

LAB 01

COMPUTER ORGANIZATION AND ASSEMBLY LANG(COAL)



Ammar Siddiqui

STUDENT NAME

20K-0177

ROLL NO

E
SEC

11/09/2021

SIGNATURE & DATE

MARKS AWARDED: _____

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

Prepared by: Amin Sadiq

Version: 1.0

Date:

Lab Session 01: CONFIGURATION OF VS 2019

Objectives:

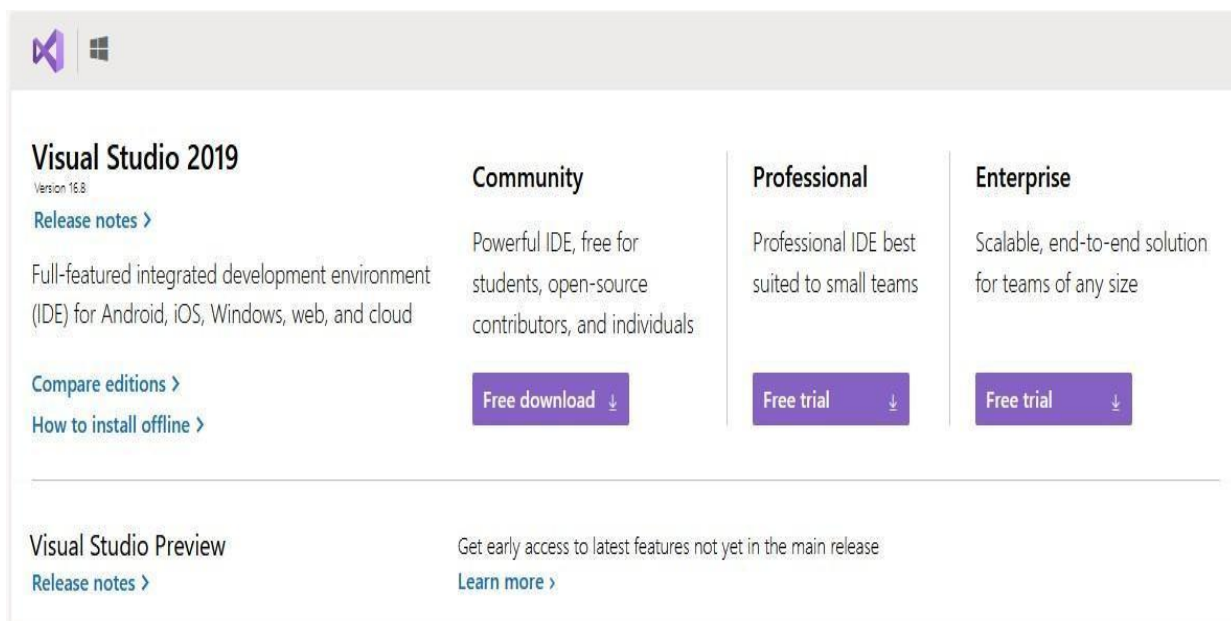
- Introduction to Visual Studio 2019
- An introduction to Assembly Language
- Understanding the Visual Studio 2019
- Configuring Visual Studio 2019 to activate MASM assembler
- Running a test program

SECTION 1: INTRODUCTION TO VISUAL STUDIO 2019

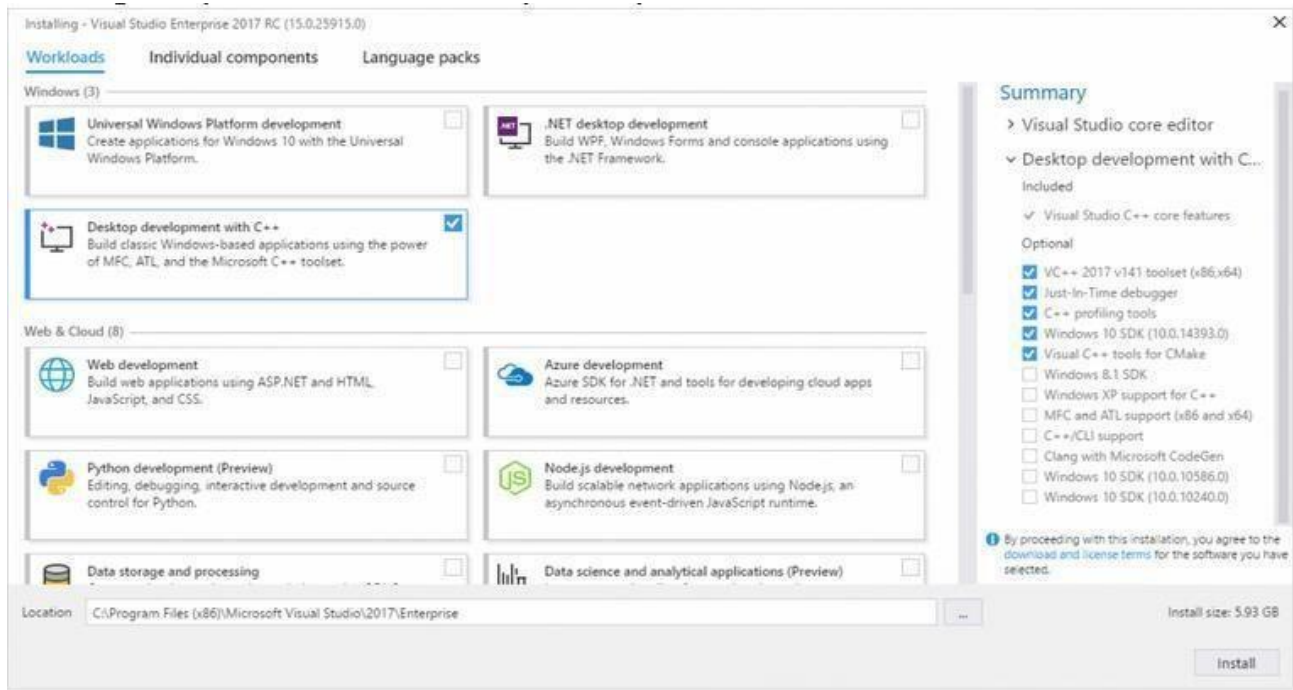
Visual Studio (for all its versions) is an integrated development environment (IDE) from Microsoft. It is a leading tool to develop computer programs, as well as web sites, web applications and web services. For this course, we will use Visual Studio version 2019 to develop programs in Assembly Language. We could, however, use a stand-alone assembler like NASM or MASM to code in Assembly Language.

INSTALLATION PROCESS

Go to this link <https://visualstudio.microsoft.com/downloads/> and select VS 2019 Download for community version



Run that downloaded setup on your system and when it's complete, you have to download and install **Desktop Development with C++**. When it's done you are ready to go.



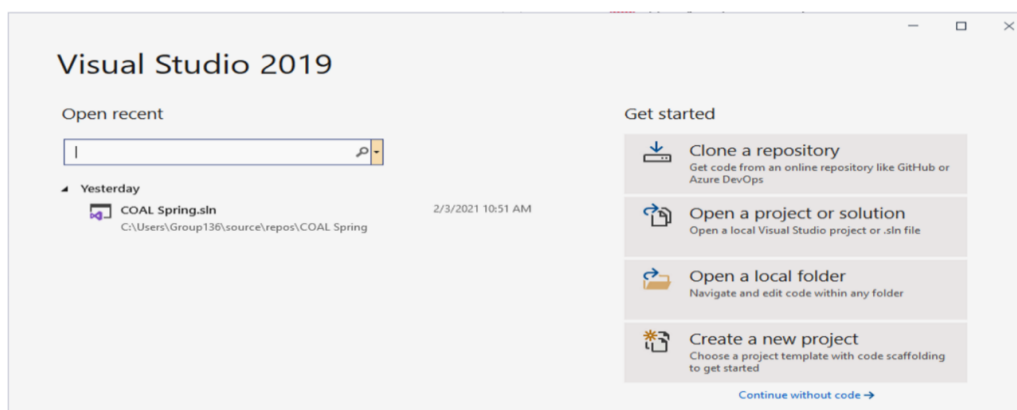
CONFIGURATION VS2019 FOR ASSEMBLY LANGUAGE

Click here to download Irvine library from this

link: www.asmirvine.com/gettingStartedVS2019/Irvine.zip

Once you have downloaded the required Irvine library, install it in your computer and verify that a folder named Irvine has been created in your C:\ drive. Now, follow these steps to configure Visual Studio 2019:

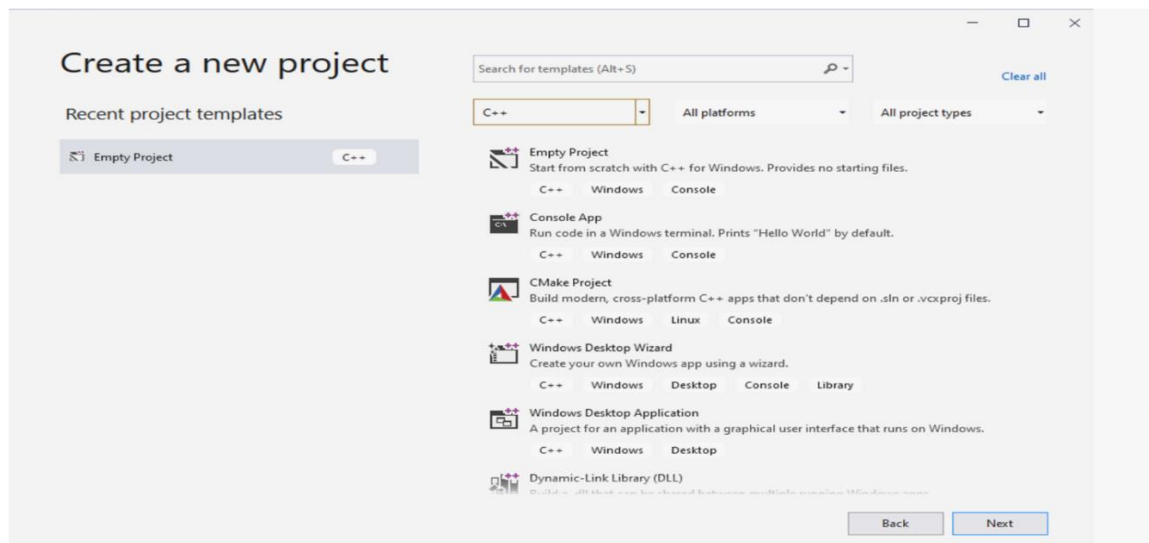
1. Start Microsoft Visual Studio 2019. If you are running it for the first time then this would be the screen you may see



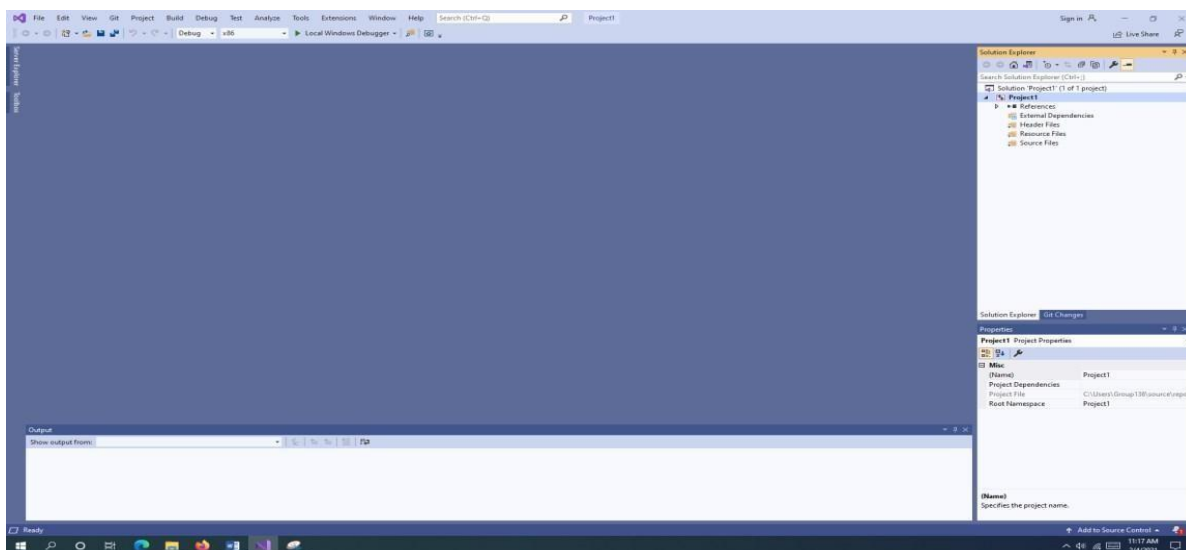
Select
Create a
new project.

2. From languages select C++, and then create an empty project

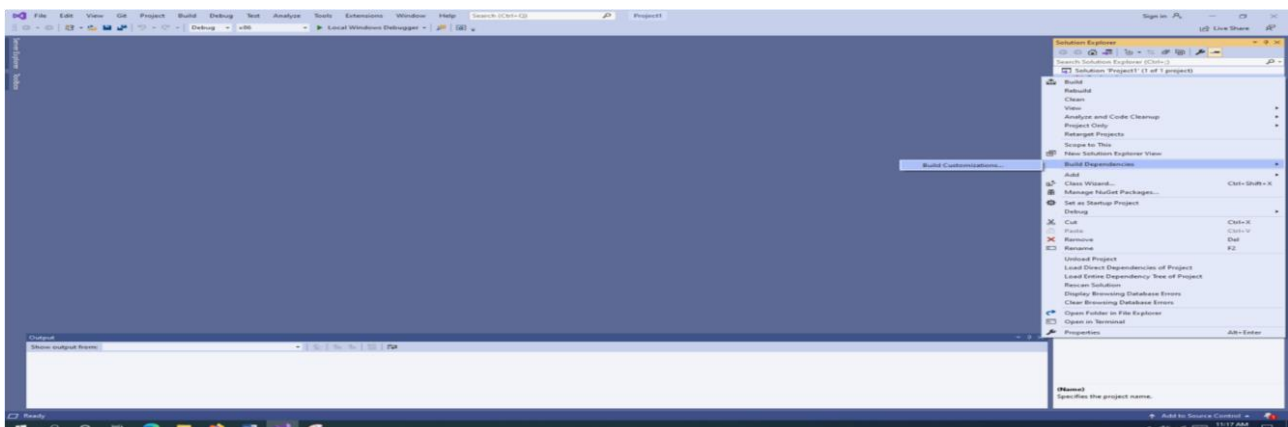




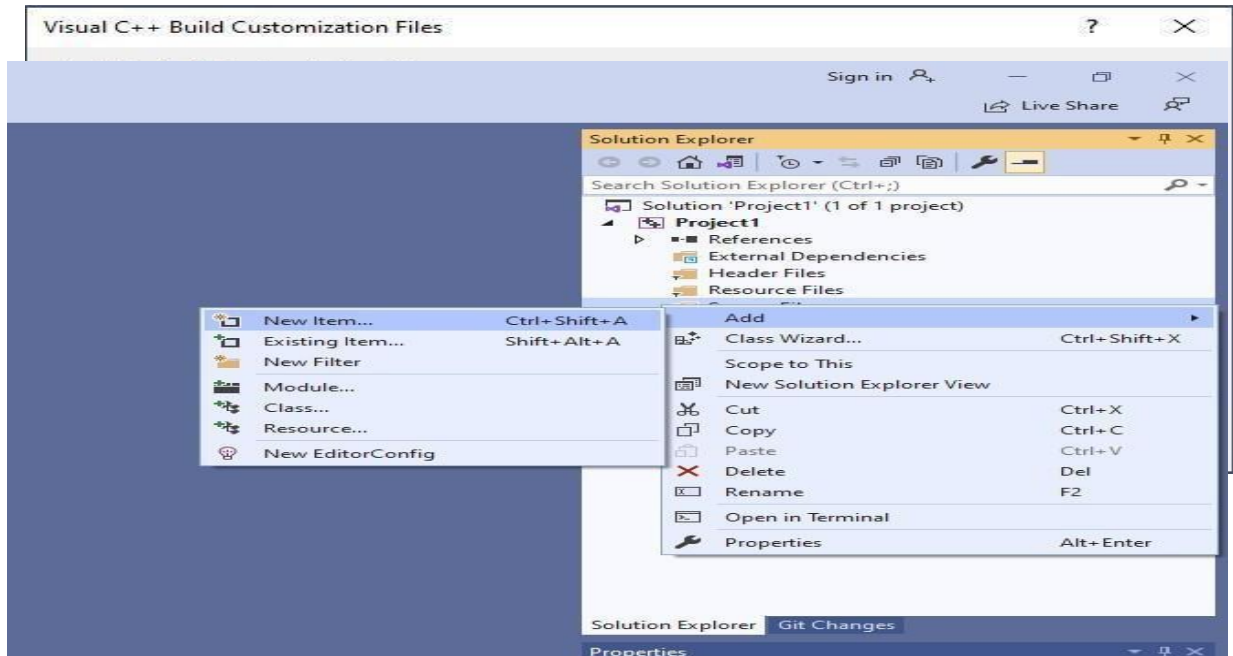
3. Once your new project is created, press **Ctrl+Alt+L** to open **Solution Explorer**. In the solution explorer window, you would see your project's file hierarchy.



Now right click on your project. Go to **Build Dependencies** and then select **Build Customization**

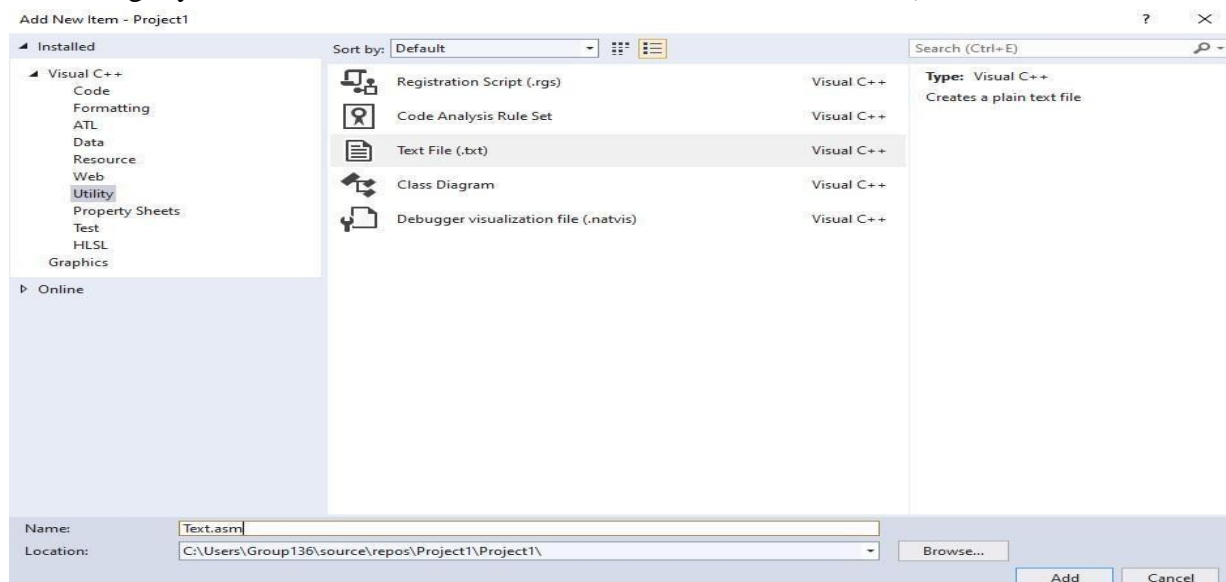


Tick the **masm** checkbox & select **OK**.

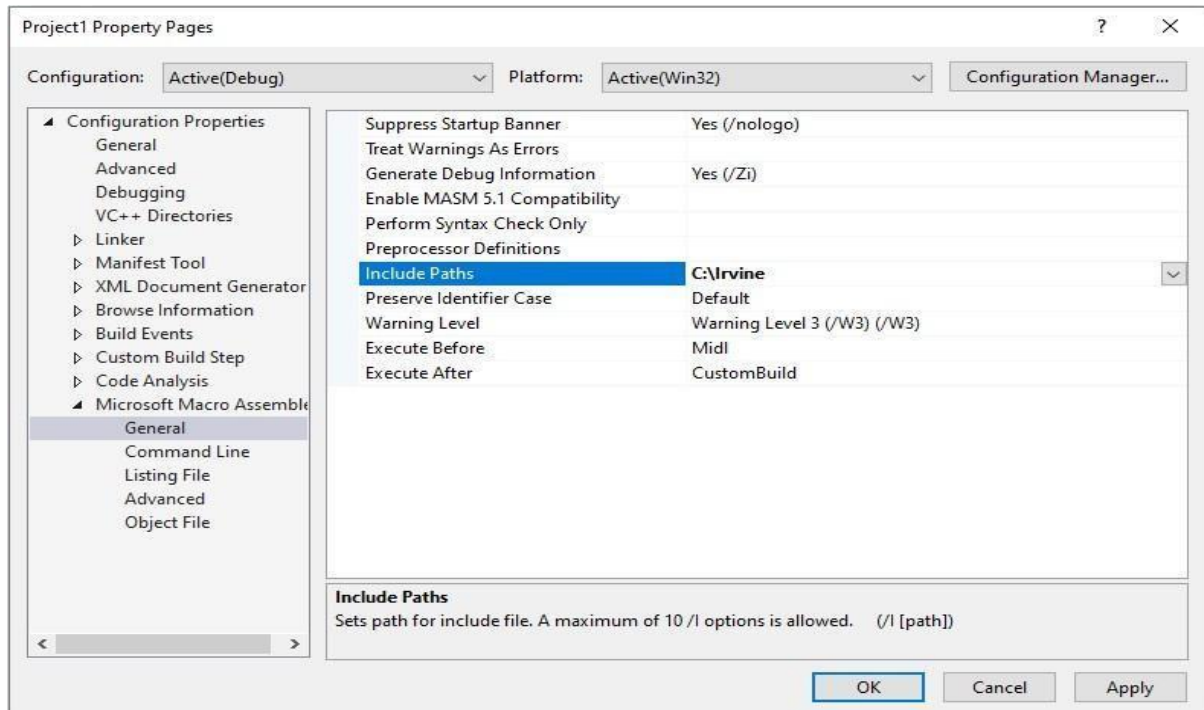


4. Right-click on **Source Files** in solution explorer & select **Add > New Item**.

Now go to **Utility > Text File** to add a new file, but we do not want to add .txt file, instead we want to add a .asm file. So, rename your new text file as Test.asm (we can choose any other name e.g. xyz.asm but for this tutorial we will use the name Test.asm).

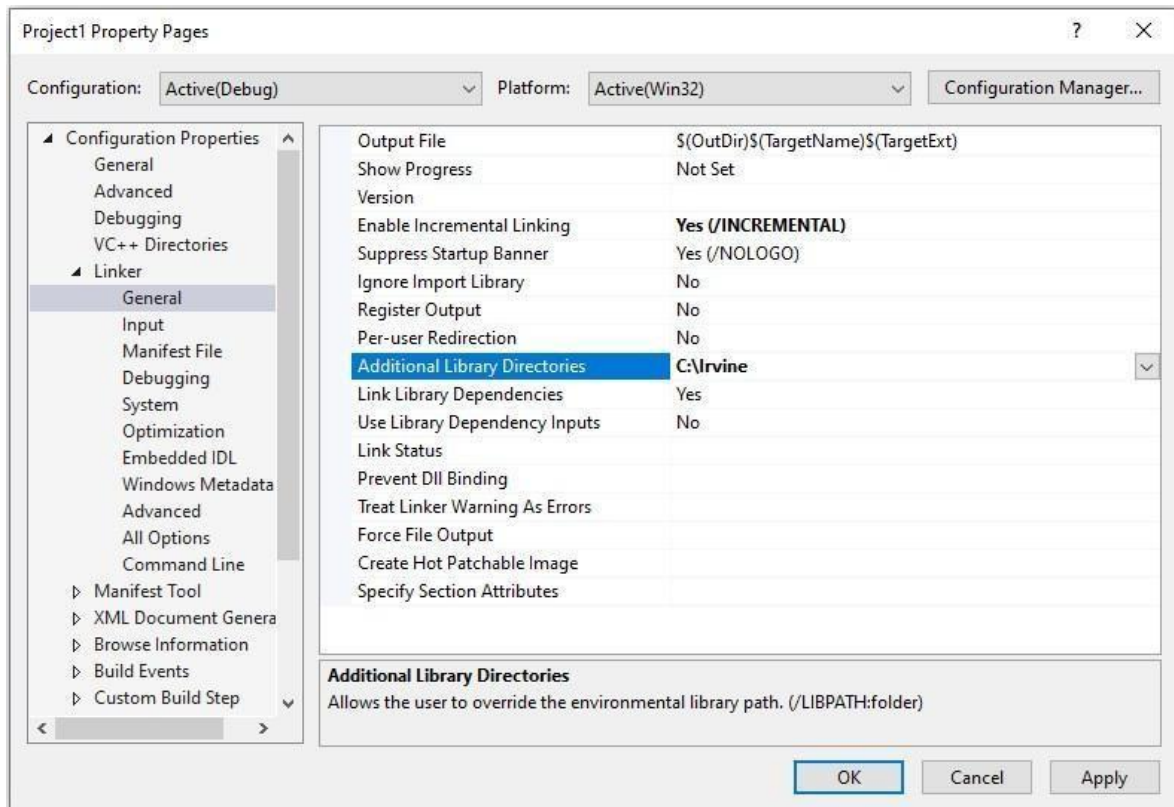


5. Now right-click your project again and click **Properties**. Now click the tiny arrow marker besides **Configuration Properties** to expand it. Now click **Microsoft Macro Assembler** and expand it.

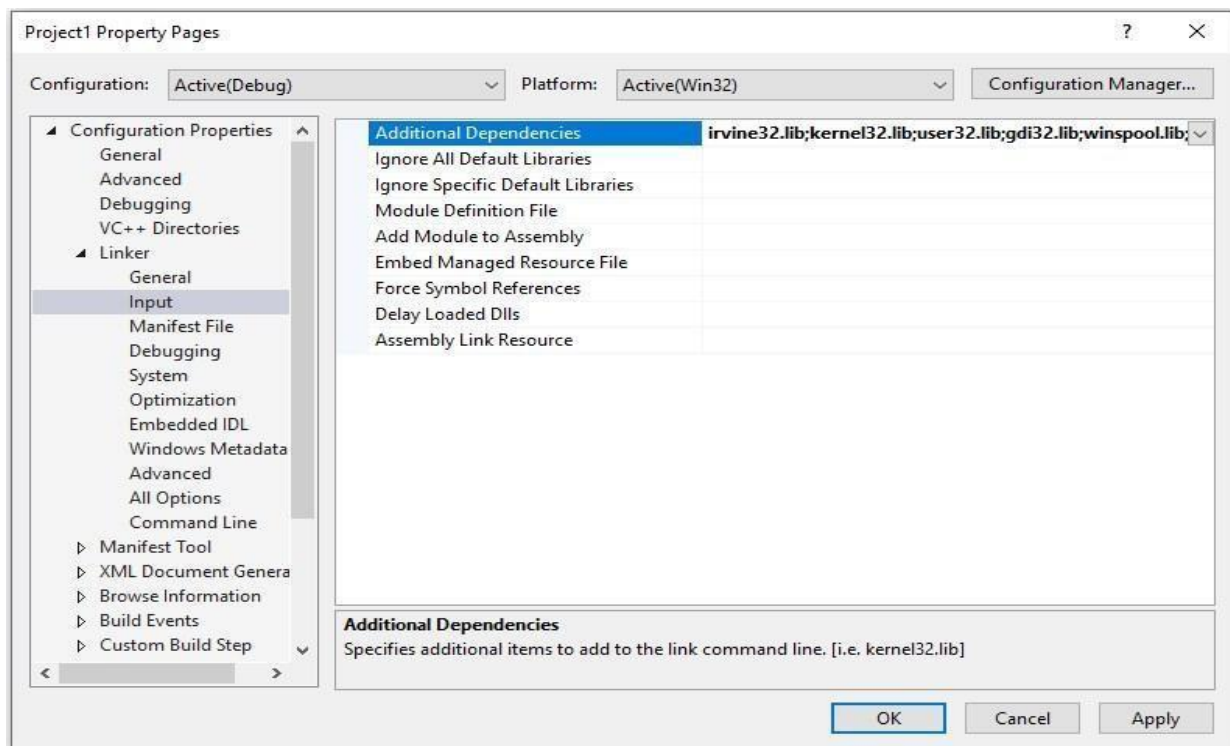


6. Now click **General** entry under Microsoft Macro Assembler and then set the value of **Include Paths** as **C:\Irvine**. The menu should now look like this.

7. Click **Linker** tab to expand it. Select **General** and set the value of **Additional Library Directories** to **C:\Irvine**



8. Click **Input**, select **Additional Dependencies**. You will see a list of different .lib file names written there, do not alter any of those. Write **irvine32.lib**; at the start of the list like this.



Our Visual Studio 2019 configuration for Assembly Language is complete. We can now write a sample program and run it to test our project. Open Test.asm from the solution explorer by double-clicking it. The Test.asm file will contain all the code that we write in our program. Go on and copy the following code onto your Test.asm file.

TITLE My First Program (Test.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 (Test.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 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 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 and ten mark the end of the main procedure.

SECTION 2: DEBUGGING OUR PROGRAM

We have seen how to configure Visual Studio 2019 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:

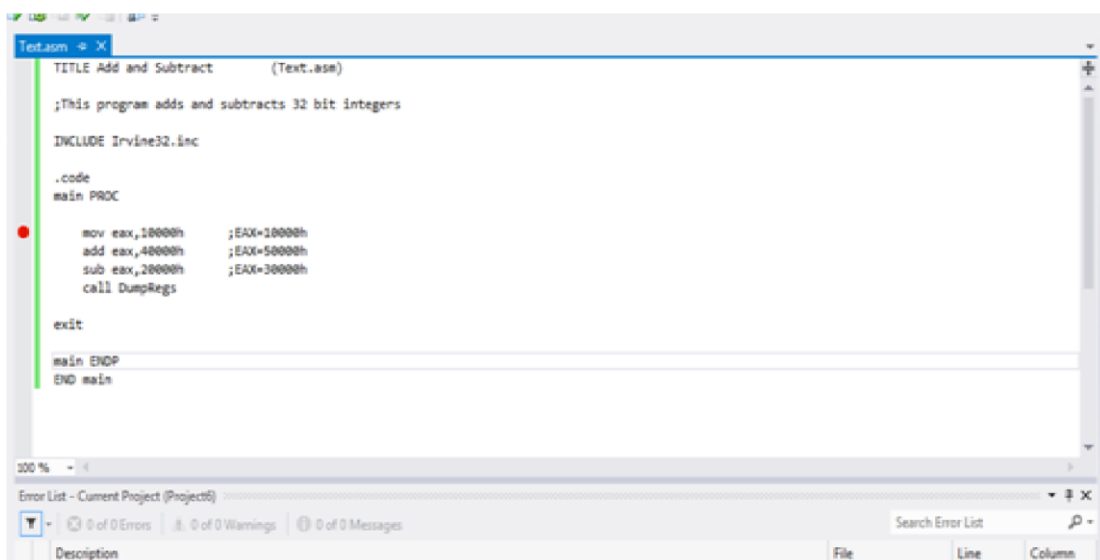


- Step Into (By pressing F11)
- Step Over (By pressing F10)
- Step Out (Shift+F11)

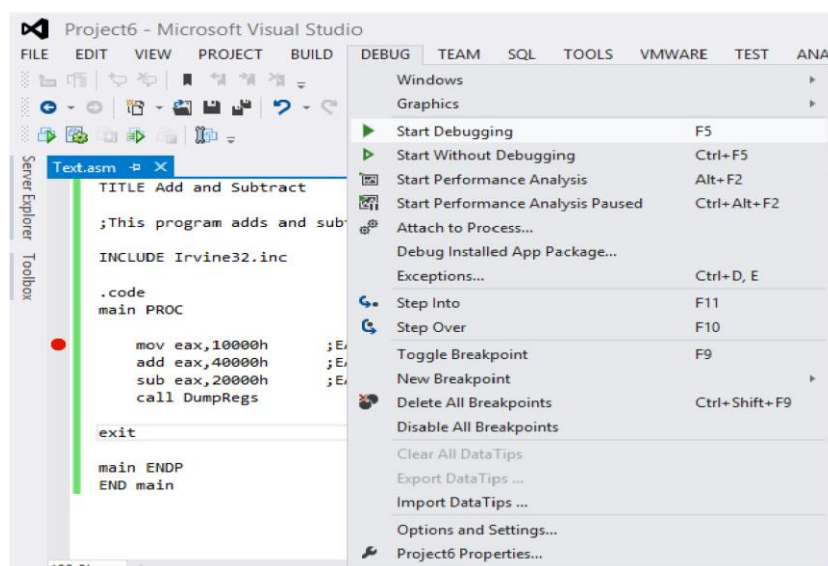
SINGLE STEPPING

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

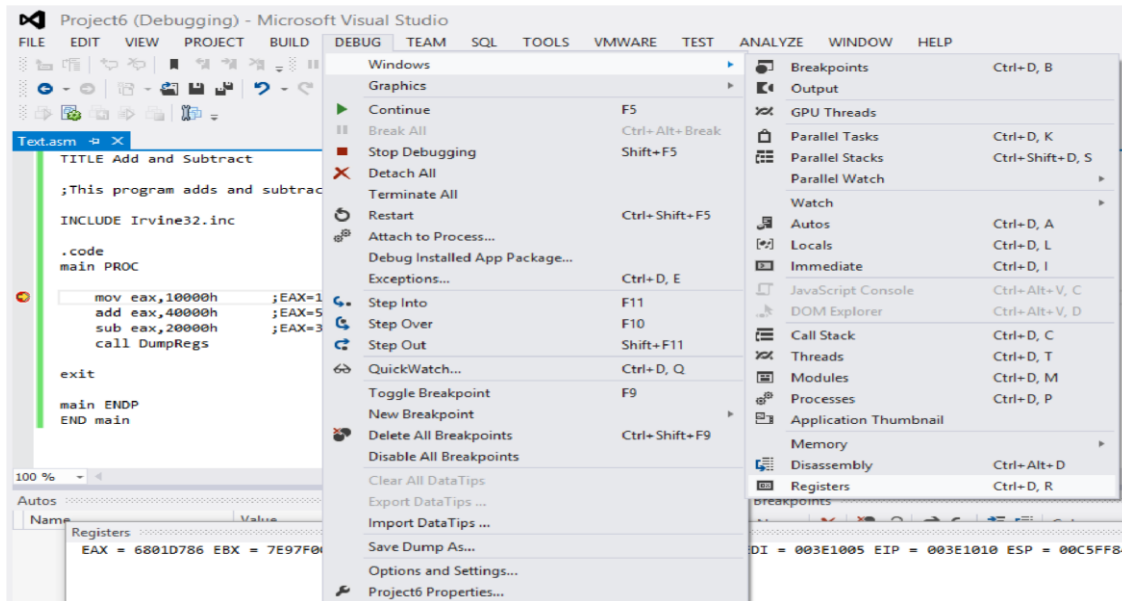
1. Right-click on line 6 to insert a breakpoint.



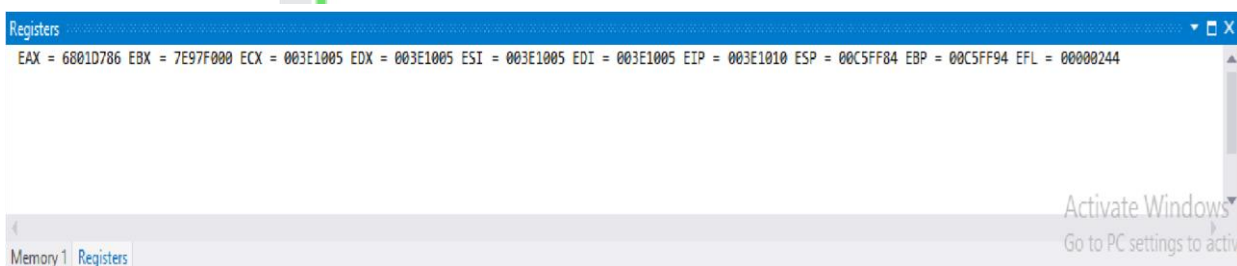
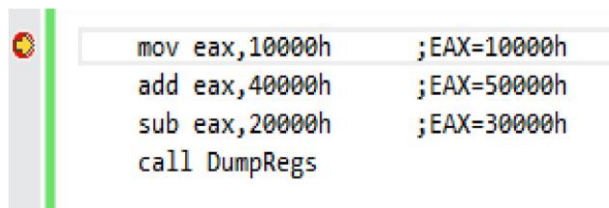
2. Click on **Debug** tab from the toolbar, select **Start Debugging** OR press **F10** to start stepping over the code.



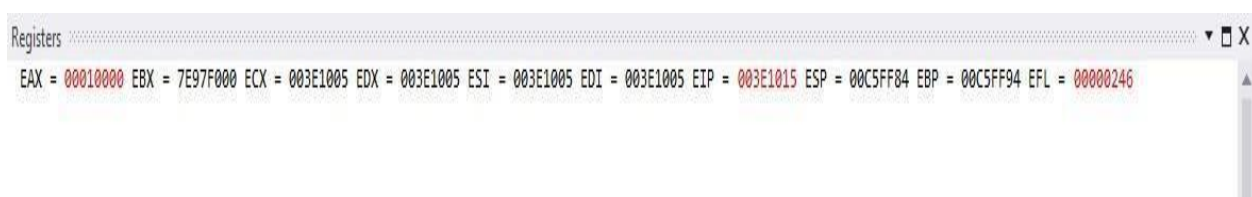
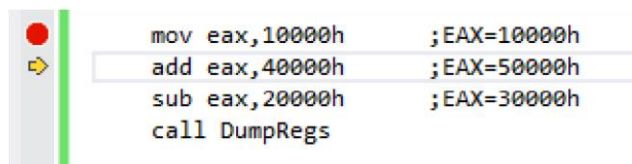
3. Click on **Debug** tab than select Windows after that open menu and select **Registers** option.



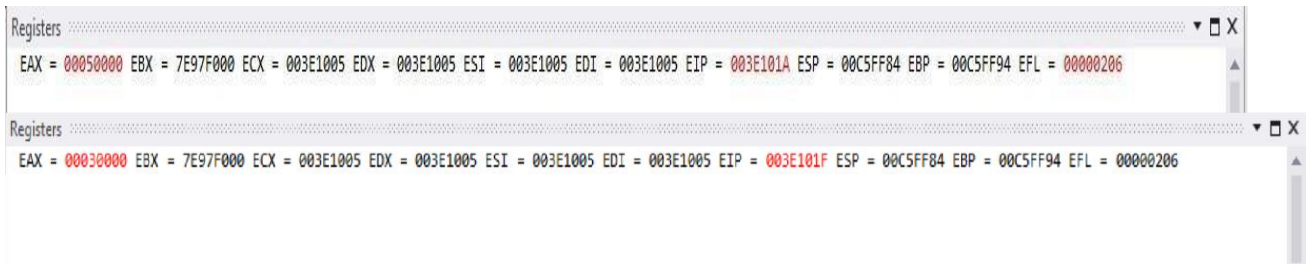
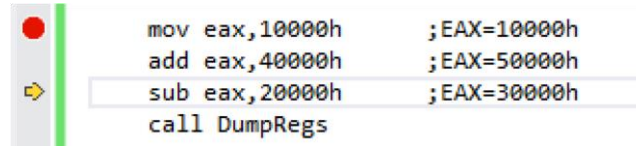
4. Breakpoint set on 1st instruction



Press **F10** again to execute next line.



Again press **F10** key for next instruction execution.



Press **F10** again, the program will not terminate after executing the current instruction and as soon as it reaches the line with a call to **DumpRegs**

SECTION 2: EXERCISE

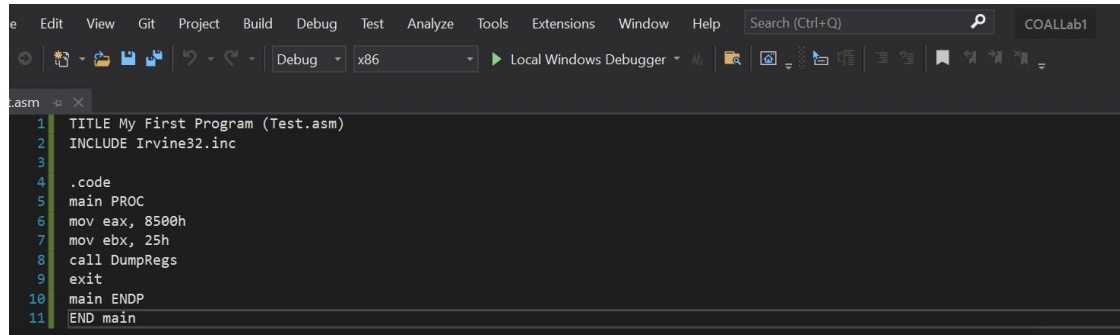
1. Install Visual Studio 2019 & create a new Visual C++ project for Assembly Language.
2. Configure the project using the steps show in this lab.
3. Run a test program in console window by changing the value of EAX in line 6 to 8500h.
4. Debug the below program and note down the values of all the registers after the execution of each line.

```
TITLE My First Program (Test.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
```

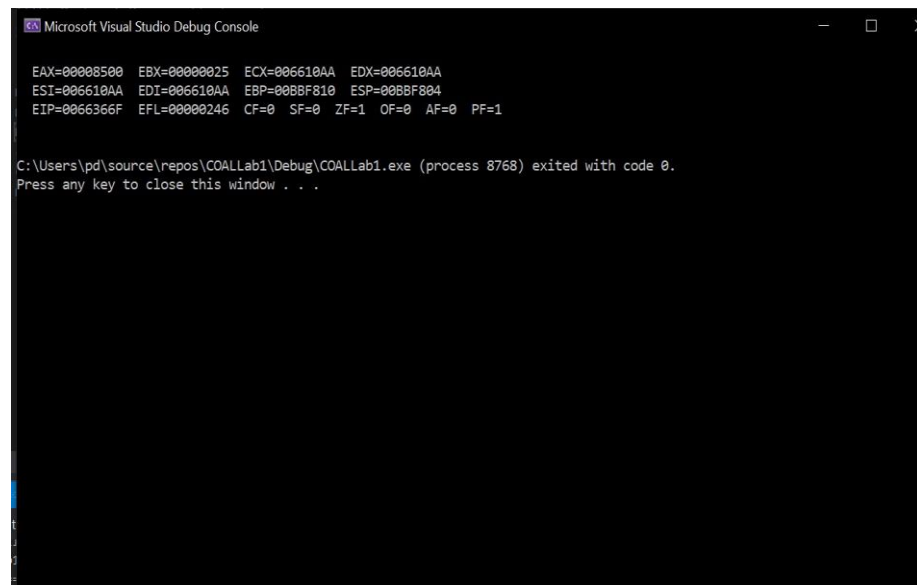
```
    call DumpRegs
    exit
main ENDP
END main
```

TASK 1:



```
1 TITLE My First Program (Test.asm)
2 INCLUDE Irvine32.inc
3
4 .code
5 main PROC
6 mov eax, 8500h
7 mov ebx, 25h
8 call DumpRegs
9 exit
10 main ENDP
11 END main
```

OUTPUT:

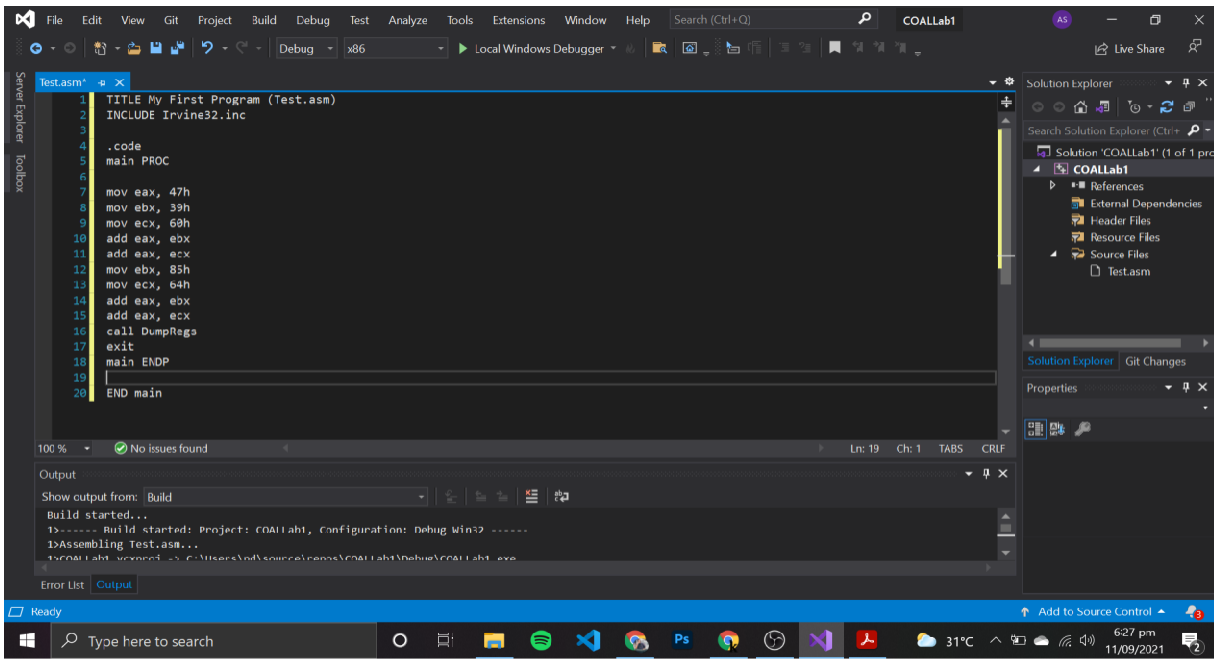


```
Microsoft Visual Studio Debug Console

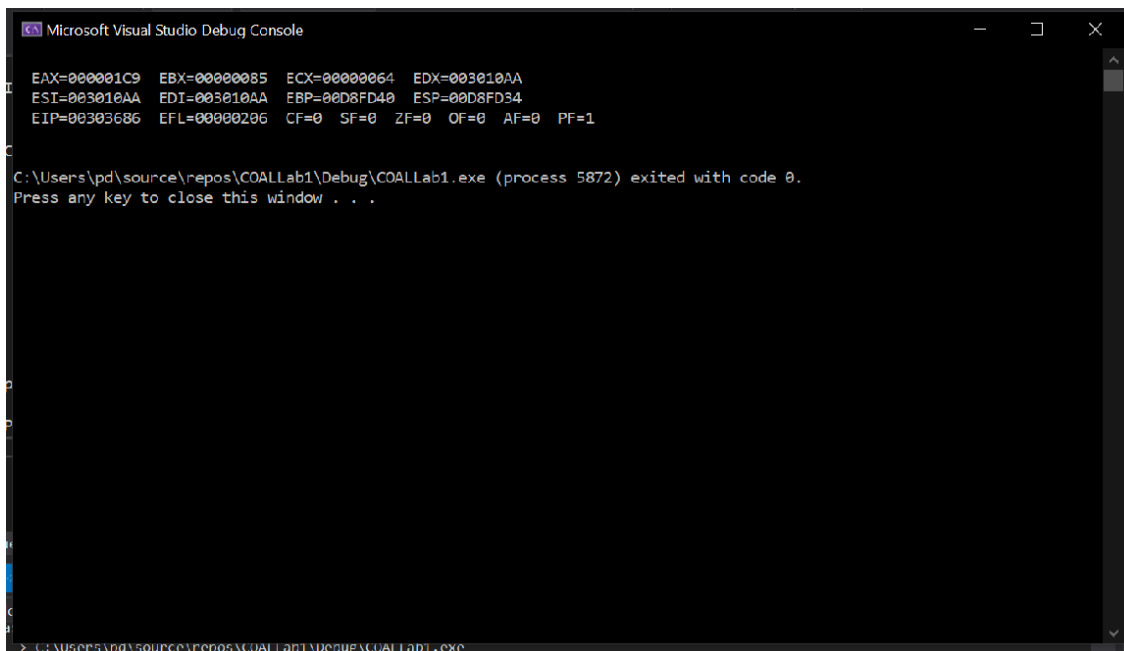
EAX=00008500 EBX=00000025 ECX=006610AA EDX=006610AA
ESI=006610AA EDI=006610AA EBP=008BF810 ESP=008BF804
EIP=0066366F EFL=00000246 CF=0 SF=0 ZF=1 OF=0 AF=0 PF=1

C:\Users\pd\source\repos\COALLab1\Debug\COALLab1.exe (process 8768) exited with code 0.
Press any key to close this window . . .
```

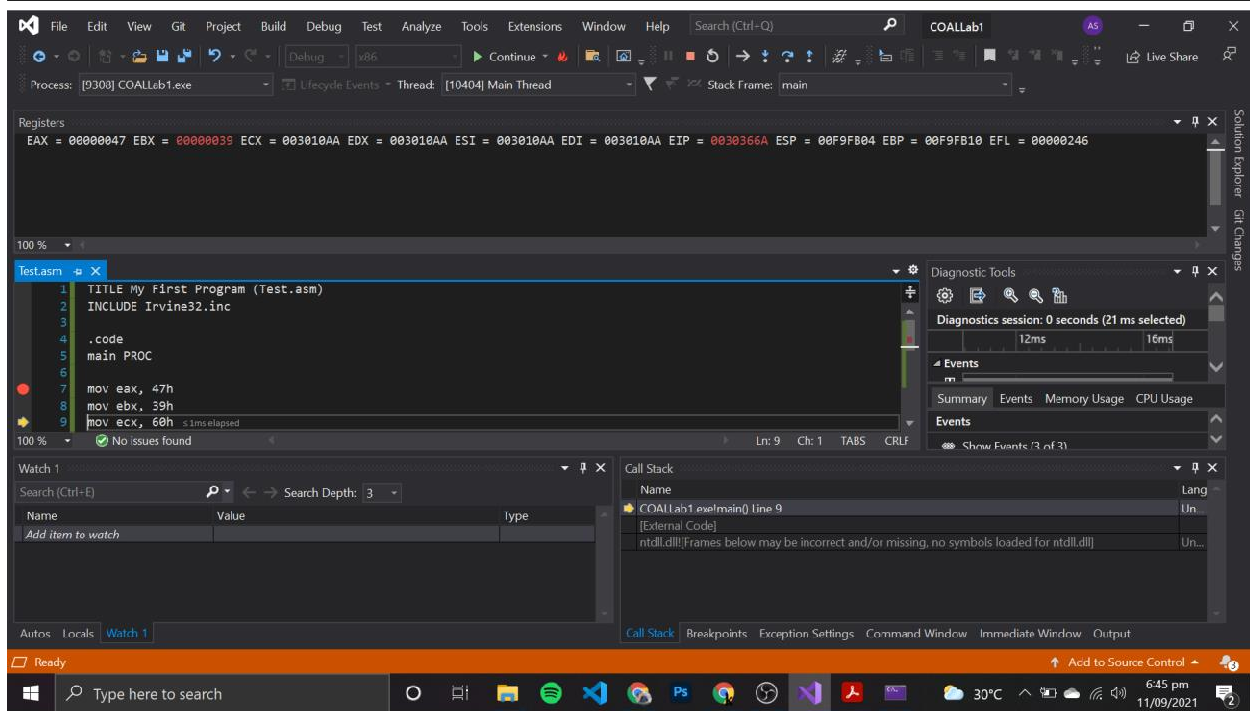
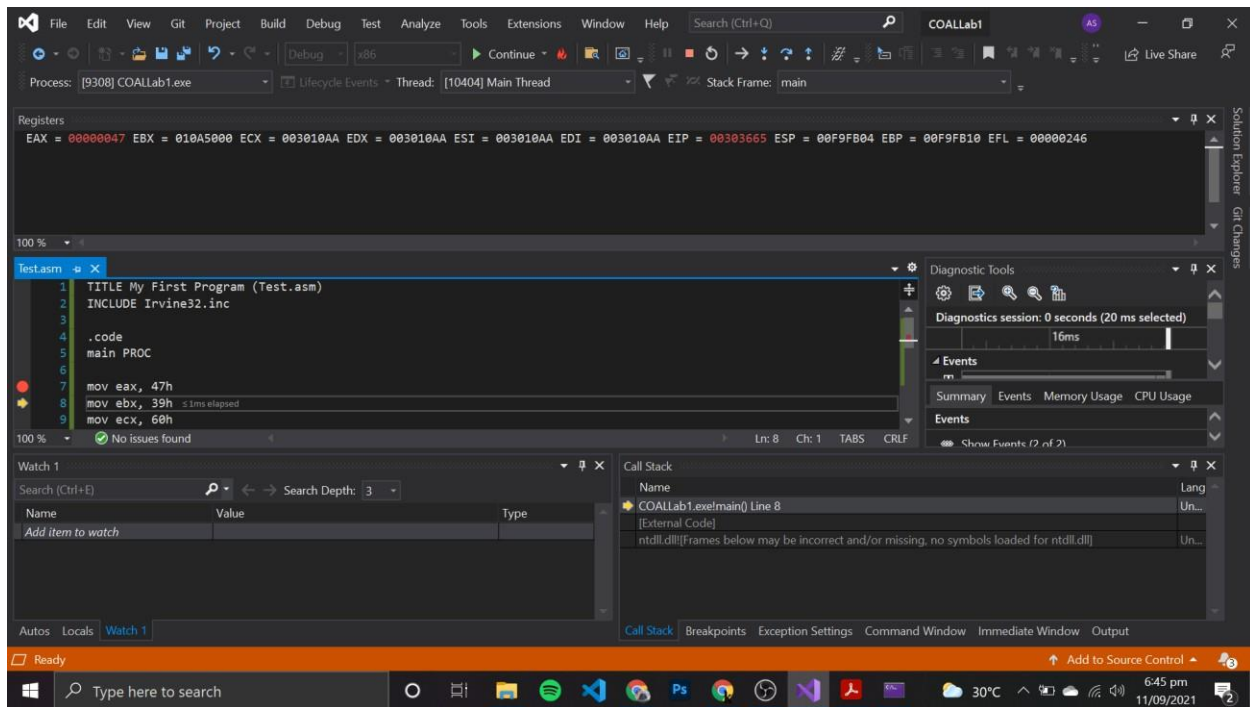
TASK 2:



OUTPUT:



DEBUGGING:



Visual Studio Code interface showing the debugger for the process [9308] COALLab1.exe. The thread is [10404] Main Thread. The registers display the following values: EAX = 00000047, EBX = 00000039, ECX = 00000060, EDX = 003010AA, ESI = 003010AA, EDI = 003010AA, EIP = 0030366F, ESP = 00F9FB04, EBP = 00F9FB10, EFL = 00000246.

The assembly code in the test.asm file is as follows:

```
1 INCLUDE Irvine32.inc
2
3
4 .code
5 main PROC
6
7 mov eax, 47h
8 mov ebx, 39h
9 mov ecx, 60h
10 add eax, ebx
```

The Watch window shows the following variables:

Name	Value	Type
Add item to watch		

The Call Stack shows the current frame: COALLab1.exe/main() Line 10. The diagnostic tools panel shows a diagnostics session of 0 seconds (21 ms selected).

Windows taskbar: Type here to search, 30°C, 6:45 pm 11/09/2021.

Visual Studio Code interface showing the debugger for the process [9308] COALLab1.exe. The thread is [10404] Main Thread. The registers display the following values: EAX = 00000060, EBX = 00000039, ECX = 00000060, EDX = 003010AA, ESI = 003010AA, EDI = 003010AA, EIP = 00303671, ESP = 00F9FB04, EBP = 00F9FB10, EFL = 00000212.

The assembly code in the test.asm file is as follows:

```
7 mov eax, 47h
8 mov ebx, 39h
9 mov ecx, 60h
10 add eax, ebx
11 add eax, ecx
12 mov ebx, 85h
13 mov ecx, 64h
14 add eax, ebx
15 add eax, ecx
```

The Watch window shows the following variables:

Name	Value	Type
Add item to watch		

The Call Stack shows the current frame: COALLab1.exe/main() Line 11. The diagnostic tools panel shows a diagnostics session of 0 seconds (21 ms selected).

Windows taskbar: Type here to search, 30°C, 6:45 pm 11/09/2021.

Visual Studio Code interface showing a debug session for `COALLab1.exe`. The process is running on the `[10404] Main Thread`. The registers window shows the following values:

EAX = 000000E0 EBX = 00000039 ECX = 00000060 EDX = 003010AA ESI = 003010AA EDI = 003010AA EIP = 00303673 ESP = 00F9FB04 EBP = 00F9FB10 EFL = 00000202

The assembly window shows the following code:

```
7 mov eax, 47h
8 mov ebx, 39h
9 mov ecx, 60h
10 add eax, ebx
11 add eax, ecx
12 mov ebx, 85h
13 mov ecx, 64h
14 add eax, ebx
15 add eax, ecx
```

The Watch window shows the following variables:

Name	Value	Type
Add item to watch		

The Call Stack window shows the following stack frames:

Name	Lang
COALLab1.exe/main() Line 12	Un...
[External Code]	Un...
ntdll.dll	Un...

The Diagnostic Tools window shows the following events:

Summary	Events	Memory Usage	CPU Usage
Events			

Visual Studio Code interface showing a debug session for `COALLab1.exe`. The process is running on the `[10404] Main Thread`. The registers window shows the following values:

EAX = 000000E0 EBX = 00000085 ECX = 00000060 EDX = 003010AA ESI = 003010AA EDI = 003010AA EIP = 00303678 ESP = 00F9FB04 EBP = 00F9FB10 EFL = 00000202

The assembly window shows the following code:

```
7 mov eax, 47h
8 mov ebx, 39h
9 mov ecx, 60h
10 add eax, ebx
11 add eax, ecx
12 mov ebx, 85h
13 mov ecx, 64h
14 add eax, ebx
15 add eax, ecx
```

The Watch window shows the following variables:

Name	Value	Type
Add item to watch		

The Call Stack window shows the following stack frames:

Name	Lang
COALLab1.exe/main() Line 13	Un...
[External Code]	Un...
ntdll.dll	Un...

The Diagnostic Tools window shows the following events:

Summary	Events	Memory Usage	CPU Usage
Events			

Visual Studio Code interface showing a debug session for `COALLab1.exe`. The process is running on the `[10404] Main Thread`. The registers display values for `EAX`, `EBX`, `ECX`, `EDX`, `ESI`, `EDI`, `EIP`, `ESP`, `EBP`, and `EFL`.

The assembly code in the `test.asm` file is as follows:

```
7 mov eax, 47h
8 mov ebx, 39h
9 mov ecx, 60h
10 add eax, ebx
11 add eax, ecx
12 mov ebx, 85h
13 mov ecx, 64h
14 add eax, ebx
15 add eax, ecx
```

The `Watch` window shows a search for `Watch 1` with a search depth of 3. The `Call Stack` window shows the current frame as `COALLab1.exe/main() Line 14`. The `Diagnostics` window shows a session of 0 seconds (22 ms selected) with 18ms of events. A `Screenshot saved` notification is visible in the bottom right corner.

Visual Studio Code interface showing a debug session for `COALLab1.exe`. The process is running on the `[10404] Main Thread`. The registers display values for `EAX`, `EBX`, `ECX`, `EDX`, `ESI`, `EDI`, `EIP`, `ESP`, `EBP`, and `EFL`.

The assembly code in the `test.asm` file is as follows:

```
7 mov eax, 47h
8 mov ebx, 39h
9 mov ecx, 60h
10 add eax, ebx
11 add eax, ecx
12 mov ebx, 85h
13 mov ecx, 64h
14 add eax, ebx
15 add eax, ecx
```

The `Watch` window shows a search for `Watch 1` with a search depth of 3. The `Call Stack` window shows the current frame as `COALLab1.exe/main() Line 15`. The `Diagnostics` window shows a session of 0 seconds (22 ms selected) with 18ms of events. A `Screenshot saved` notification is visible in the bottom right corner.

Visual Studio Code interface showing the debugger for COALLab1.exe. The process is [9308] COALLab1.exe, Thread: [10404] Main Thread. The registers display EAX = 000001C9, EBX = 00000085, ECX = 00000064, EDX = 003010AA, ESI = 003010AA, EDI = 003010AA, EIP = 00303681, ESP = 00F9FB04, EBP = 00F9FB10, EFL = 00000206.

The assembly code in the Test.asm window shows the following instructions:

```
8 mov ebx, 39h
9 mov ecx, 60h
10 add eax, ebx
11 add eax, ecx
12 mov ebx, 85h
13 mov ecx, 64h
14 add eax, ebx
15 add eax, ecx
16 call DumpRegs
```

The Watch window shows a search for "Watch 1" with a search depth of 3. The Call Stack shows the current frame is COALLab1.exe/main() Line 16, [External Code].

The Diagnostic Tools window shows a diagnostics session of 0 seconds (22 ms selected). The Events window shows a summary of events.

The bottom status bar indicates the system is Ready.

Visual Studio Code interface showing the debugger for COALLab1.exe. The process is [9308] COALLab1.exe, Thread: [10404] Main Thread. The registers display EAX = 000001C9, EBX = 00000085, ECX = 00000064, EDX = 003010AA, ESI = 003010AA, EDI = 003010AA, EIP = 00303686, ESP = 00F9FB04, EBP = 00F9FB10, EFL = 00000206.

The assembly code in the Test.asm window shows the following instructions:

```
13 mov ecx, 64h
14 add eax, ebx
15 add eax, ecx
16 call DumpRegs
17 exit
18 main ENDP
19
20 END main
```

The Watch window shows a search for "Watch 1" with a search depth of 3. The Call Stack shows the current frame is COALLab1.exe/main() Line 17, [External Code].

The Diagnostic Tools window shows a diagnostics session of 0 seconds (24 ms selected). The Events window shows a summary of events.

The bottom status bar indicates the system is Ready.