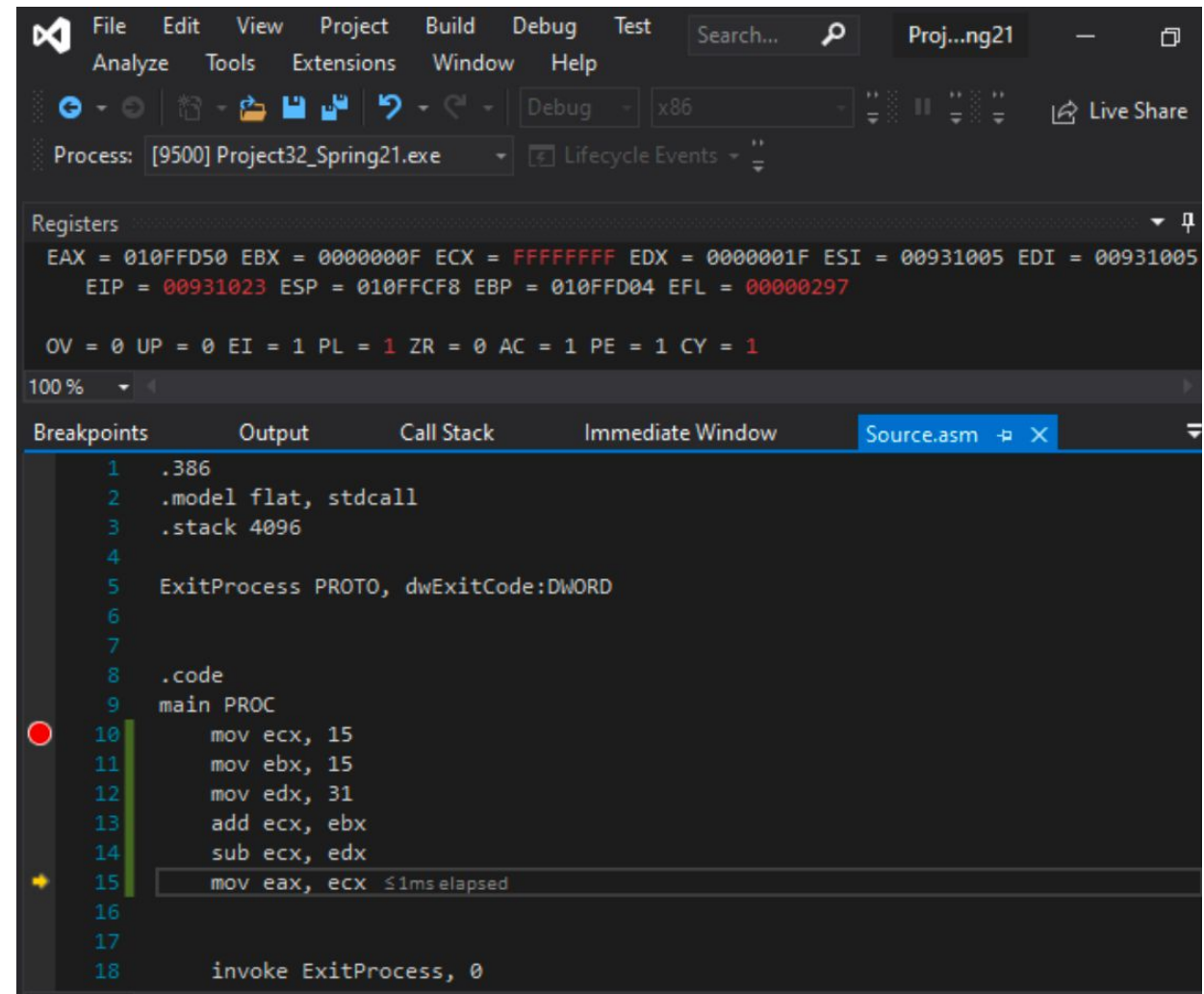


The screenshot shows the Visual Studio Code debugger interface. The top menu bar includes File, Edit, View, Project, Build, Debug, and Test. The toolbar shows various debugging icons. The 'Process' dropdown is set to '[9500] Project32_Spring21.exe'. The 'Registers' window displays the following values: EAX = 010FFD50, EBX = 0000000F, ECX = 0000001E, EDX = 0000001F, ESI = 00931005, EDI = 00931005, EIP = 00931021, ESP = 010FFCF8, EBP = 010FFD04, EFL = 00000216. The 'Source.asm' window shows the following assembly code:

```
1 .386
2 .model flat, stdcall
3 .stack 4096
4
5 ExitProcess PROTO, dwExitCode:DWORD
6
7
8 .code
9 main PROC
10 mov ecx, 15
11 mov ebx, 15
12 mov edx, 31
13 add ecx, ebx
14 sub ecx, edx ≤ 1ms elapsed
15 mov eax, ecx
16
17
18 invoke ExitProcess, 0
```

Debug the Project

- To execute line 14, Select 'Step over' from debug menu
 - **OR** You can also use shortcut : Fn+F10
- Now the yellow pointer moved to line 15. That means line 14 is executed.
- Check the ECX register content and the flags.
- Sub instruction subtracts edx from ecx.
- That is (30-31) in decimal.
- Result = -1
- You can see the result in ECX = FFFFFFFF
- That is 2's complement of -1 in 32-bit.



The screenshot shows the Visual Studio Code interface with the debugger active. The 'Registers' window at the top displays the following values: EAX = 010FFD50, EBX = 0000000F, ECX = FFFFFFFF, EDX = 0000001F, ESI = 00931005, EDI = 00931005, EIP = 00931023, ESP = 010FFCF8, EBP = 010FFD04, EFL = 00000297. Below the registers, the 'Source.asm' window shows the assembly code. Line 14, 'sub ecx, edx', is highlighted with a yellow pointer. The 'Breakpoints' window shows a red dot at line 10. The 'Output' window shows the message '≤ 1ms elapsed'.

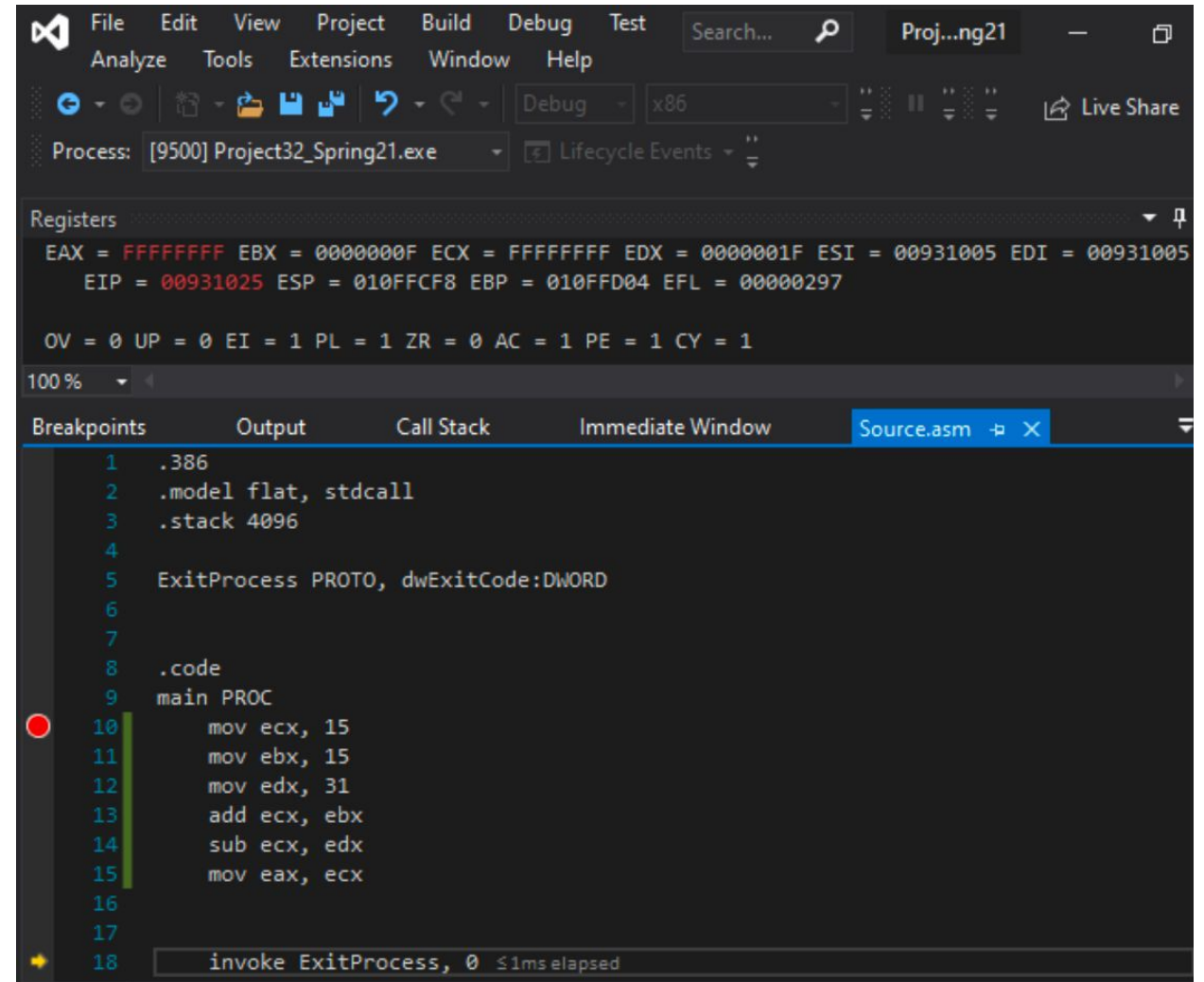
```
.386
.model flat, stdcall
.stack 4096

ExitProcess PROTO, dwExitCode:DWORD

.code
main PROC
    mov ecx, 15
    mov ebx, 15
    mov edx, 31
    add ecx, ebx
    sub ecx, edx
    mov eax, ecx
    invoke ExitProcess, 0
```


Debug the Project

- To execute line 15, Select 'Step over' from debug menu
 - **OR** You can also use shortcut : Fn+F10
- Now the yellow pointer moved to line 16. That means line 15 is executed.
- Check the EAX register content.



The screenshot displays the Visual Studio Code debugger interface. The top menu bar includes File, Edit, View, Project, Build, Debug, and Test. The toolbar shows various debugging icons, including a search icon and a 'Live Share' button. The 'Process' dropdown is set to '[9500] Project32_Spring21.exe'. The 'Registers' window shows the following values: EAX = FFFFFFFF, EBX = 0000000F, ECX = FFFFFFFF, EDX = 0000001F, ESI = 00931005, EDI = 00931005, EIP = 00931025, ESP = 010FFCF8, EBP = 010FFD04, EFL = 00000297. The 'Source.asm' window shows the following assembly code:

```
1 .386
2 .model flat, stdcall
3 .stack 4096
4
5 ExitProcess PROTO, dwExitCode:DWORD
6
7
8 .code
9 main PROC
10     mov ecx, 15
11     mov ebx, 15
12     mov edx, 31
13     add ecx, ebx
14     sub ecx, edx
15     mov eax, ecx
16
17
18     invoke ExitProcess, 0 ≤1ms elapsed
```

Debug the Project

- After you reach the "INVOKE ExitProcess, 0"
 - Do not click Step over or Fn+F10
- What is the status of the sign flag (PL) ?

Lab 4

Submission Instruction

Problem 1

- Assume that you have 8-bit registers.
- Write and run a program to evaluate the following expression:

$$AL = (AL - DL) + CL - BL$$

- Store the following decimal values into the registers
- Store 245 (decimal) to AL register
- Store 41 (decimal) to BL register
- Store 11 (decimal) to CL register
- Store 215 (decimal) to DL register
- Evaluate the above expression
- Result should be at AL register at the end

Submission Instruction

- There is an answer sheet attached to the lab (similar to lab-3b)
- Debug through each line of your code.
 - Execute the instruction
 - Take a screenshot of the code and register window
 - Record the line number, instruction, Register values in the answer sheet.
 - Also add the screenshot
 - Then explain the register contents. (similar to lab-3b)