LAB 01

Computer Organization & Assembly Language (COAL) | EL 2003

Student Name: _____

Roll No.: _____

Section: _____

Marks Awarded _____

Prepared by: Talha Shahid

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES (NUCES), KARACHI

Lab Session 01: CONFIGURATION OF VISUAL STUDIO 2022 COMMUNITY EDITION

Objectives:

- Introduction to Microsoft Visual Studio 2022 Community Edition
- An introduction to Assembly Language
- Understanding the Microsoft Visual Studio 2022 Community Edition
- Configuring Microsoft Visual Studio 2022 Community Edition for Microsoft Macro Assembler (MASM)
- Running a Test Program

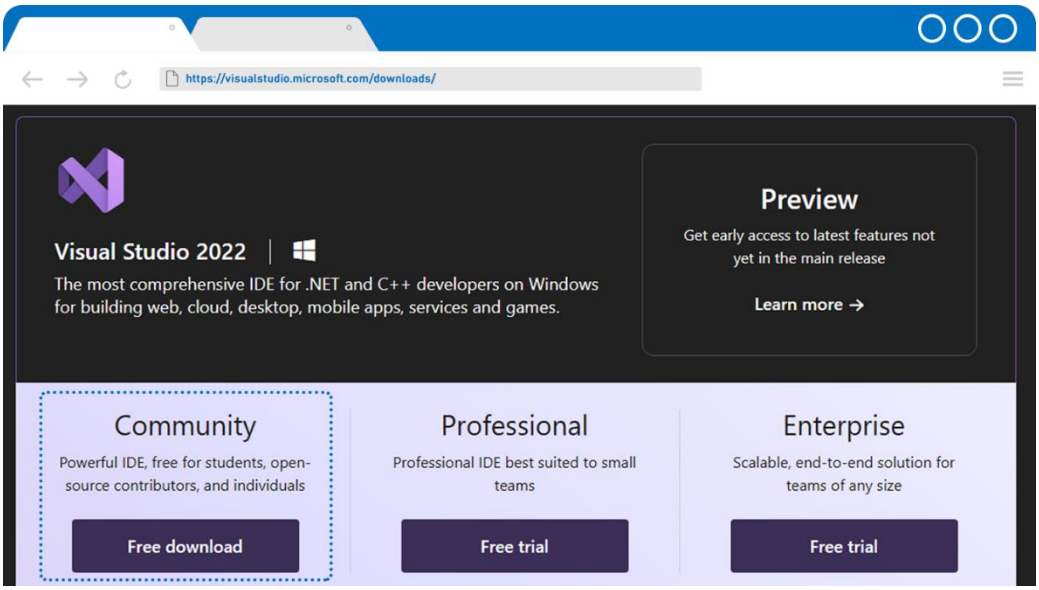
Section 01 Introduction to Microsoft Visual Studio 2022 Community Edition

Visual Studio 2022 Community Edition is a fully featured, extensible, free IDE for creating modern applications for Android, iOS, Windows, as well as web applications and cloud services, and we will use it to create programs in Assembly Language. We could, however, use a stand-alone assembler like NASM or MASM to code in Assembly Language.

Installation Process

Click on the following link below and download the IDE:

link: Microsoft Visual Studio 2022 Community Edition



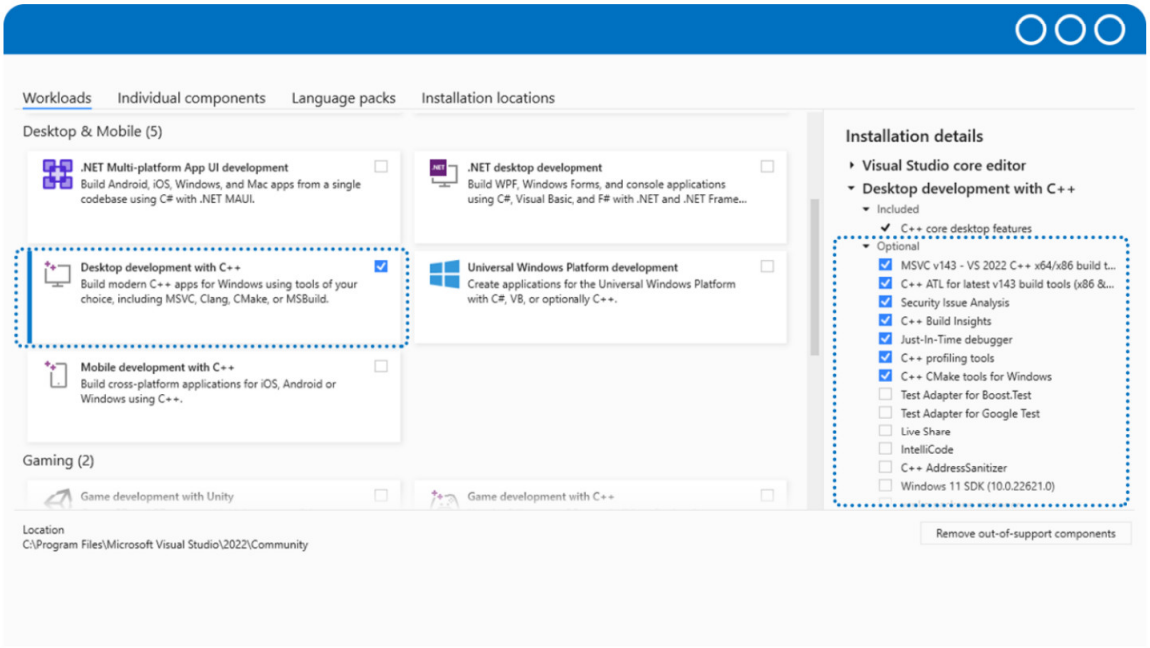
Run the downloaded installer for Microsoft Visual Studio 2022 Community Edition.

- The installer will download some files from the internet and make those files ready to proceed the installation.

After the files are downloaded and installed, the main installation window will open.

Select the following components:

1. From the "Workloads" tab:
    a. Desktop Development with C++
2. Under the section "Installation details", from the "Optional" drop down menu:
   (you may have to scroll to select the following listed components)
    a. MSVC v143 – VS 2002 C++ x64/86 build tools (Latest)
    b. C++ ATL for Latest v143 build tools (x86 & x64)
    c. Security Issue Analysis
    d. C++ Build Insights
    e. Just-In-Time debugger
    f. C++ proofing tools
    g. C++ CMake tools for Windows
    h. Windows 11 SDK OR Windows 10 SDK (based on your Operating System)
    i. MSVC v142 – VS 2019 C++ x64/86 build tools (v14.29)



3. Click on "Install" button.
   (Sometimes individual components failed to install during the installation. The installer then opens a window dialog box to retry manually. Just click on "Retry" and you are good to go.)

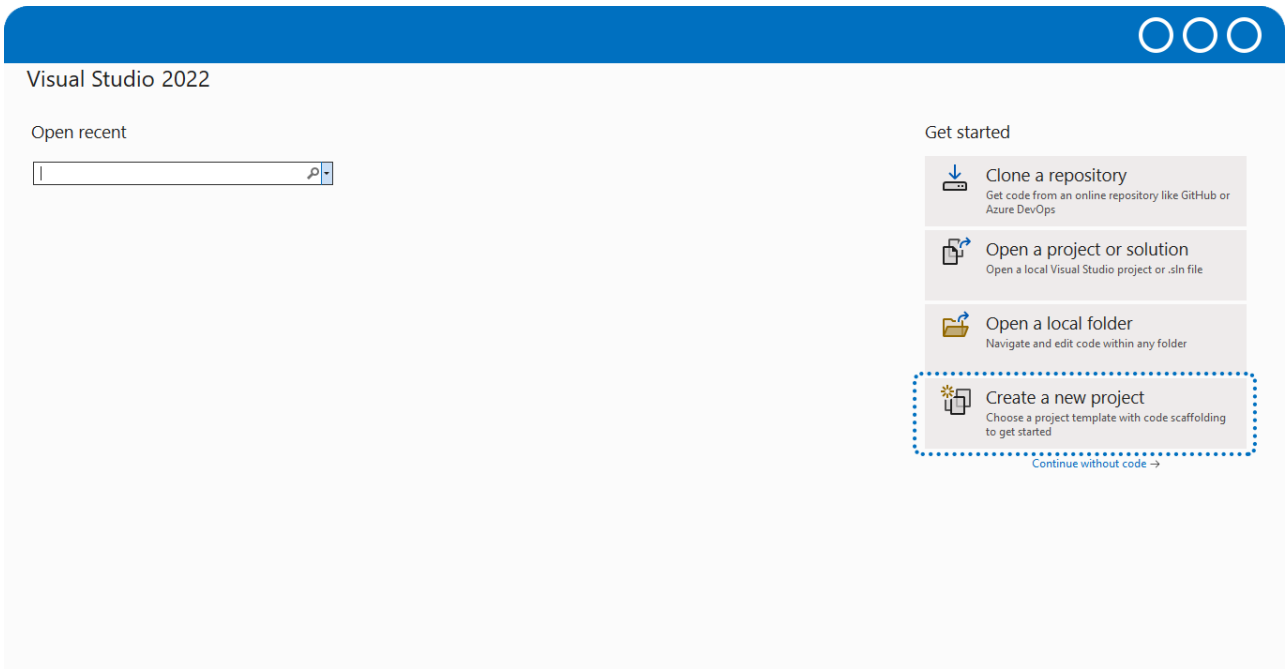
Download the “Irvine Library” for Visual Studio 2022 Community Edition

link: Irvine

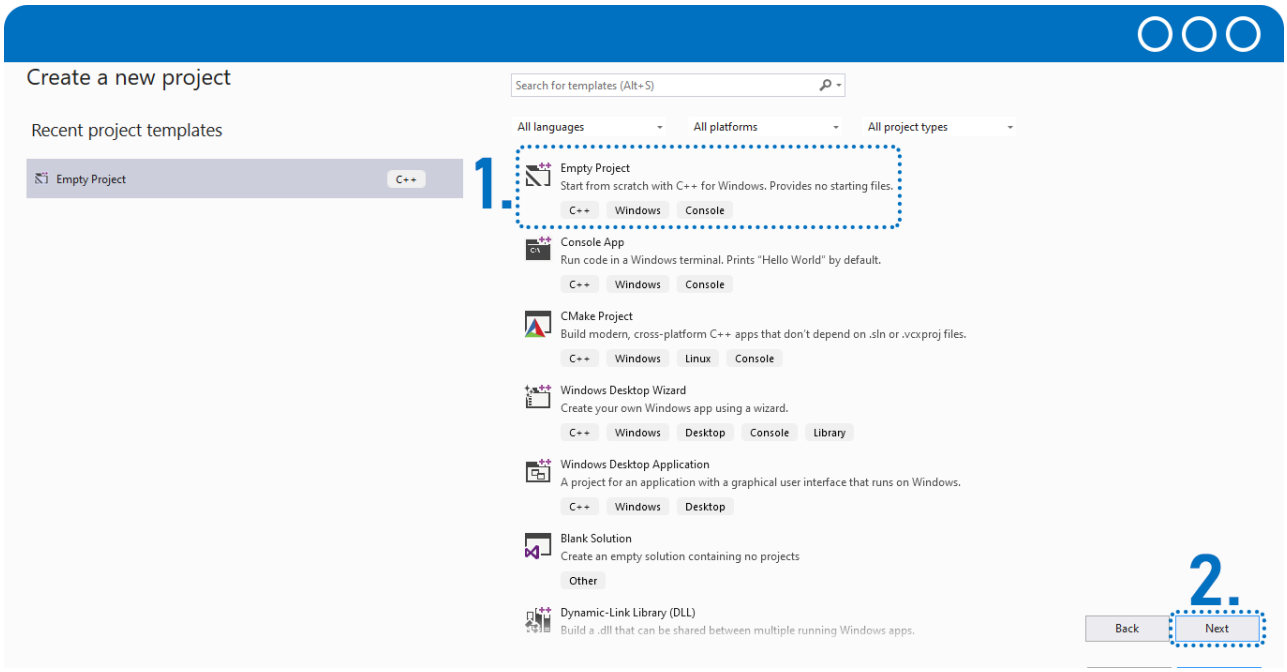
- Extract the downloaded “.zip” file with name “Irvine” and copy the sub folder named “Irvine” into your “C” drive so that the location of “Irvine” should be “C:\Irvine”.

Configure Visual Studio Community Edition 2022 for Assembly Language

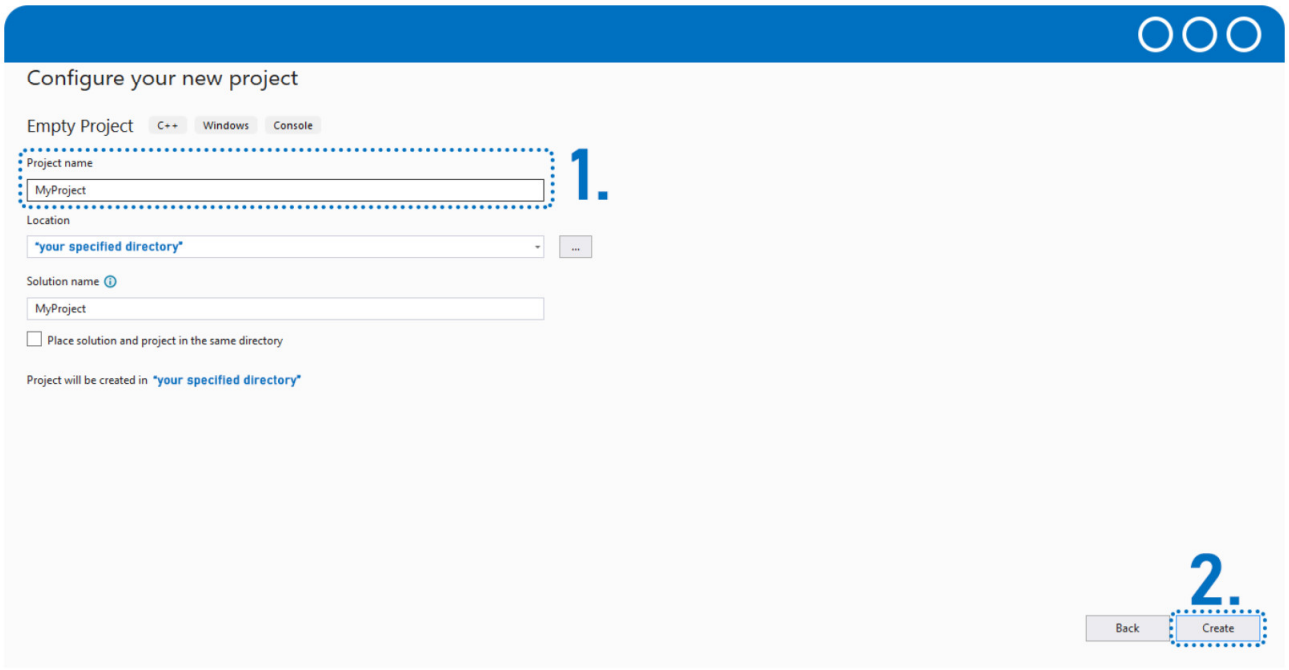
1. Open the Visual Studio 2022 Community Edition and click on “Create a new project”.

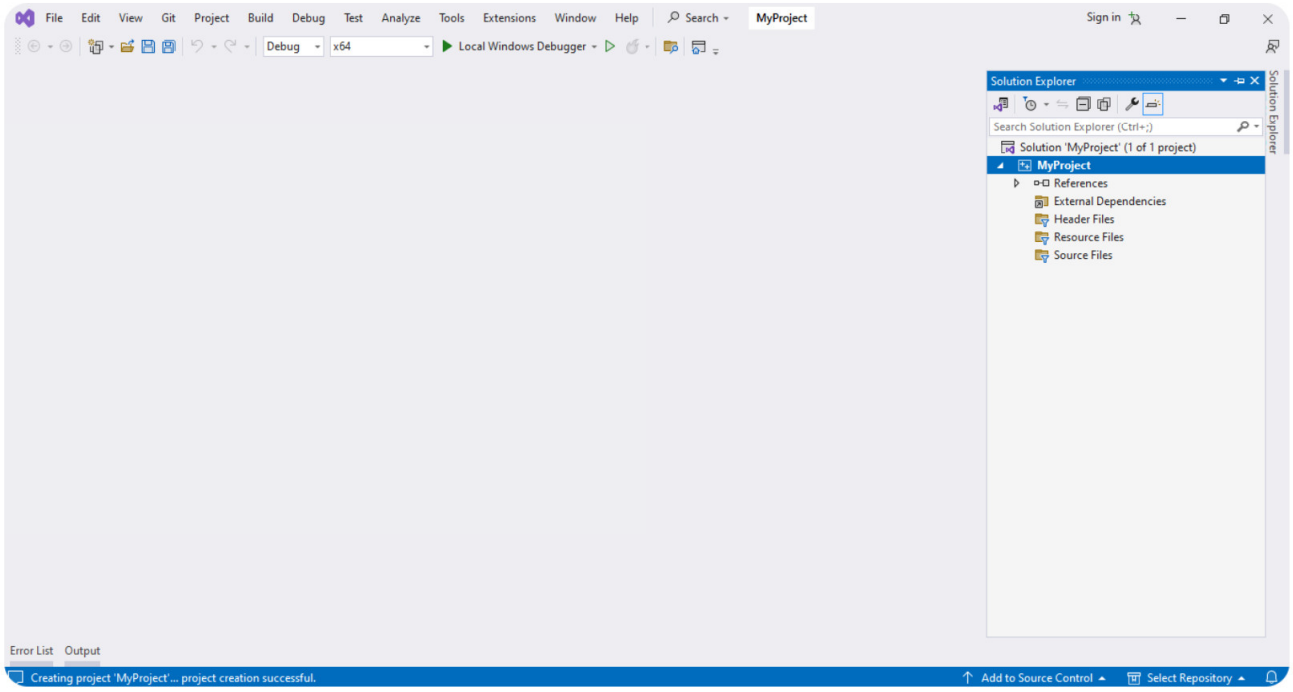


2. Create an “Empty Project” and click on “Next”

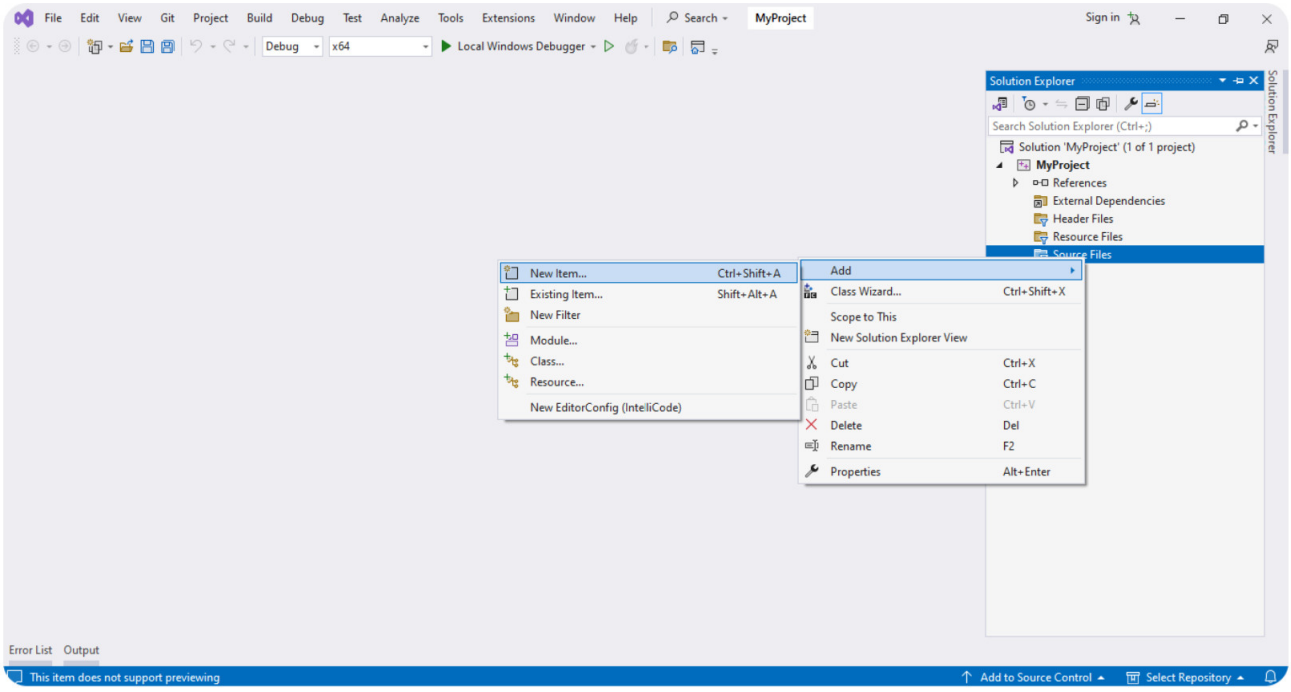


3. Type the “Project name”. You can change the “Location” of the project, but it is better to leave it to the default “Location”. Click on “Create”

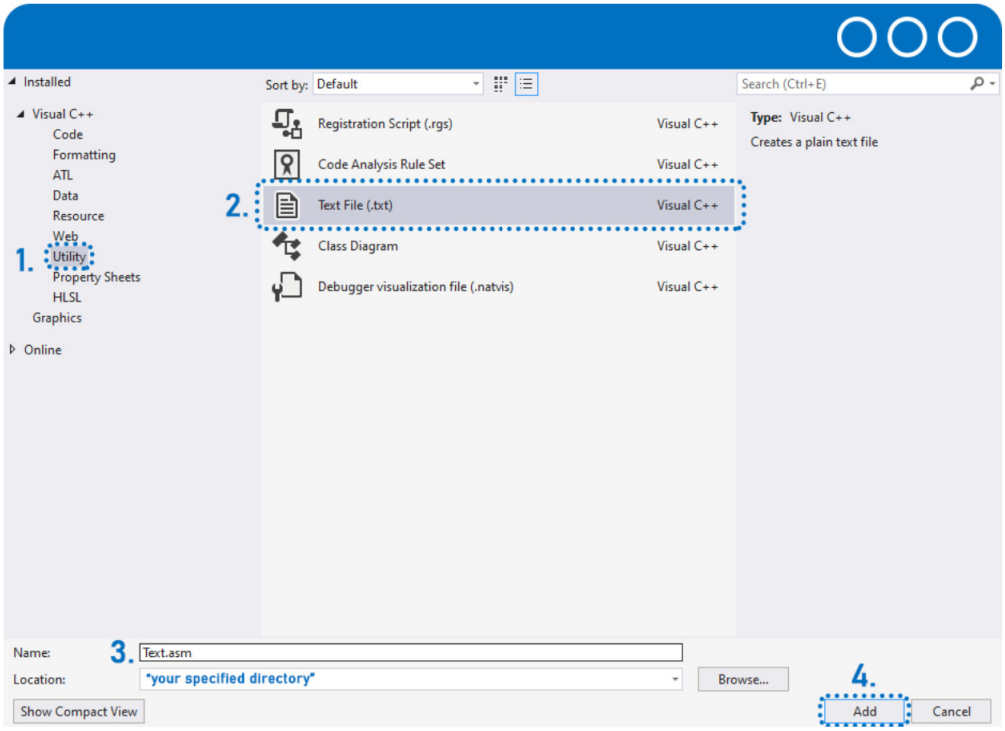


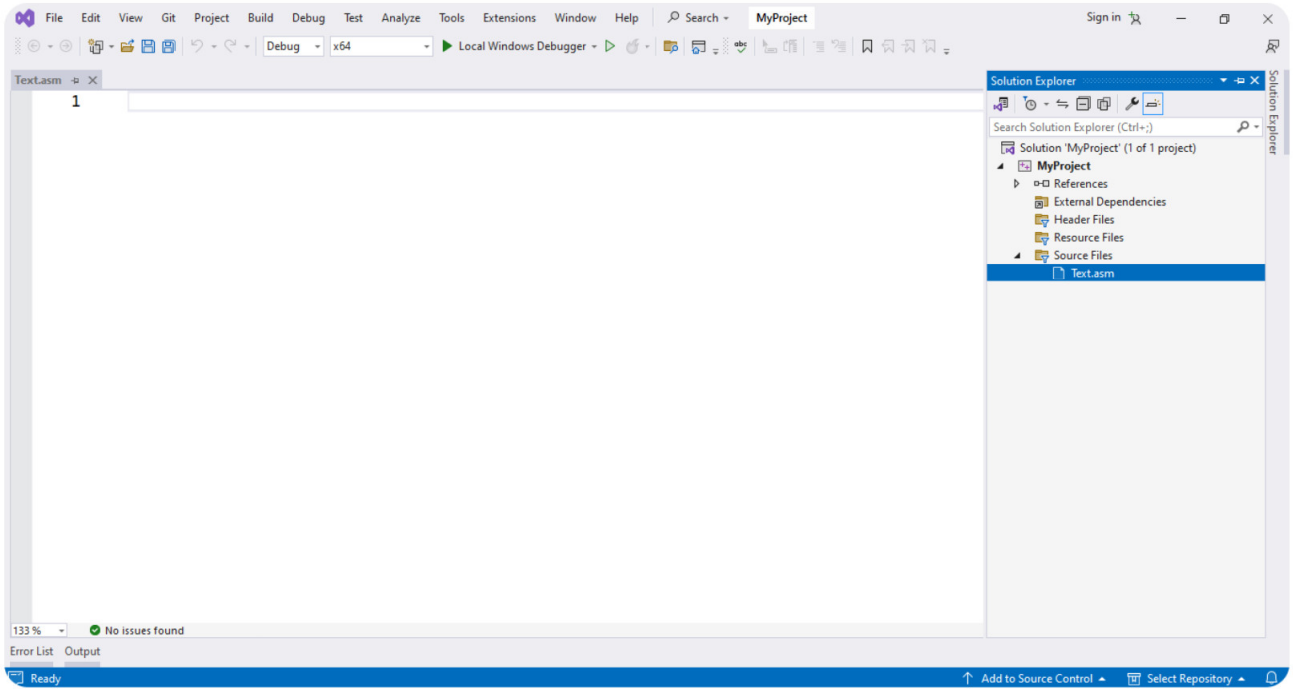


4. In the “Solution Explorer” (Ctrl+Alt+L):
    a. right-click on “Source Files”
    b. click on “Add”
    c. click on “New Item”

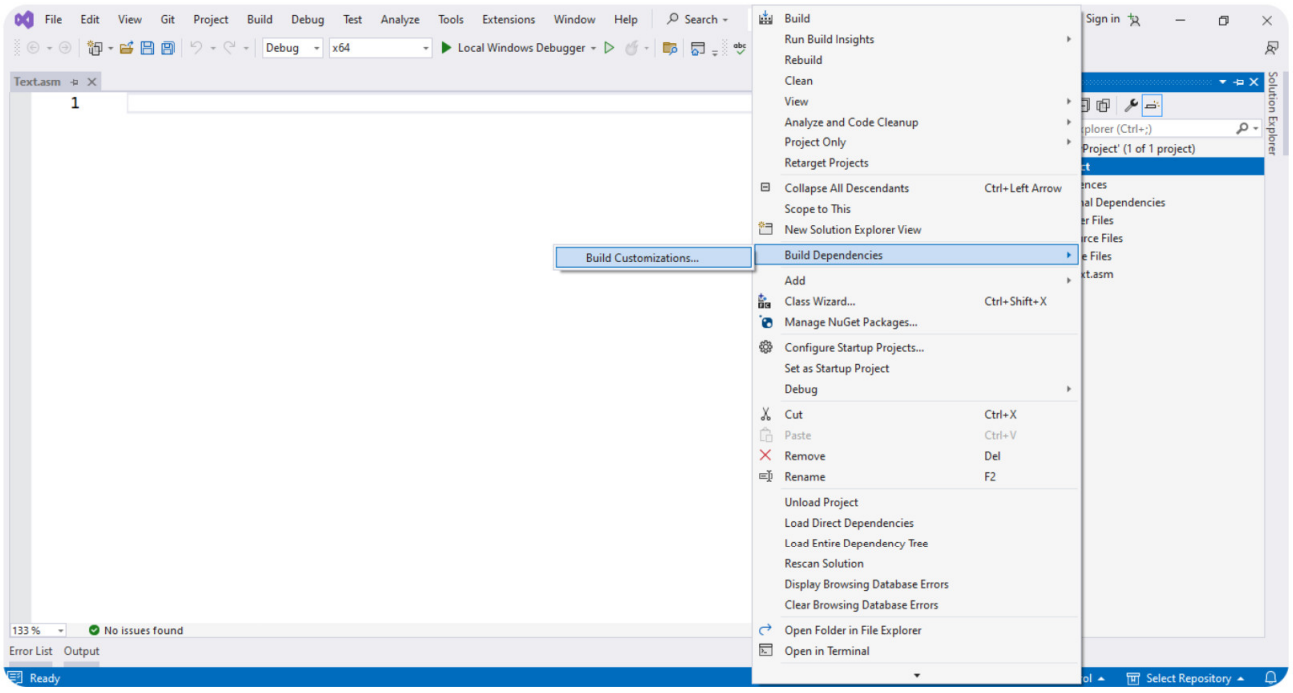


5. From the “Utility” section:
    a. Select “Text File (.txt)”
    b. Type the “Name” of your file, followed by “.asm” extension (important)
    c. Click on “Add”

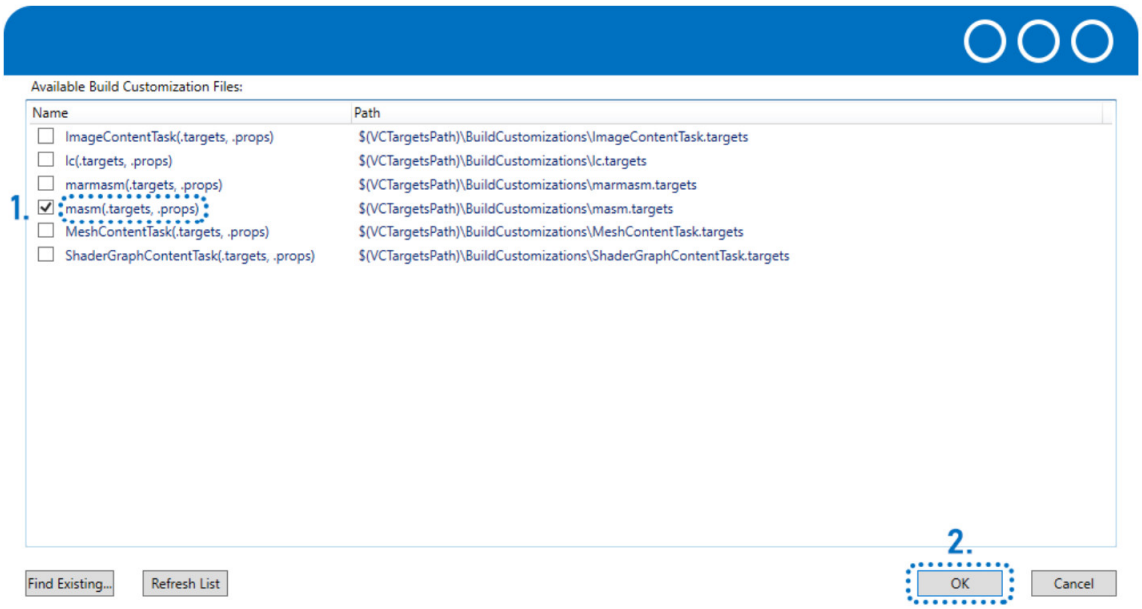




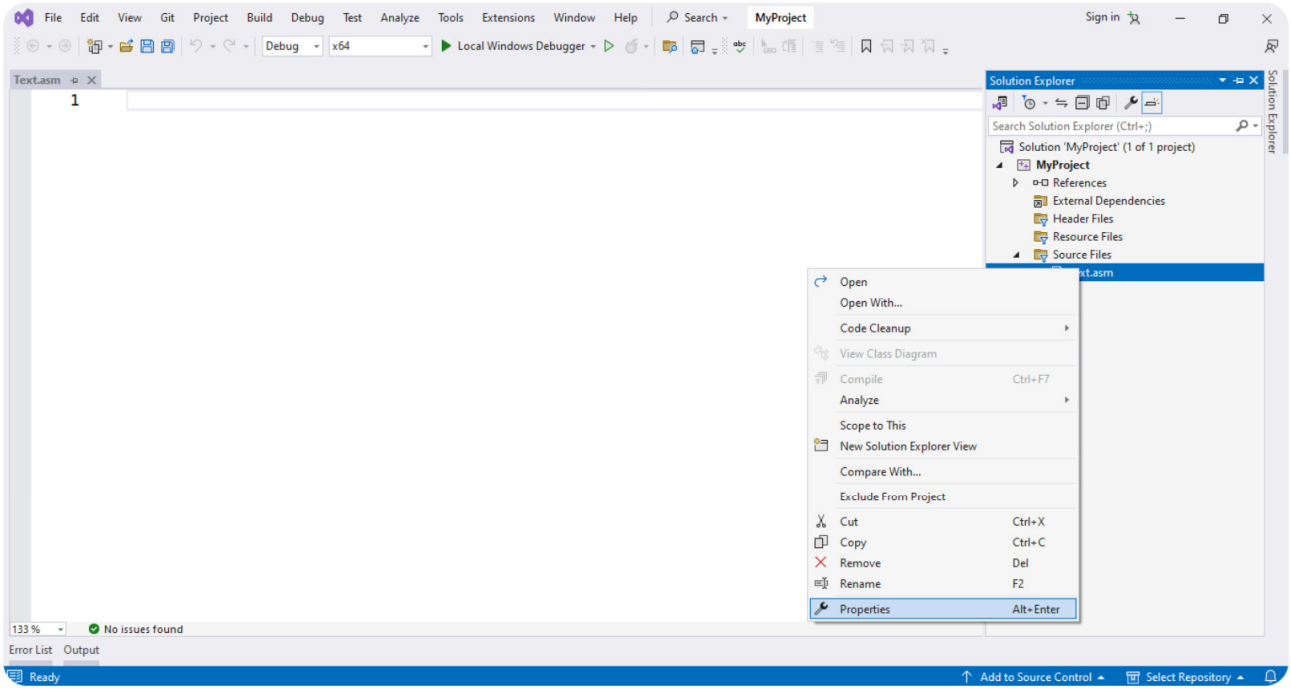
6. Under the “Solution” tab:
    a. right-click on “MyProject” (the name of your project)
    b. click on “Build Dependencies”
    c. click on “Build Customizations…”



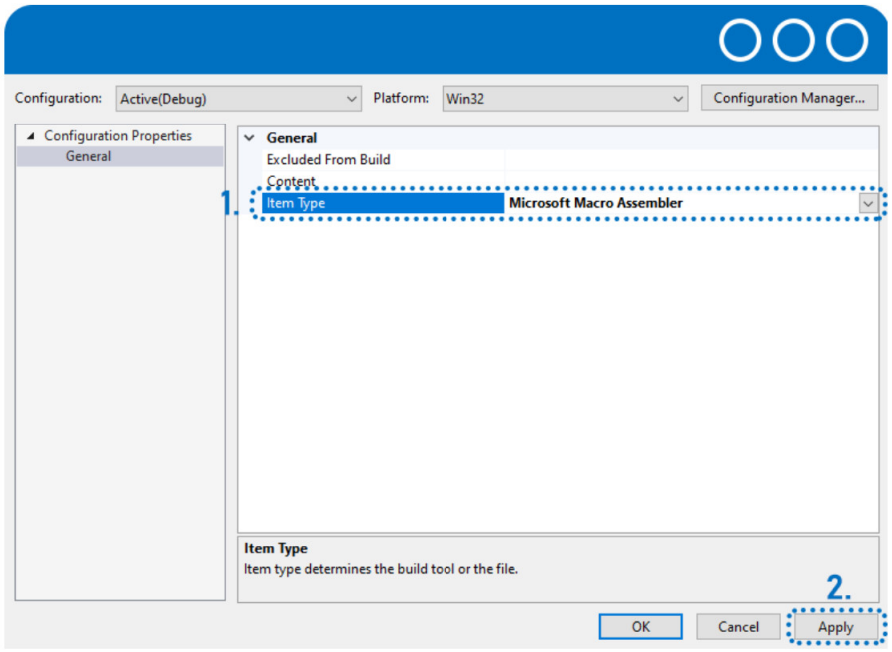
7. Select “masm(.target, .props)” then click on “Ok”



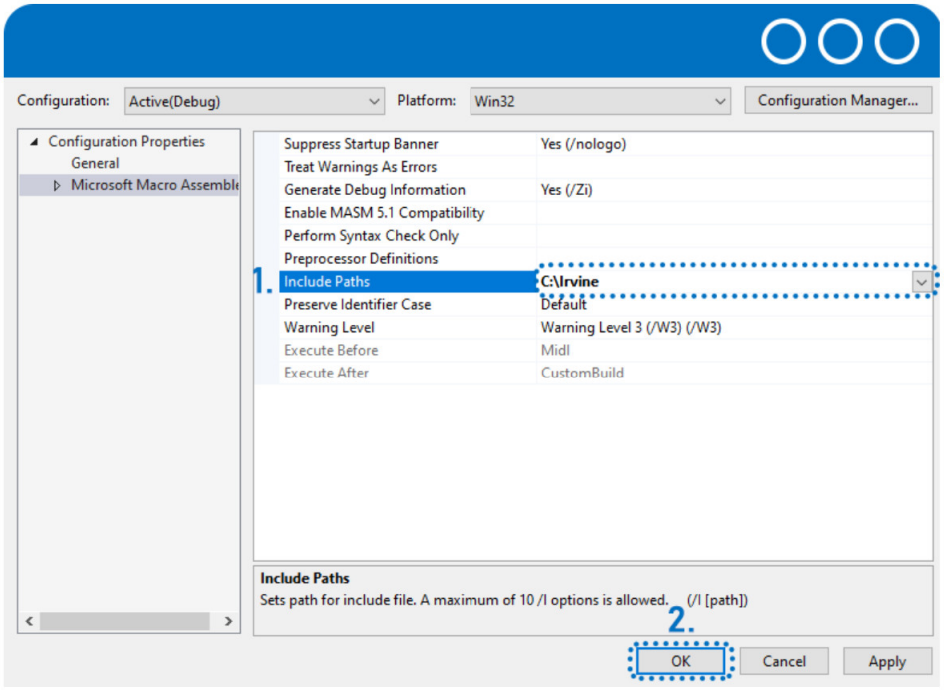
8. Under the “Source Files”
   a. right-click on the file you have created earlier with “.asm” extension (in step no. 5)
   b. click on “Properties”



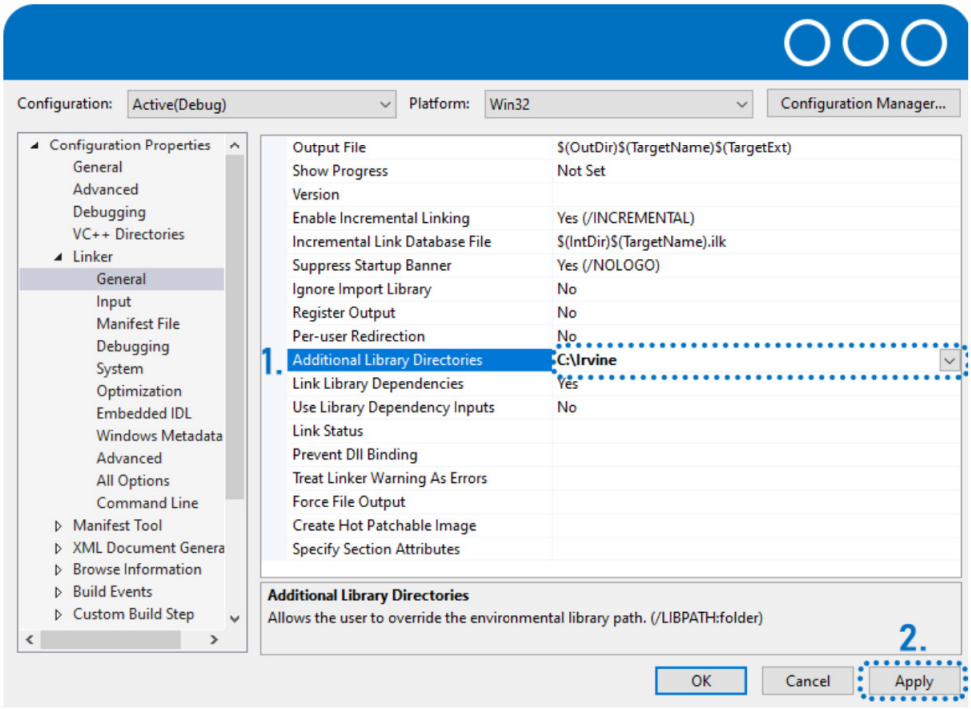
9. Under “General”:
    a. click on “Item Type”
    b. select “Microsoft Macro Assembler” (MASM) from the drop down menu
    c. click on “Apply”



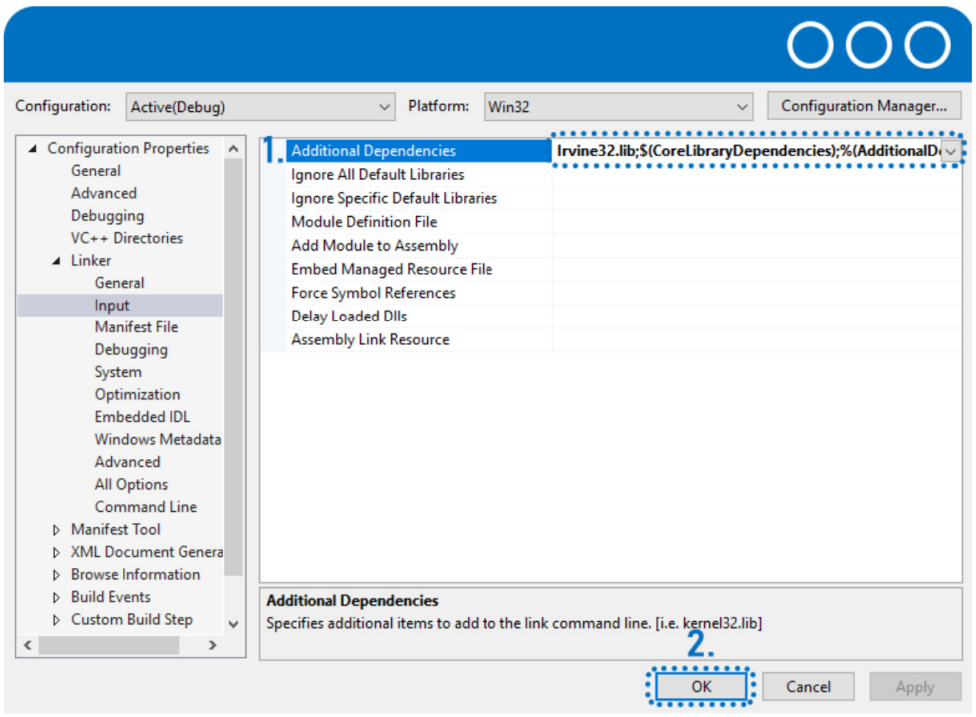
10. Under “Microsoft Macro Assembly”:
    a. click on “Include Paths”
    b. enter the directory of the “Irvine” library which should be “C:\Irvine” (or you can enter the directory where you have extracted the “Irvine” library’s folder)
    c. click on “Ok”



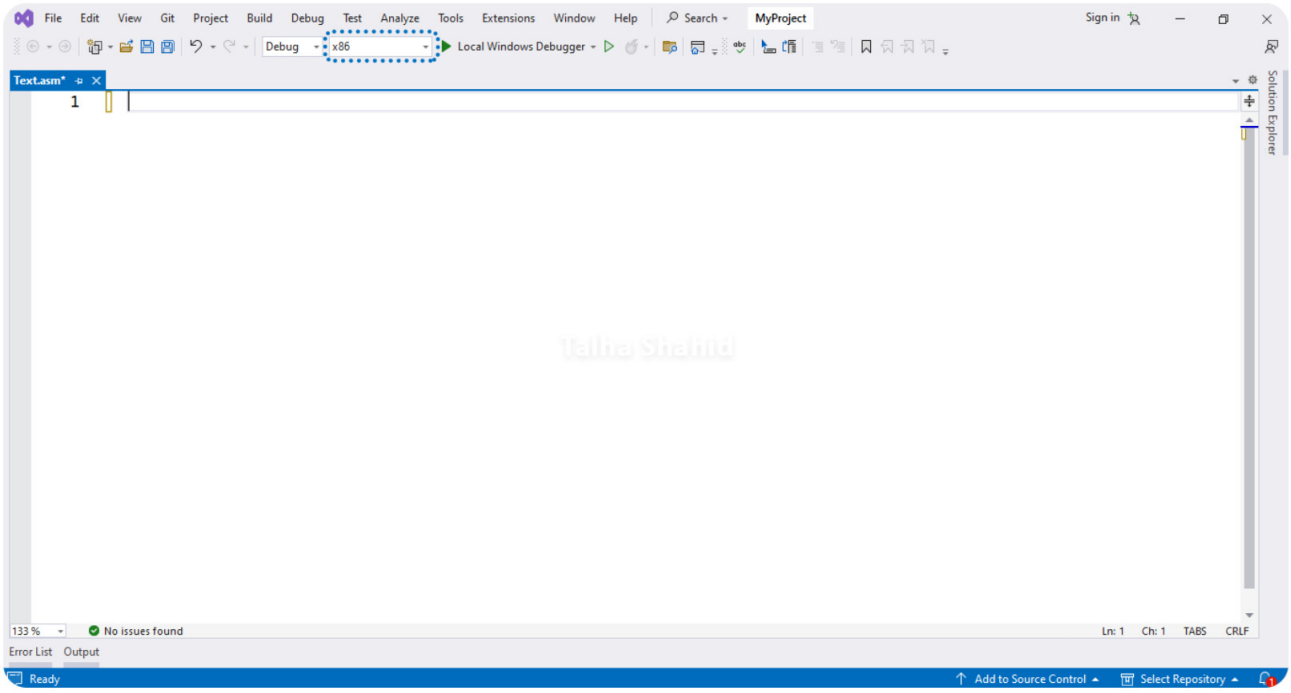
11. Under the “Solution” tab:
    a. right-click on “MyProject” (the name of your project)
    b. click on “Properties”



12. Under “Linker”:
    a. click on “General”
        i. in “General”, click on “Additional Library Directories”
        ii. enter the location of the “Irvine” library which should be “C:\Irvine” (or you can enter the directory where you have extracted the “Irvine” library’s folder)
        iii. click on “Apply”
    b. click on “Input”
        i. in “Input”, click on “Additional Dependencies”
        ii. add the name of “Object File Library” named “Irvine32.lib;” inside the “Irvine” library’s folder at the beginning
        iii. click on “Ok”



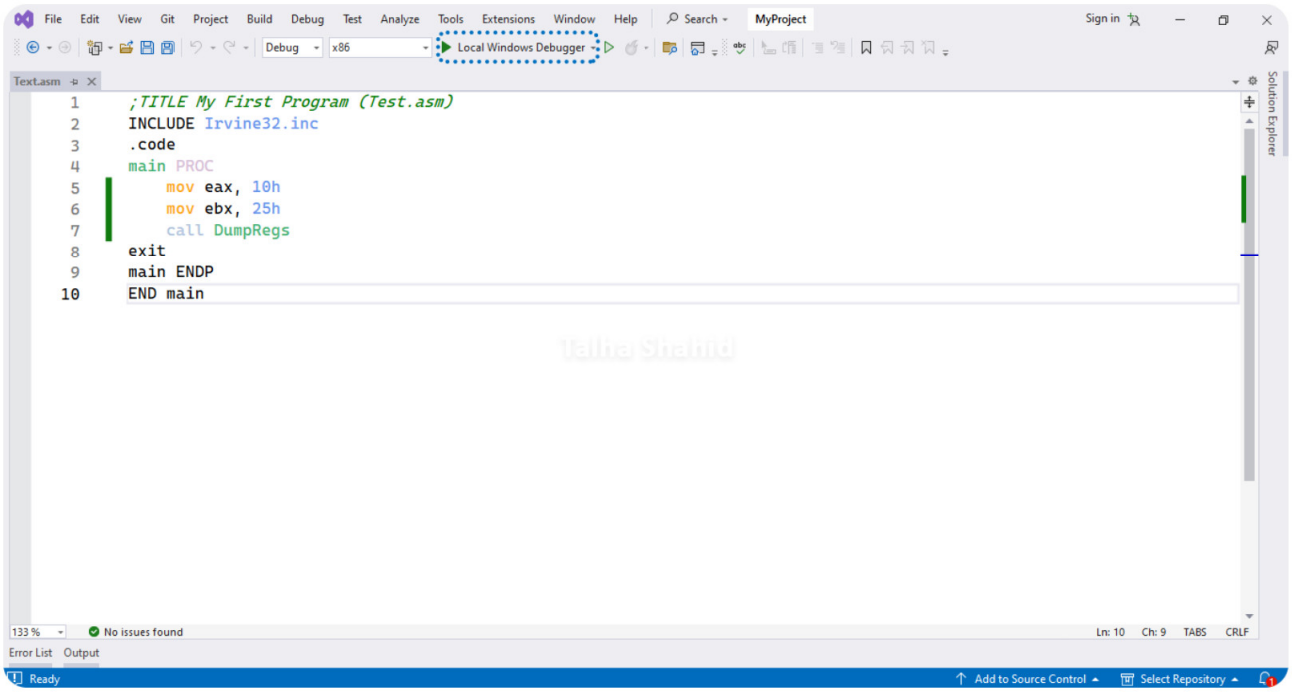
13. Select “x86” from the “Solution Platforms”

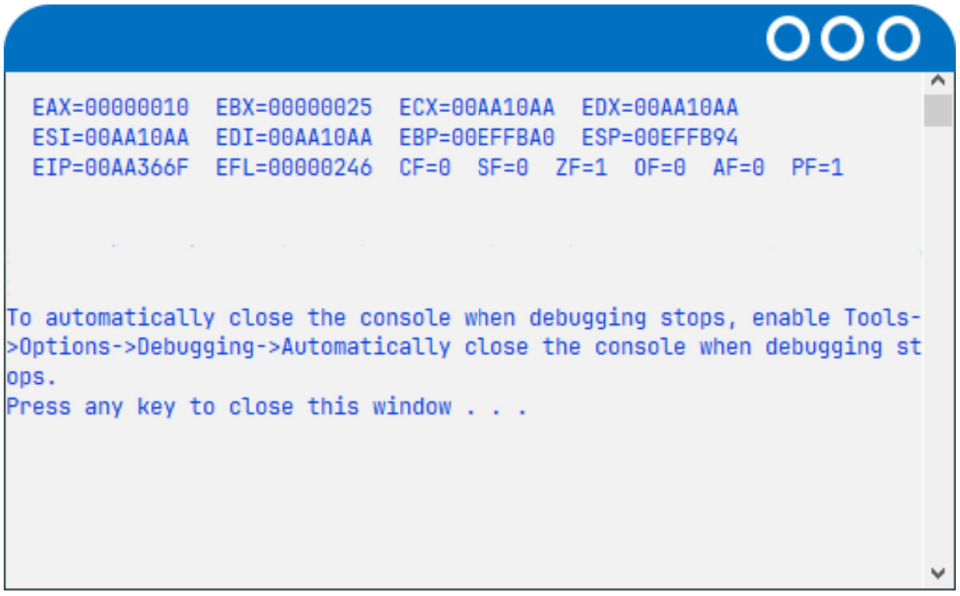


14. Restart Visual Studio

Run Your First Assembly Code

```
;TITLE My First Program (Text.asm)
INCLUDE Irvine32.inc
.code
main PROC
      mov eax, 10h
      mov ebx, 25h
      call DumpRegs
exit
main ENDP
END main
```





Press Ctrl+F5 to see the output in console window.

- As we can see in the output window, the program has affected two registers eax & ebx. Let us dissect our code line by line to see what it does.
- The first line `;TITLE MyFirstProgram (Text.asm)` gives an optional title to our program.
- The second line `INCLUDE Irvine32.inc` adds a reference to the include file that links your program to the Irvine library.
- The third line `.code` defines the beginning of the code segment (to be covered in detail later). The code segment is the segment of memory where all your code resides.
- In the fourth line, a `main PROC` (procedure) is defined.
- The fifth and sixth lines show a mnemonic `mov` (to be covered in detail later) that 'moves' values `10h` and `25h` to `eax` and `ebx`, respectively. The radix `h` defines a hexadecimal constant.
- The lines seven and eight lines calls the procedure `DumpRegs` that outputs the current values of the registers followed by a call to windows procedure named `exit` that halts the program.
- The lines nine `main ENDP` and ten `END main` mark the end of the main procedure.

Section 02 | Debugging our Program

We have seen how to configure Visual Studio 2022 Community Edition for Assembly Language and tested it with a sample program. The output of our sample program was displayed using a console window, but it is usually more desirable to watch the step-by-step execution of our program with each line of code using breakpoints.

Let us briefly define the keywords relevant to debugging in Visual Studio and then we will cover an example for understanding.

Debugger

The (Visual Studio) debugger helps us observe the run-time behavior of our program and find problems. With the debugger, we can break execution of our program to examine our code, examine and edit variables, view registers, see the instructions created from our source code, and view the memory space used by our application.

Breakpoint

A breakpoint is a signal that tells the debugger to temporarily suspend execution of your program at a certain point. When execution is suspended at a breakpoint, your program is said to be in break mode.

Code Stepping

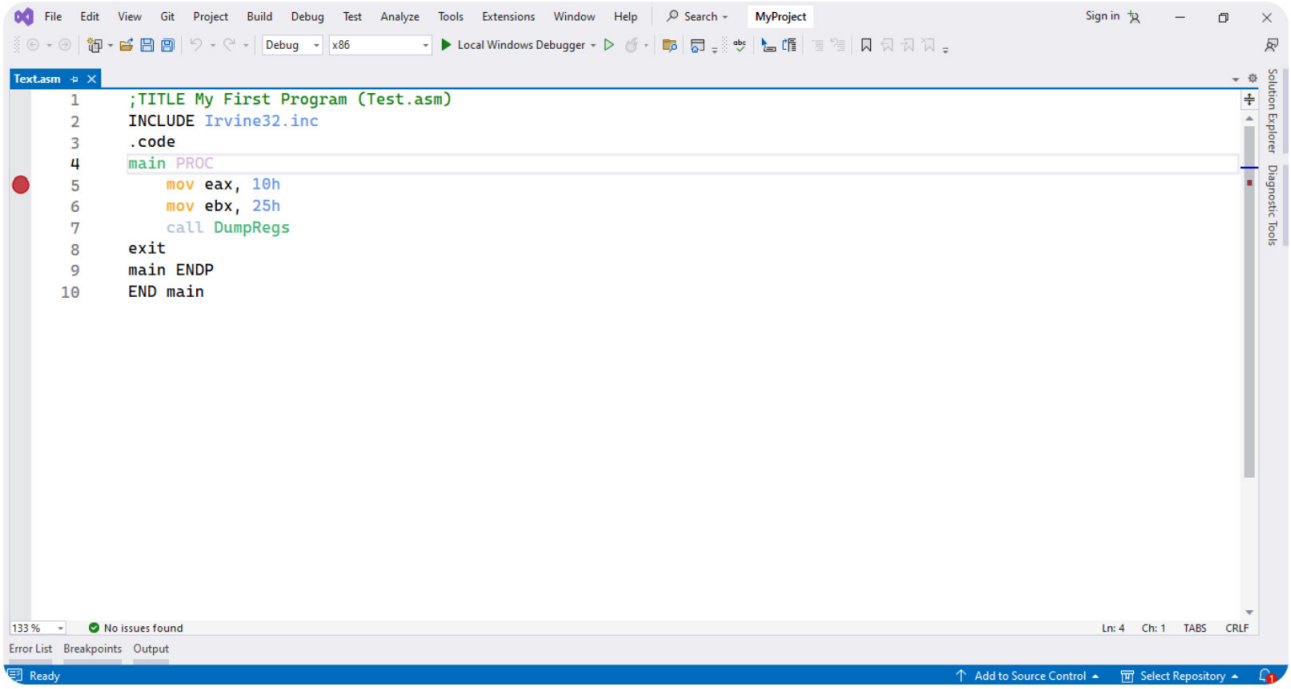
One of the most common debugging procedures is stepping: executing code one line at a time. The Debug menu provides three commands for stepping through code:

- Step Into (By pressing F11)
- Step Over (By pressing F10)
- Step Out (Shift+F11)

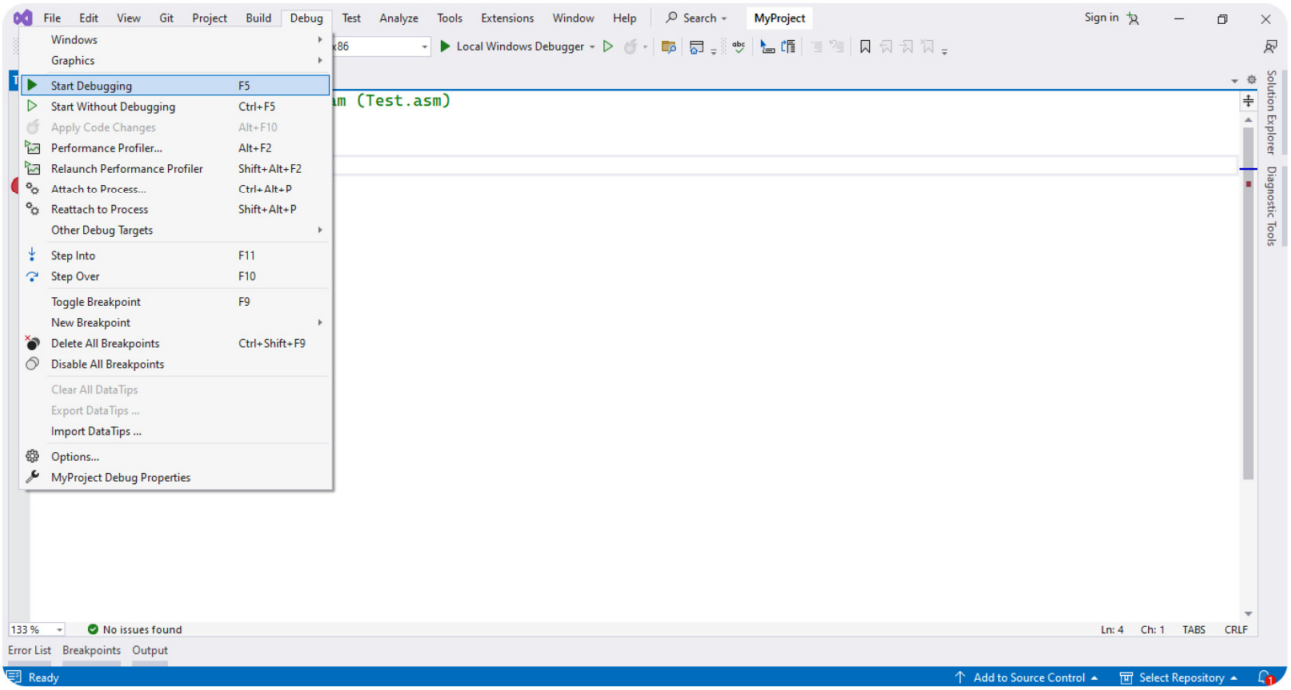
Single Stepping

To see the values of internal registers and memory variables during execution, let us use an example. Copy the following code onto your "Text.asm" file.

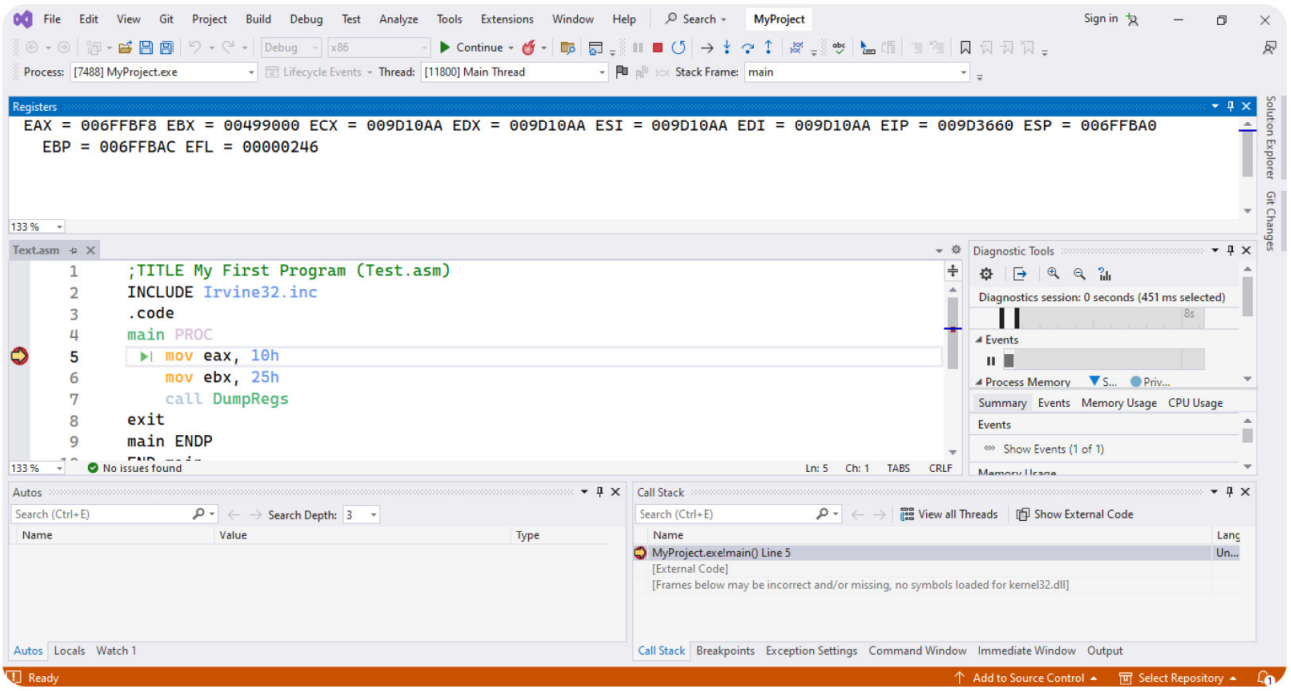
1. Right-Click on line 5 to create a breakpoint OR hover over the cursor at line and insert the breakpoint.



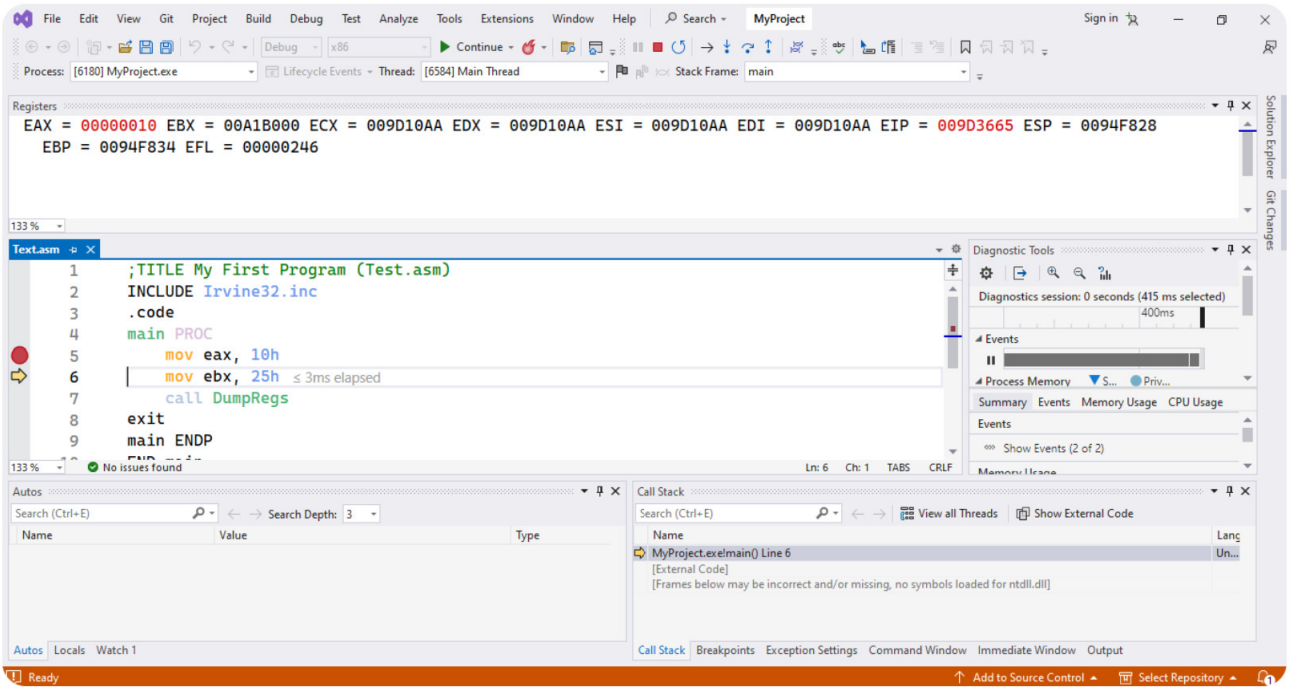
2. Click on Debug tab from the toolbar, select Start Debugging (F5).

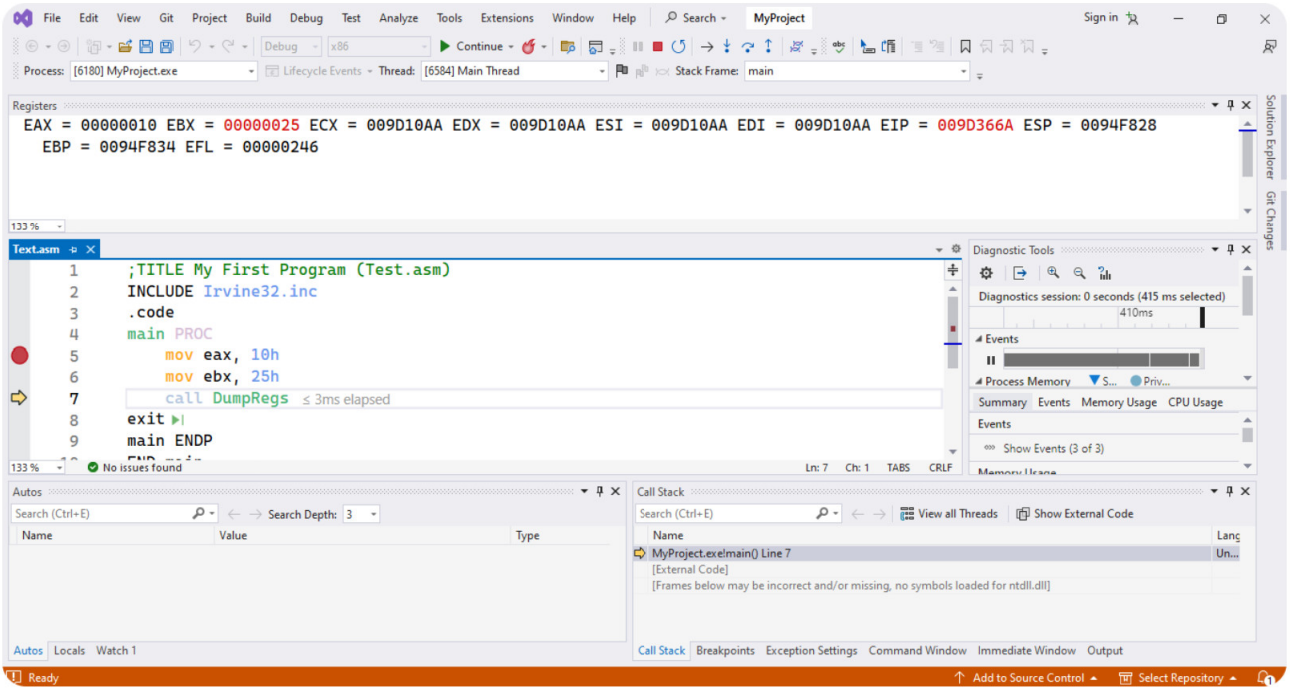


3. Click on “Debug” tab, select “Windows” and then select “Registers” option OR press “CTRL + ALT + G”.



4. Press F11 to step into each instruction. You can see the reflected stepping for each instruction in “Editor” and in the “Registers” windows.





5. The stepping will terminate when it reaches the line with a call to “exit”.

Section 02: Exercise

1. Install Visual Studio 2022 Community Edition and create a new empty project.
2. Configure the project for Assembly using the steps show in this lab.
4. Run a test program in console window by changing the value of EAX in line 6 to 8500h.
5. Debug the below program and note down the values of all the registers after the execution of each line.

```
TITLE My First Program (Text.asm)
INCLUDE Irvine32.inc
.code
main PROC
     mov eax, 47h
     mov ebx, 39h
     mov ecx, 60h
     add eax, ebx
     add eax, ecx
     mov ebx, 85h
     mov ecx, 64h
     add eax, ebx
     add eax, ecx
call DumpRegs
exit
main ENDP
END main
```