

# Lab 01

## Objectives

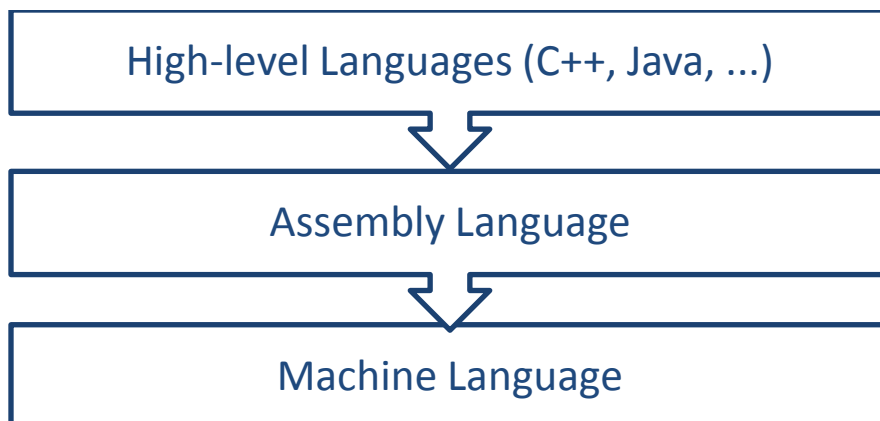
- An introduction to Assembly Language
- Understanding the Visual Studio 2010
- Configuring Visual Studio 2010 to activate MASM assembler
- Running a test program

## Section 1: Introduction

In this section we will give a brief introduction to what is Assembly Language and why is it so useful to study. Let us begin:

### 1.1 Assembly Language

Assembly Language is an intermediary or low-level language that provides direct access to computer hardware.



## 1.2 A simple example

Consider this C code fragment:

```
int Y;  
int X = (Y + 4) * 3;
```

Following is the equivalent translation to Assembly Language:

<b>mov eax,Y</b>	<i>; move Y to the EAX register</i>
<b>add eax,4</b>	<i>; add 4 to the EAX register</i>
<b>mov ebx,3</b>	<i>; move 3 to the EBX register</i>
<b>mul ebx</b>	<i>; multiply EAX by EBX</i>
<b>mov X,eax</b>	<i>; move EAX to X</i>

## Section 2: Setting up Visual Studio 2010

In this section, we will introduce Microsoft Visual Studio 2010 & configure it for Assembly Language programming.

### 2.1 Introduction to Visual Studio 2010

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 2010 to develop programs in Assembly Language. We could, however, use a stand-alone assembler like NASM or MASM to code in Assembly Language.

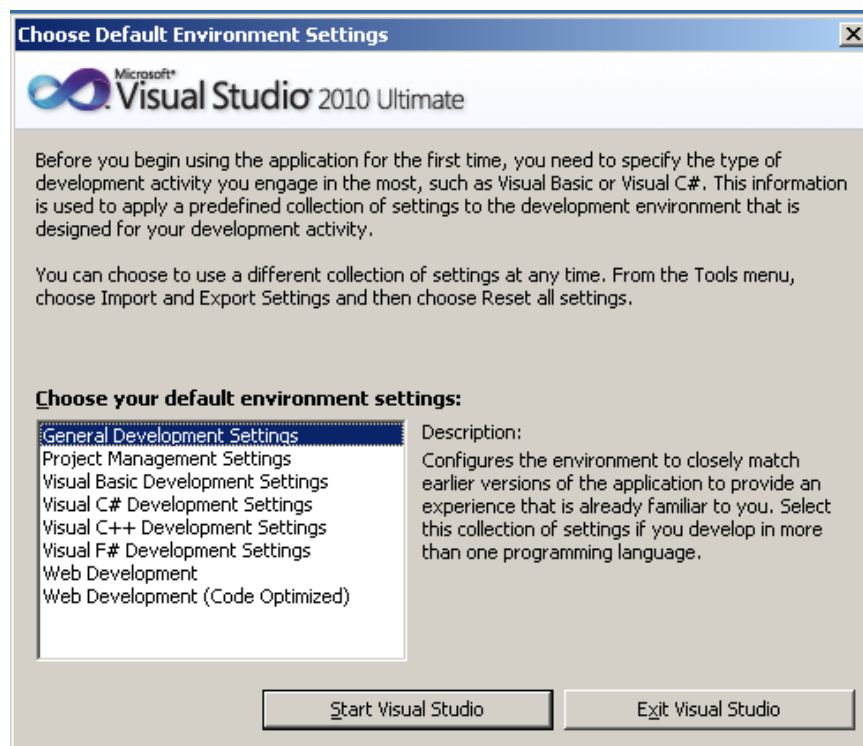
## 2.2 Configuring VS 2010 for Assembly Language

Copy & install the latest version (for 6<sup>th</sup>/7<sup>th</sup> edition) of Irvine's link library from \\netstorage or alternatively download to install these libraries from this link:

<http://kipirvine.com/asm/examples/index.htm>

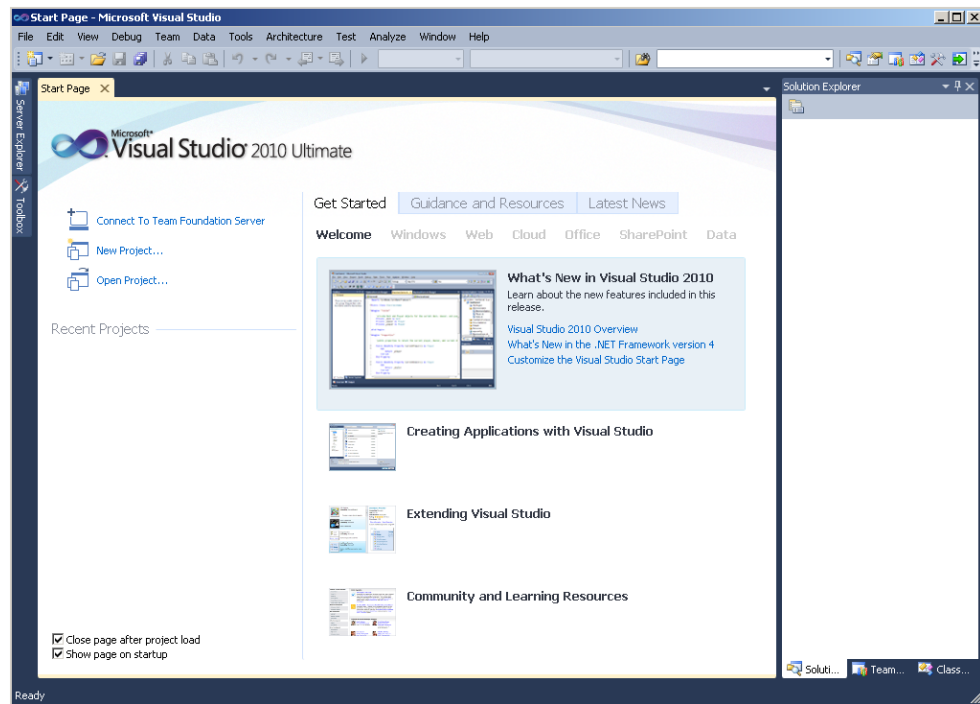
Once you have downloaded the required Irvine library, install it in your computer and verify that a folder named *Irvine* has been created in your C:\ drive. Now, follow these steps to configure Visual Studio 2010:

1. Start Microsoft Visual Studio 2010. If you are running it for the first time then this would be the screen you may see.



Select **General Development Settings** and press **Start Visual Studio**.

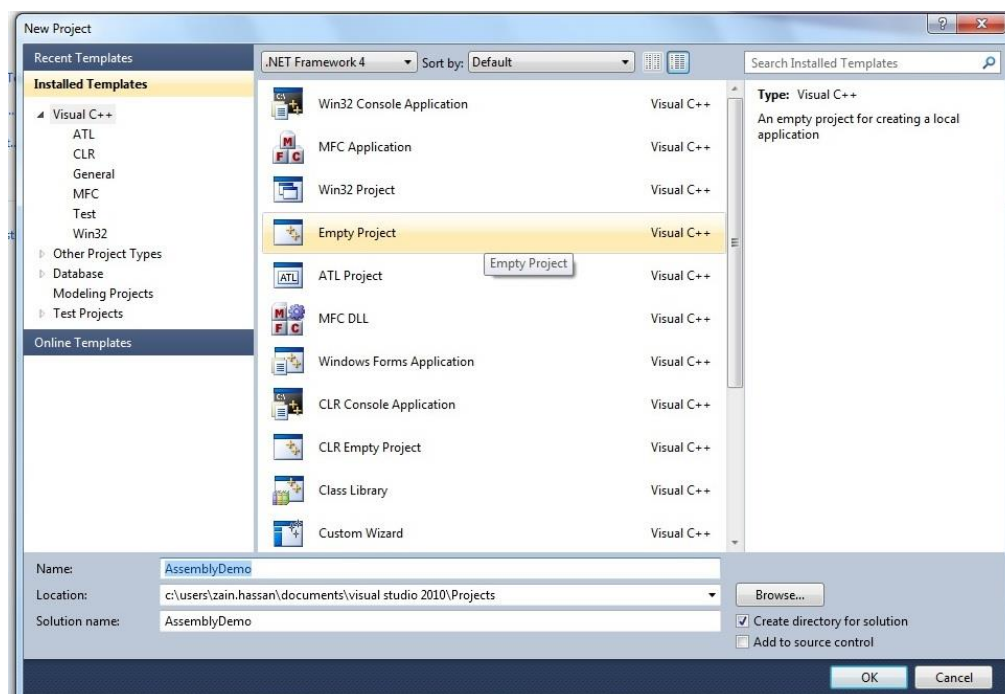
2. After startup, this would be the welcome screen.



Click on the **New Project** link.

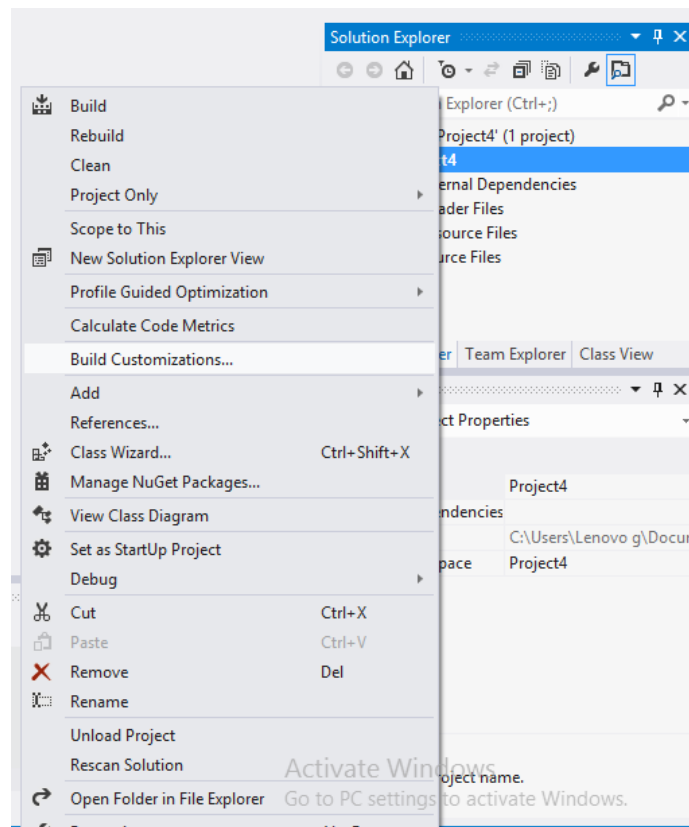
3. In the New Project dialog (shown in the image below), select **Other Languages**, select **Visual C++**, select **General**, and then select **Empty Project**.

Give a name to the project (assume it *AssemblyDemo* for this tutorial). Now click the **OK** button.

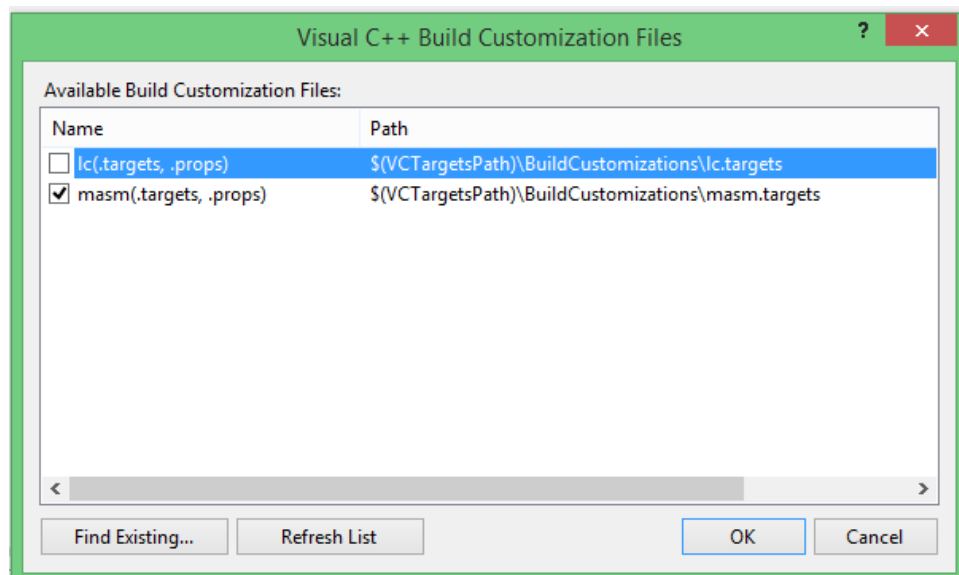


4. Once your new project is created, press **Ctrl+Shift+L** to open Solution Explorer. In the solution explorer window, you would see your project's file hierarchy.

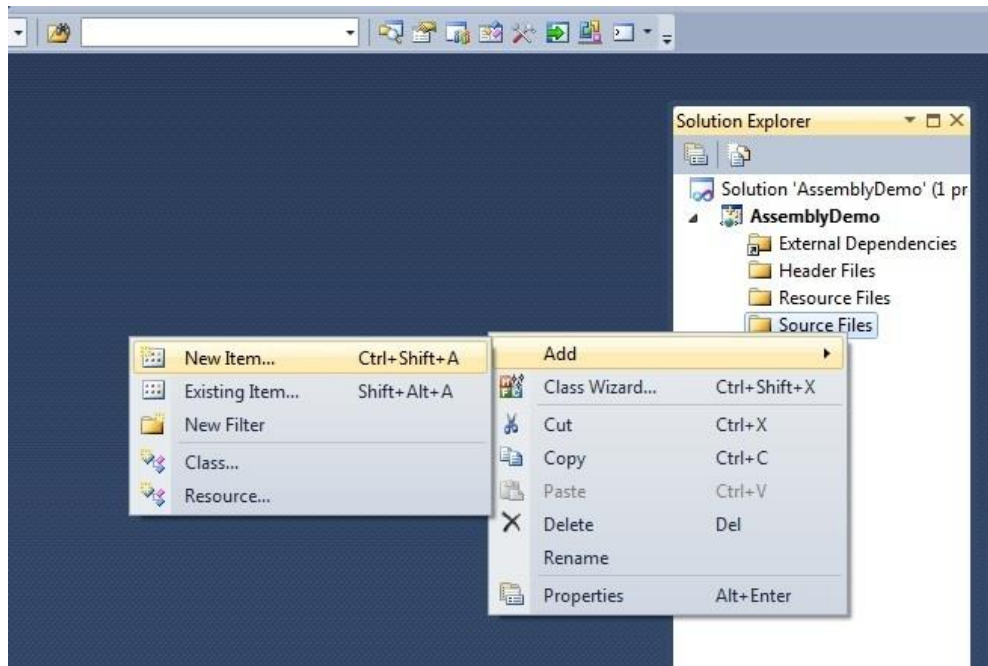
Now, right-click on your project in solution explorer and select **Build Customizations**.



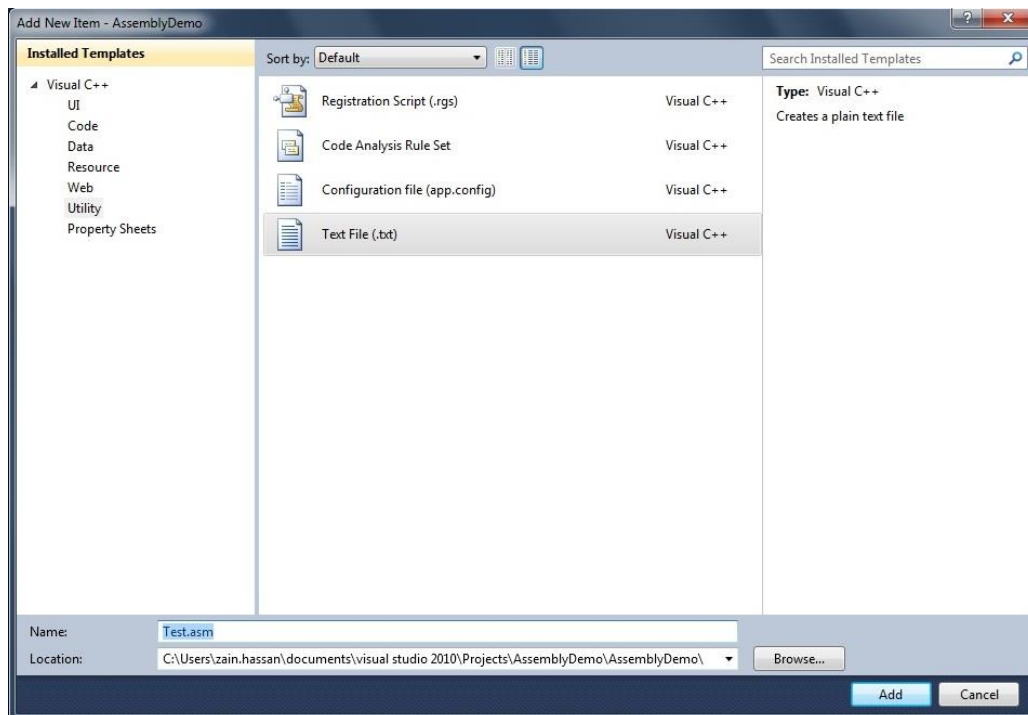
Tick the **masm** checkbox & select **OK**.



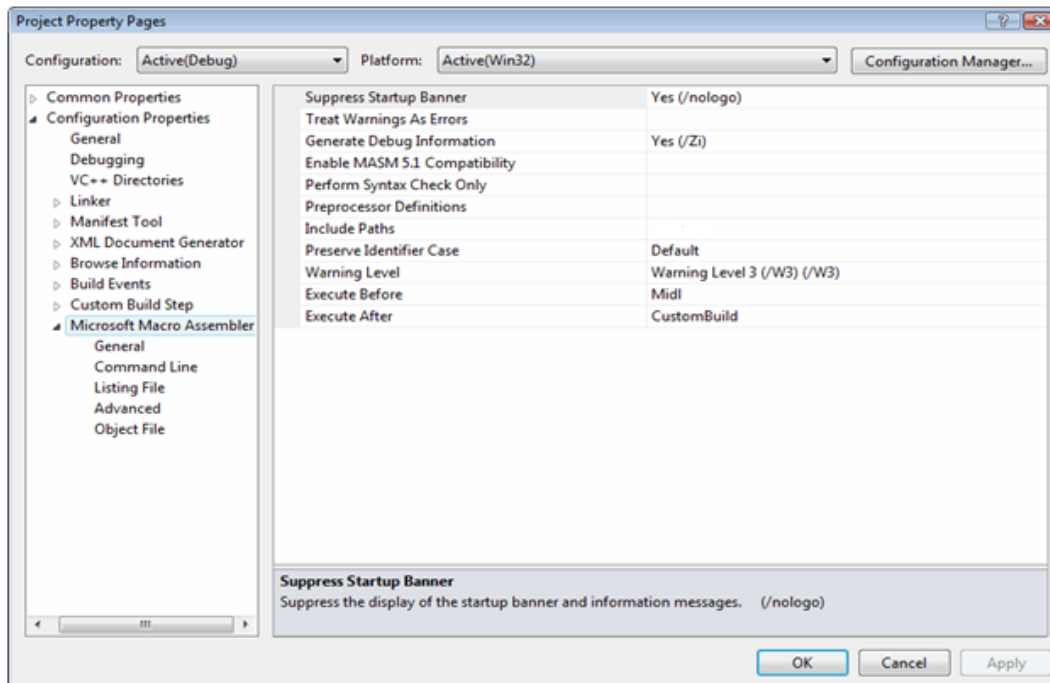
5. Right-click on **Source Files** in solution explorer & select **Add > New Item**.



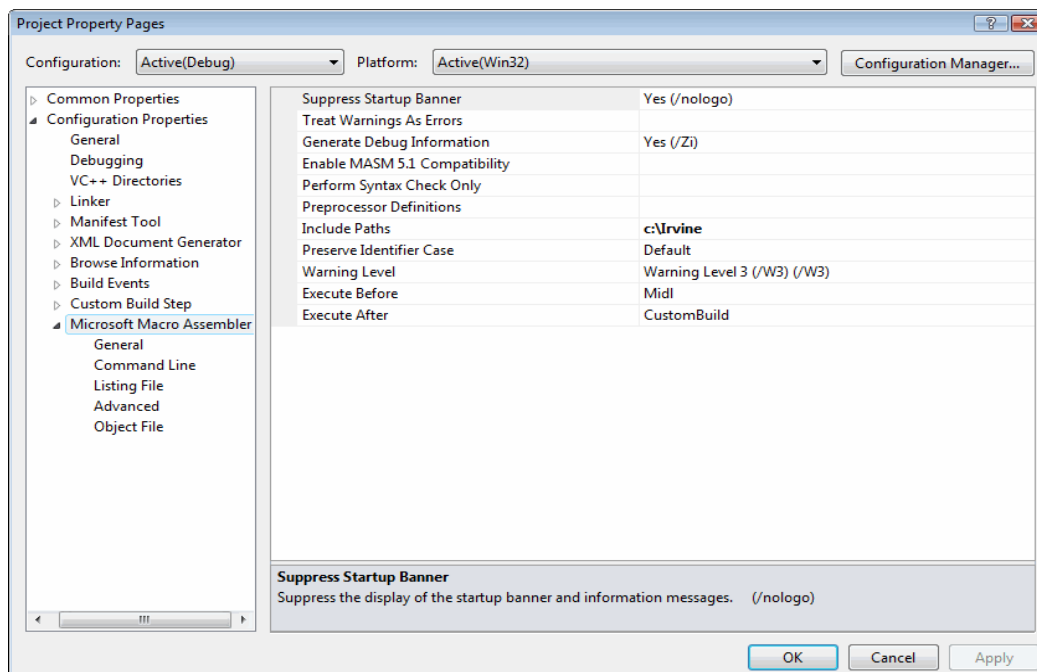
Now go to **Utility > Text File** to add a new file, but we do not want to add .txt file, instead we want to add a .asm file. So, rename your new text file as **Test.asm** (we can choose any other name e.g. *xyz.asm* but for this tutorial we will use the name *Test.asm*).



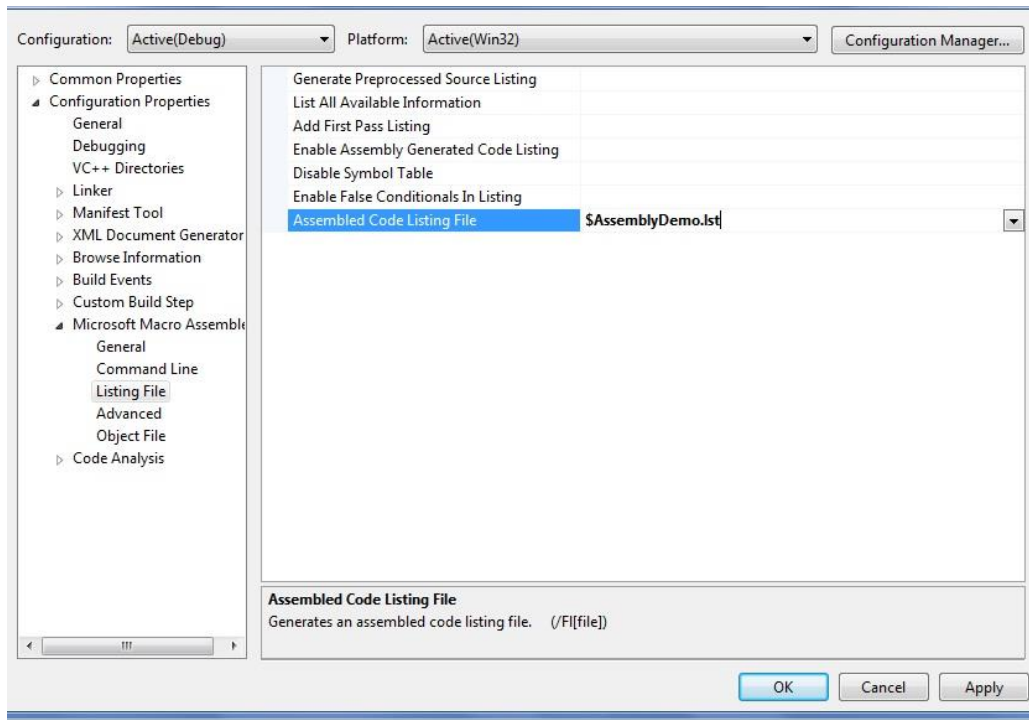
6. Now right-click your project again and click **Properties**. Now click the tiny arrow marker besides **Configuration Properties** to expand it. Now click **Microsoft Macro Assembler** and expand it.



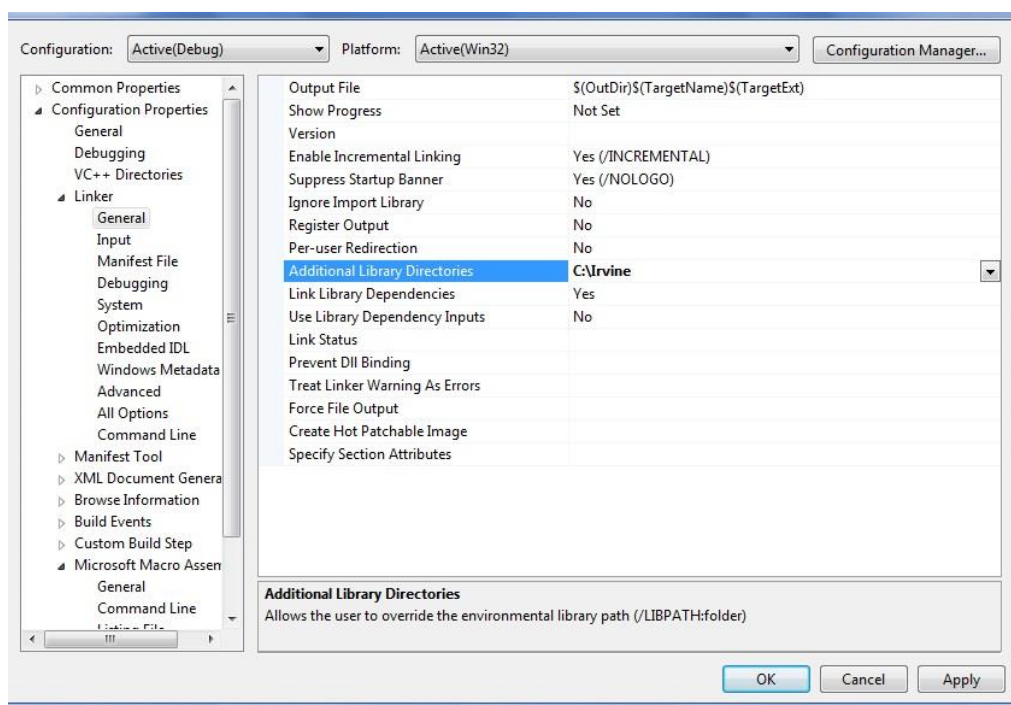
7. Now click **General** entry under Microsoft Macro Assembler and then set the value of **Include Paths** as **C:\Irvine**. The menu should now look like this.



8. Click **Listing File**. Select **Assembled Code Listing File** option and set its value as **\$AssemblyDemo.lst** (we have written *AssemblyDemo* here. If you have given some other name to your project then it will be like *\$YourProjectName.lst*).

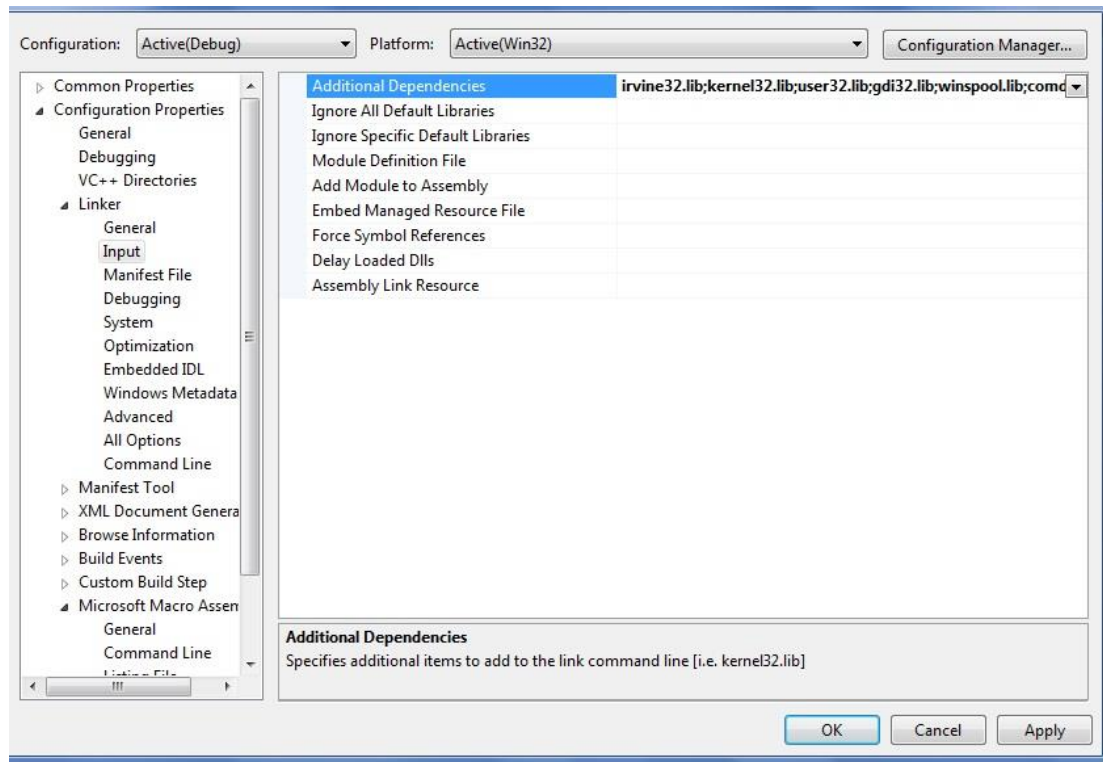


9. Click **Linker** tab to expand it. Select **General** and set the value of **Additional Library Directories** to **C:\Irvine**

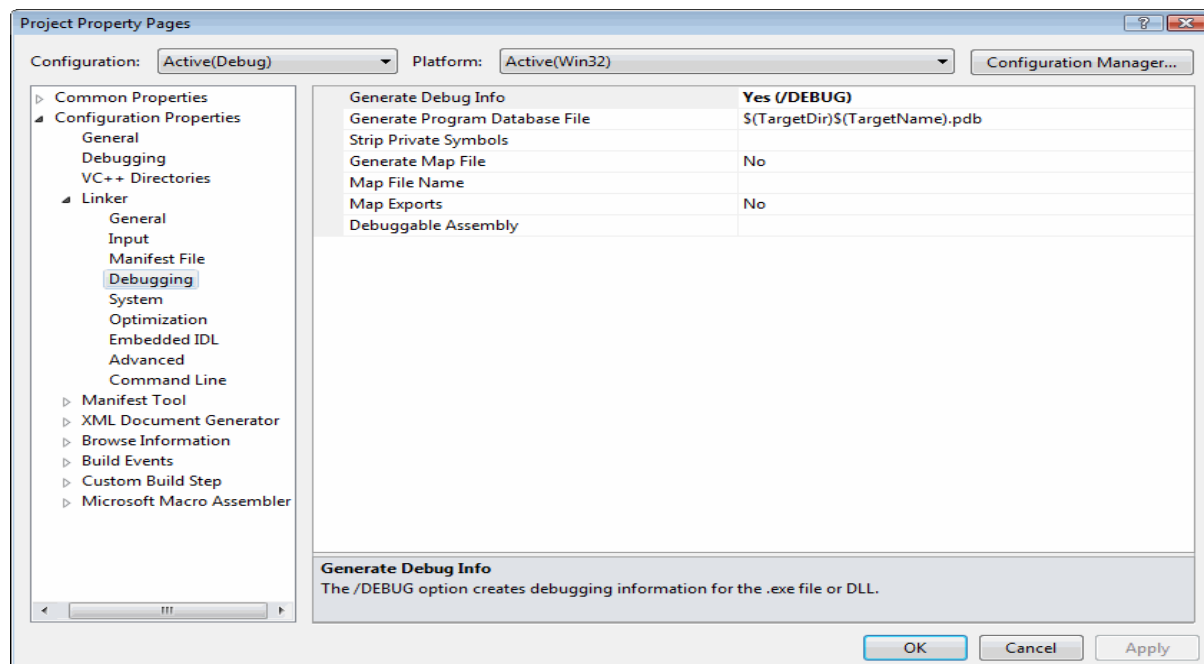




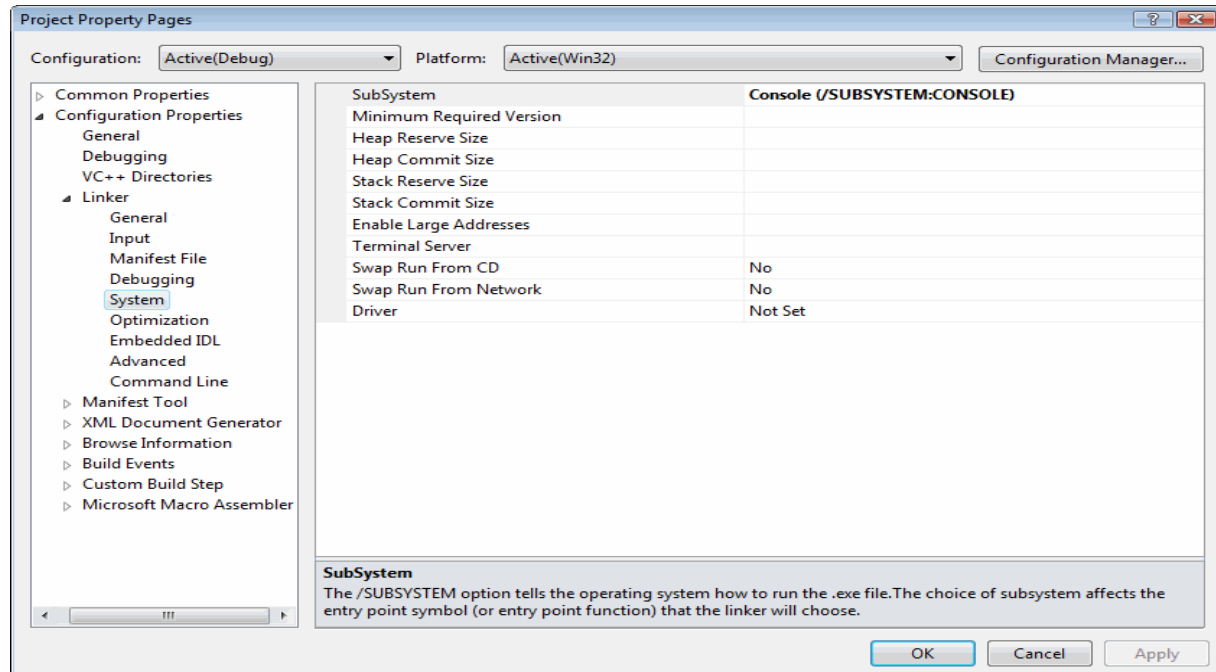
10. Click **Input**, select **Additional Dependencies**. You will see a list of different .lib file names written there, do not alter any of those. Write **irvine32.lib**; at the start of the list like this.



11. Click **Debugging** and make sure its **Generate Debug Info** option is set to **Yes (/DEBUG)**.



12. Click **System**, select **SubSystem** and set its value to **Console (/SUBSYSTEM:CONSOLE)**. Now click **Apply** and **OK** to close the Properties window.



Our Visual Studio 2010 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.

```

C:\Windows\system32\cmd.exe
EAX=00000010  EBX=00000025  ECX=00000000  EDX=000E1005
ESI=00000000  EDI=00000000  EBP=0031FD2C  ESP=0031FD24
EIP=000E101F  EFL=00000246  CF=0  SF=0  ZF=1  OF=0  AF=0  PF=1
Press any key to continue . . .

```

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 15h 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.

## 2.3 Debugging our program

We have seen how to configure Visual Studio 2010 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.

### 2.3.1 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.

### 2.3.2 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.

### 2.3.3 Code Stepping

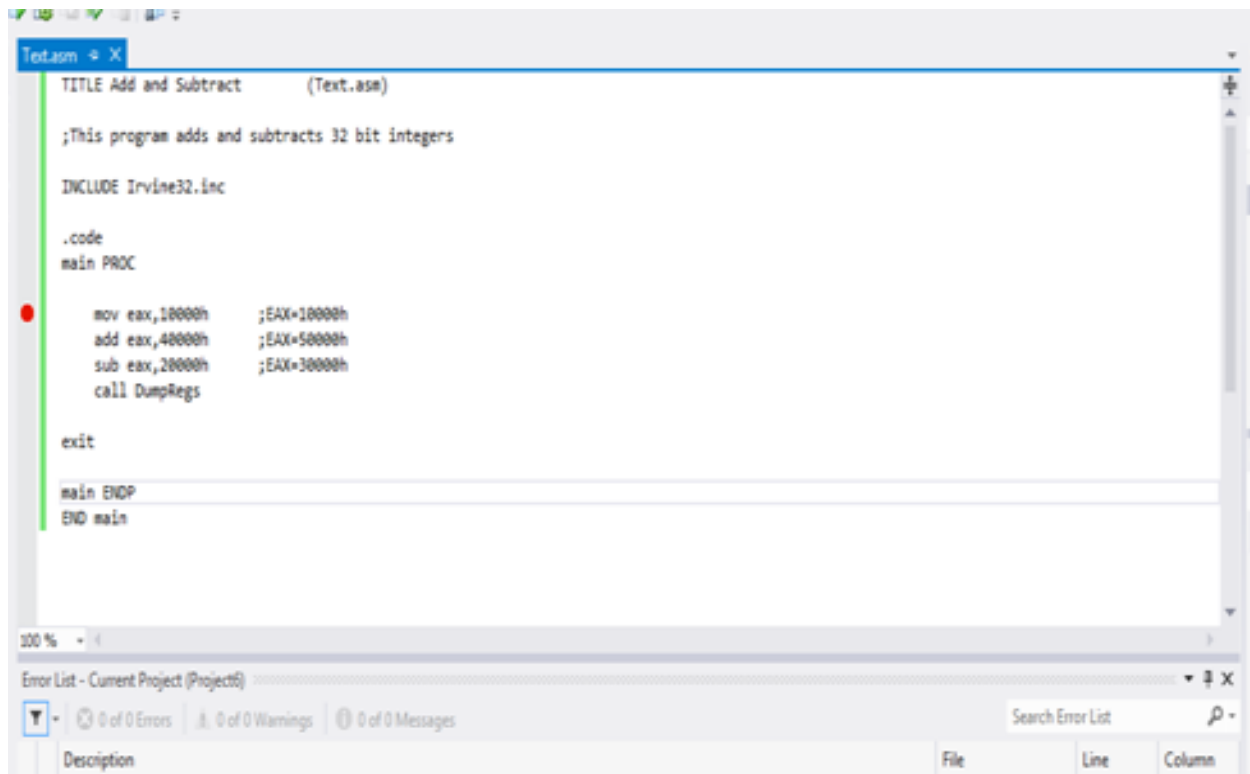
One of the most common debugging procedures is stepping: executing code one line at a time. The **Debug** menu provides three commands for stepping through code:

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

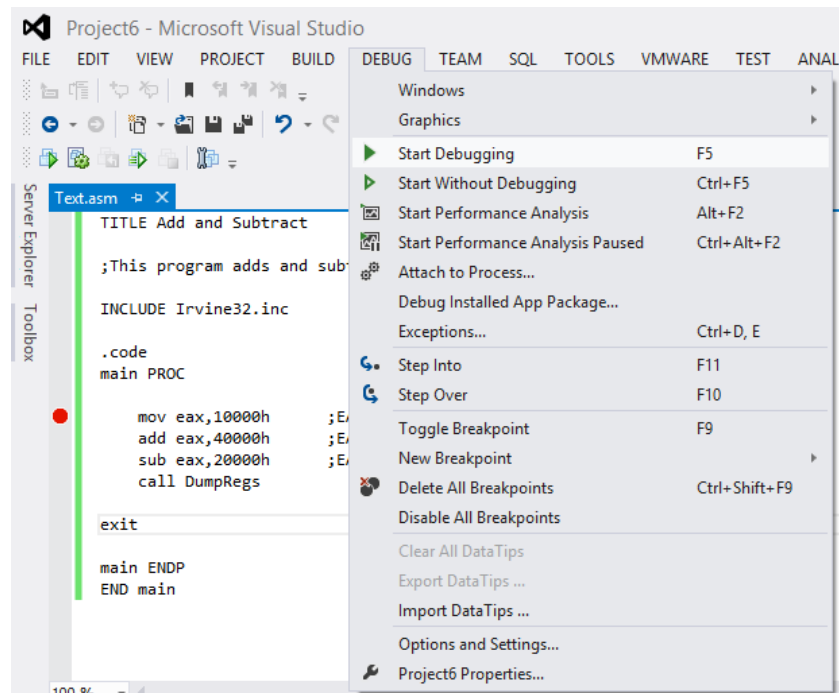
## 2.3.4 Single Stepping

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.

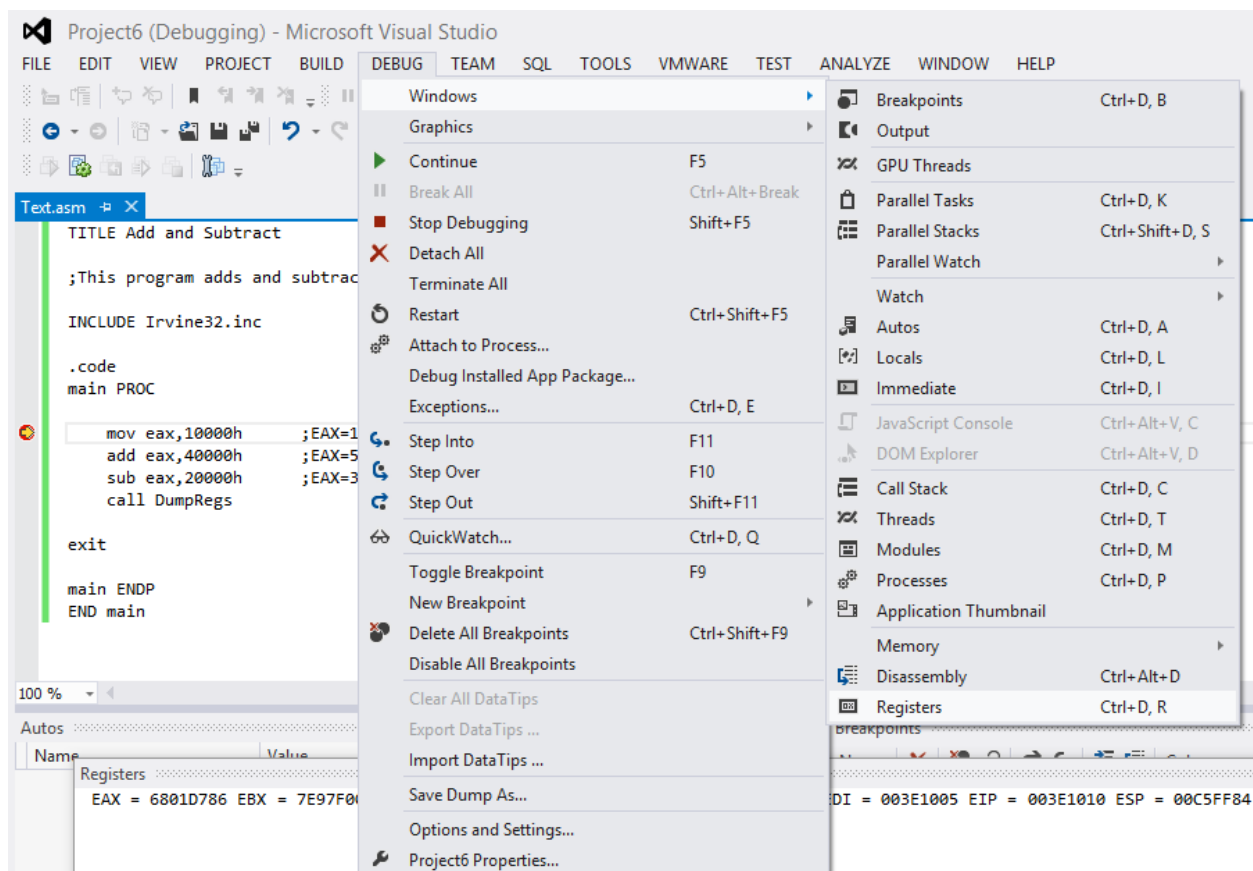
1. Right-click on line 6 to insert a breakpoint.




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.



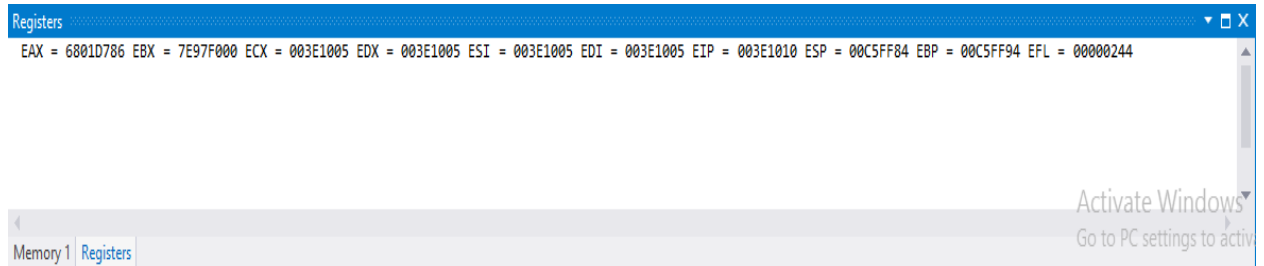
4. Breakpoint set on 1<sup>st</sup> instruction.




```

mov eax,10000h    ;EAX=10000h
add eax,40000h    ;EAX=50000h
sub eax,20000h    ;EAX=30000h
call DumpRegs

```



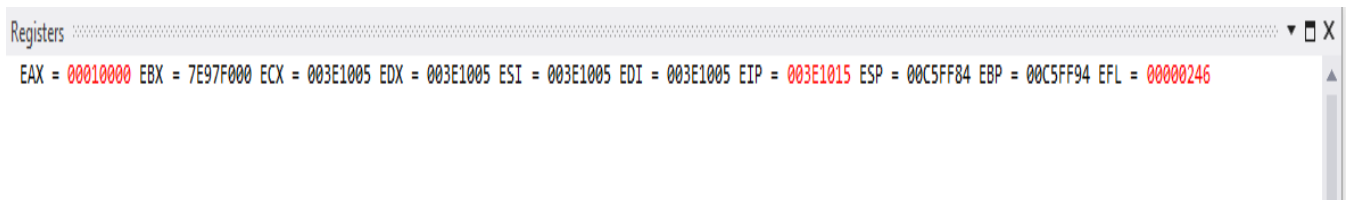
Press **F10** again to execute next line.




```

mov eax,10000h    ;EAX=10000h
add eax,40000h    ;EAX=50000h
sub eax,20000h    ;EAX=30000h
call DumpRegs

```



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



```

mov eax,10000h    ;EAX=10000h
add eax,40000h    ;EAX=50000h
sub eax,20000h    ;EAX=30000h
call DumpRegs

```

```
Registers
EAX = 00050000 EBX = 7E97F000 ECX = 003E1005 EDX = 003E1005 ESI = 003E1005 EDI = 003E1005 EIP = 003E101A ESP = 00C5FF84 EBP = 00C5FF94 EFL = 00000206
```

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**.

```
Registers
EAX = 00030000 EBX = 7E97F000 ECX = 003E1005 EDX = 003E1005 ESI = 003E1005 EDI = 003E1005 EIP = 003E101F ESP = 00C5FF84 EBP = 00C5FF94 EFL = 00000206
```



# Exercise

1. Install Visual Studio 2010 & 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 your program and note down the value of EAX after the execution of each line.
-