`

How to setup Irvine for VS 2013

1. GO TO THE FOLLOWING LINK

<http://kipirvine.com/asm/gettingStartedVS2013/index.htm>

1. This tutorial assumes that you are using the Seventh Edition of *Assembly Language for x86 Processors*. We show you how to set up Visual Studio 2013 (including Visual Studio 2013 Express for Windows Desktop, and Visual Studio Community 2013 edition) to work with the Microsoft assembler. Visual Studio 2013 Express and Visual Studio Community 2013 may be downloaded from Microsoft.com. (Note: the directions shown here are ***not*** designed for use with "Visual Studio Express 2013 for Windows" That product is designed for creating Windows Store apps in Visual Basic and C#.)

Topics:

[Tutorial: Building a 32-Bit Assembly Language Program](http://kipirvine.com/asm/gettingStartedVS2013/index.htm#tutorial32)

[Tutorial: Building and Running a 64-Bit Program](http://kipirvine.com/asm/gettingStartedVS2013/index.htm#tutorial64)

[MASM syntax highlighting](http://kipirvine.com/asm/gettingStartedVS2013/index.htm#syntax)

[Assembling, linking, and debugging with a batch file](http://kipirvine.com/asm/gettingStartedVS2013/index.htm#batch)

[Creating a Project from Scratch](http://kipirvine.com/asm/gettingStartedVS2013/index.htm#CreatingProject)

[Using the Visual Studio debugger](http://kipirvine.com/asm/debug/vstudio2013/index.htm)

**Required Setup for 32-bit Applications**

1. First, you must install Visual Studio and select the C++ language configuration option the first time you run it. All versions of Visual Studio include the Microsoft Assembler (MASM) version 12.0. You can verify that the Microsoft Assembler is installed by looking for the file **ml.exe** in the \vc\bin folder of your Visual Studio installation directory, such as c:\Program Files\Microsoft Visual Studio 12.0\vc\bin.

#### Installing the Book's Example Programs

[Click this link](http://kipirvine.com/asm/examples/index.htm) to get the latest copy of the book's link libraries and example programs. The examples are stored in a Microsoft Install (.MSI) file that installs into the **c:\Irvine** folder. Unless you have some objection to using that location, do not alter the path. (Note to lab administrators: you can designate c:\Irvine directory as read-only.) If you plan to change the installation location, read our instructions relating to [Creating a Project from Scratch](http://kipirvine.com/asm/gettingStartedVS2013/index.htm#CreatingProject).

The following files will be copied into the c:\Irvine directory:

|  |  |
| --- | --- |
| Filename | Description |
| b16.asm, b32.asm | Blank templates for 16-bit and 32-bit assembly language source files |
| GraphWin.inc | Include file for writing Windows applications |
| Irvine16.inc | Include file used with the Irvine16 link library (16-bit applications) |
| Irvine16.lib | 16-bit link function library used with this book |
| Irvine32.inc | Include file used with the Irvine32 link library (32-bit applications) |
| Irvine32.lib | Irvine's 32-bit link library |
| Irvine64.obj | Irvine's 64-bit library |
| Kernel32.lib | 32-bit link library for Windows API |
| Link16.exe | 16-bit Microsoft linker |
| Macros.inc | Irvine's macro include file (see Chapter 10) |
| make16\_vs2012.bat | Visual Studio 2012 batch file for building 16-bit applications |
| make16\_vs2013.bat | Visual Studio 2013 batch file for building 16-bit applications |
| SmallWin.inc | Small-sized include file containing MS-Windows definitions, used by Irvine32.inc |
| User32.lib | MS-Windows basic I/O link library |
| VirtualKeys.inc | Keyboard code definitions file, used by Irvine32.inc |

A subdirectory named **Examples** will contain all the example programs shown in the book, source code for the book's 16-, 32-, and 64-bit libraries, and two sample Visual Studio projects.

### Setting up Visual Studio

You will only have to do these steps the first time you use Visual Studio.

#### Add the Start Without Debugging command to the Debug menu

It's very useful to run programs without having to debug them. To do that, you will want to add a new command to the Debug menu: Start Without Debugging. Here's how to do it:

1. From the Tools, menu, select Customize.
2. Select the Commands tab.
3. Select Menu bar (radio button).
4. Click the Add Command button.
5. Select Debug from the Categories list.
6. Select Start Without Debugging in the right-hand list box.
7. Click the OK button.
8. Click the Close button.

In fact, you can use the same sequence to customize any of the menus and toolbars in Visual Studio.

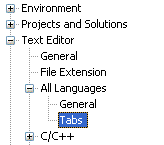
#### Select the C++ Configuration

(Skip this topic if you installed Visual Studio Express.) Visual Studio Professional, Ultimate, and Premium editions support multiple programming languages and application types. The C++ programming language configuration most closely matches that of assembly language programming, so we suggest the following steps:

1. Select Tools | Import and Export Settings from the menu
2. Select the "Import selected environment settings" radio button
3. Select the "No, just import..." radio button
4. Select "Visual C++" from the Default Settings List and click the Next button
5. Click the Finish button, then click the Close button
6. Notice the tabs on the left and right sides of the Visual Studio workspace. Close the Server Explorer, Toolbox, and Properties tabs. Use the mouse to drag the Solution Explorer tool window to the right side of the workspace. You can also select other tabs at the bottom of this window, such as "Class View", "Property Manager", and "Team Explorer", and close them. They will not be used in the future. If you need to bring back the Solution Explorer window at any time in the future, select View from the menu, and locate Solution Explorer in the list of views.

#### Set the Tab Size to 5

(This is an optional step.) Start Visual Studio, and select **Options** from the **Tools** menu. Select **Text Editor**, Select **All Languages**, and select **Tabs**. Optionally, you may want to select the **Insert spaces** radio button:



Set the Tab Size and Indent Size to 5.

## Tutorial: Building a 32-Bit Assembly Language Program

Now you're ready to open and build your first 32-bit project.

#### Opening a Project

Visual Studio requires assembly language source files to belong to a *project*, which is a kind of container. A project holds configuration information such as the locations of the assembler, linker, and required libraries. A project has its own folder, and it holds the names and locations of all files belonging to it. We have created a sample project folder in the *c:\Irvine\examples* directory, and its name is *Project32*.

Do the following steps, in order:

1. Copy the c:\Irvine\Examples folder to a location on your hard drive that permits you to read and write files. You can use a USB drive, although Visual Studio may run a little more slowly when it creates temporary files during the build process.
2. Start Visual Studio.
3. To begin, open our sample Visual Studio project file by selecting **File/Open/Project** from the Visual Studio menu.
4. Navigate to the **c:\Irvine\Examples\Project32** folder and select the file named **Project.sln**.
5. Once the project has been opened, you will see the project name in the Solution Explorer window. If there are any files with .asm file extensions in the Solution Explorer window, select and delete them now.
6. Next, you will add an existing source code file to the project: In the Solution Explorer window, right-click on **Project**, select **Add**,  select **Existing Item**, navigate to the "Examples\ch03" folder, select **AddTwo.asm**, and click the **Add** button to close the dialog window. (You can use this sequence of commands in the future to add any asm file to a project.) You should now see the AddTwo.asm file in the Solution Explorer window.
7. Next, open the AddTwo.asm for editing by double-clicking its filename in the Solution Explorer window.

You should see the following program in the editor window:

|  |
| --- |
| ; 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 |

In the future, you can use this file as a starting point to create new programs by copying it and renaming the copy in the Solution Explorer window.

#### Build the Program

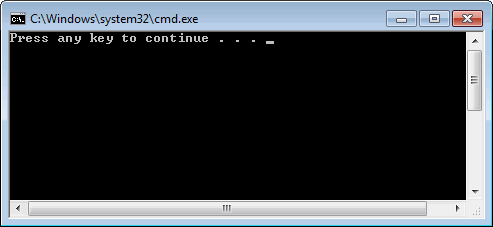
Now you will build (assemble and link) the sample program. Select **Build Project** from the Build menu. In the Output window for Visual Studio at the bottom of the screen, you should see messages similar to the following, indicating the build progress:

|  |
| --- |
| 1>------ Build started: Project: Project, Configuration: Debug Win32 ------  1> Assembling ..\ch03\AddTwo.asm...  1> Project.vcxproj -> G:\ASM Book Examples\Project32\_VS2013\Debug\Project.exe  ========== Rebuild All: 1 succeeded, 0 failed, 0 skipped ========== |

If you do not see these messages, the project has probably not been modified since it was last built. No problem--just select **Rebuild Project** from the Build menu.

#### Run the Program

Select **Start without Debugging** from the Debug menu. The following console window should appear, although your window will be larger than the one shown here:



The "Press any key to continue..." message is automatically generated by Visual Studio.

Congratulations, you have just run your first Assembly Language program!

Press any key to close the Console window.

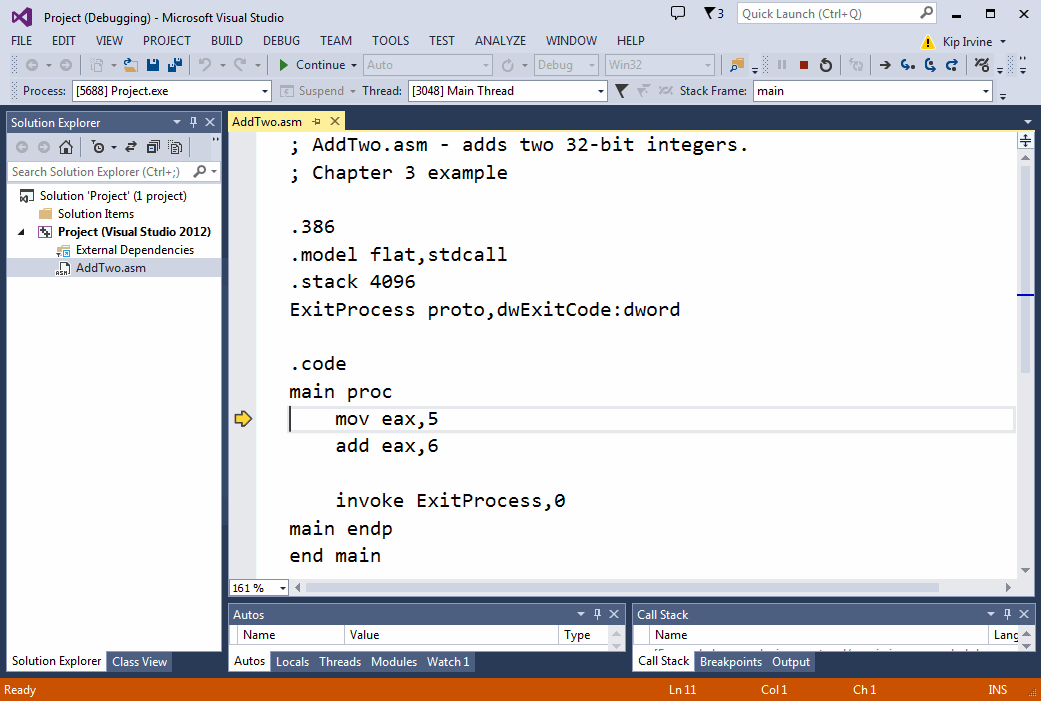
|  |
| --- |
| **TIP:** When you assembled and linked the project, a file named **Project.exe** was created inside the project's \Debug folder. This file executes when you run the project. You can execute Project.exe by double-clicking its name inside Windows Explorer, but it will just flash on the screen and disappear. That is because Windows Explorer does not pause the display before closing the command window. On the other hand, you can open a Command prompt window, move to the Debug directory, and run Project.exe by typing "Project" (without the quotes). You will need to do some reading on Windows shell commands if you plan to use the command line. |

Any time you want to remove a source file from the Visual Studio window, right-click its filename and select **Remove**. The file will not be deleted from the file system. On the other hand, if you want to delete the file, select it and press the Del key.

### Step 5: Running the Sample Program in Debug Mode

In this step, you set a breakpoint inside the sample program. Then you use the Visual Studio debugger to step through the program's execution one statement at a time.

1. Make sure the ASM source code file is open in the editor window.
2. To begin stepping through your program in Debug mode, press the F10 key.
3. A yellow arrow should appear next to the first program statement.The arrow indicates that the statement is next to be executed.
4. Press the F10 key (called *Step Over*) to execute the current statement. Continue pressing F10 until the program is about to execute the **invoke** statement.
5. A small black window icon should appear on your Windows status bar. Open it and look at the contents of the Command window. The window should be blank because this program does not display any output.
6. Press F10 one more time to end the program.



#### Registers

If you want to display the CPU registers, do the following: Start debugging the program, then select *Windows* from the *Debug* menu. Select *Registers* from the drop-down list. The Registers window may appear at the bottom of the workspace, as a tab highlighted in yellow. Use the mouse to drag the window to the right side of the work area. Right click inside the Registers window and check the item *Flags* to enable the display of CPU status flags.

You can interrupt a debugging session at any time by selecting *Stop Debugging* from the Debug menu. You can do the same by clicking the maroon-colored square button on the toolbar. To remove a breakpoint from the program, click on its red dot to make it disappear.

#### Setting a BreakPoint

If you set a breakpoint in a program, you can use the debugger to execute the program a full speed (more or less) until it reaches the breakpoint. At that point, the debugger drops into single-step mode.

1. In our sample program, click the mouse along the border to the left of the **mov eax,5** statement. A large red dot should appear in the margin.
2. Select Start Debugging from the Debug menu. The program should run, and pause on the line with the breakpoint, showing the same Yellow arrow as before.
3. Press F10 until the program finishes.

You can remove a breakpoint by clicking its red dot with the mouse. Take a few minutes to experiment with the Debug menu commands. Set more breakpoints and run the program again. For the time being, you can use the F11 key to step through the program in the same way the F10 key did.

#### Building and Running Other Programs

Suppose you want to run another example program, or possibly create your own program. You can remove the existing assembly language file from the Solution Explorer window and insert a new .asm file into the project.

* To remove a program from a project without deleting the file, right-click its name in the *Solution Explorer window*. In the context menu, select **Remove**. If you change your mind and decide to add it back to the project, right-click in the same window, select **Add,** select **Existing item,** and select the file you want to add.

#### Adding a File to a Project

An easy way to add an assembly language source file to an open project is to drag its filename with the mouse from a Windows Explorer window onto the name of your project in the Solution Explorer window. The physical file will not be copied--the project only holds a reference to the file's location. Try this now:

1. Remove the AddTwo.asm file from your project.
2. Add a reference to the file Examples\ch03\AddTwoSum.asm to the project.
3. Build and run the project.

#### Copying a Source File

One way to make a copy of an existing source code file is to use Windows Explorer to copy the file into your project directory. Then, right-click the project name in Solution Explorer, select Add, select Existing Item, and select the filename.

## Tutorial: Building and Running a 64-Bit Program

In this tutorial, we will show you how to assemble, link, and run a sample 64-bit program. We assume you have already completed our tutorial entitled *Building a 32-Bit Assembly Language Program*.

Do the following steps, in order:

1. In Visual Studio, select File/Open/Project, navigate to the **c:\Irvine\Examples\Project64** folder and select the file named **Project64.sln**.
2. Delete any existing files with the .asm extension from the Solution Explorer window.
3. Now you will add an existing source code file to the project. To do that, right-click on **Project**, select **Add**,  select **Existing Item**, navigate to the book's Examples\64 bit\ch03" folder, select **AddTwoSum\_64.asm**, and click the **Add** button to close the dialog window.
4. Next, open the AddTwoSum\_64.asm file for editing by double-clicking its filename in the Solution Explorer window.

You should see the following program in the editor window:

; AddTwoSum\_64.asm - Chapter 3 example.

ExitProcess proto

.data

sum qword 0

.code

main proc

mov rax,5

add rax,6

mov sum,rax

mov ecx,0

call ExitProcess

main endp

end

#### Build the Program

Select **Build Project** from the Build menu. You should see text written to Visual Studio's output window like the following:

1>------ Build started: Project: Project64, Configuration: Debug Win64 ------

1> Assembling ..\ch03\64 bit\AddTwoSum\_64.asm...

1> Project64.vcxproj -> D:\ASM Book Examples\Project64\Debug\Project64.exe

========== Rebuild All: 1 succeeded, 0 failed, 0 skipped ==========

If you do not see these messages, the project has probably not been modified since it was last built. No problem--just select **Rebuild Project** from the Build menu.

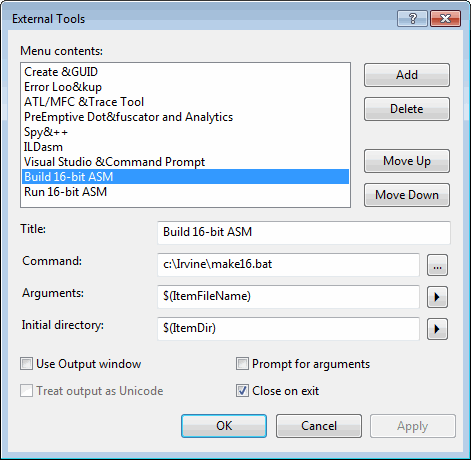
You use the same Visual Studio commands to run and debug 64-bit programs as you would for 32-bit programs.

## Building 16-bit Applications (Chapters 14-17)

Only Chapters 14 through 17 require you to build 16-bit applications. Except for a few exceptions, which are noted in the book, your 16-bit applications will run under the 32-bit versions of Windows (XP, Vista, 7). But 16-bit applications will not run directly in any 64-bit version of Windows.

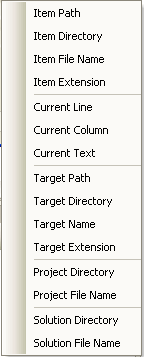
If you plan to build 16-bit applications, you need to add two new commands to the Visual Studio Tools menu. To add a command, select **External Tools** from the Tools menu. The following dialog will appear, although many of the items in your list on the left side will be missing. The filename in the sample image (make16.bat) has been replaced by two files:

* make16\_vs2012.bat for Visual Studio 2012)
* make16\_vs2013.bat for Visual Studio 2013)



### Step 1: Create the Build 16-bit ASM Command

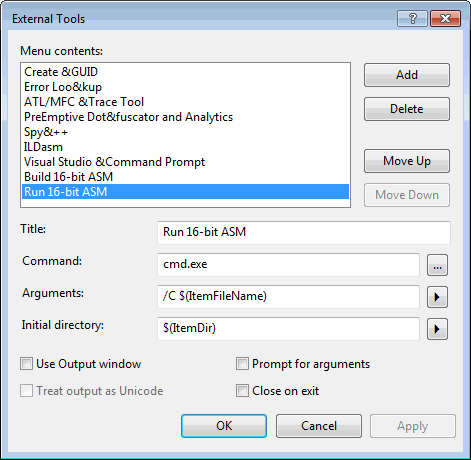
Click the **Add** button and fill in the Title, Command, Arguments, and Initial directory fields as shown in the screen snapshot. If you click the buttons with arrows on the right side of the Arguments and Initial directory fields, a convenient list appears. You can select an item without having to worry about spelling:



Click the **Apply** button to save the command.

### Step 2: Create the Run 16-bit ASM Command

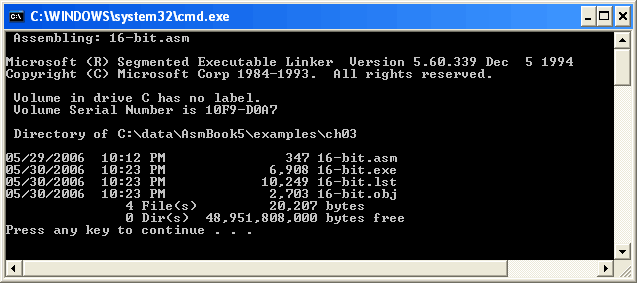
Click the Add button again, and create a new command named **Run 16-bit ASM**:



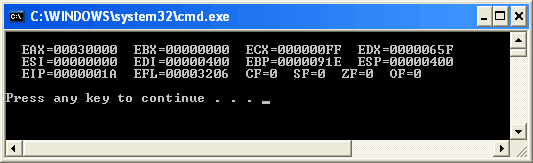
Uncheck the "Close on exit" option and click the OK button to save the command and close the External Tools dialog.

#### Testing Your new 16-Bit Commands

To test your new 16-bit commands, close any Visual Studio project that happens to be open. Then, select File | Open | File from the menu and choose the file named **16-bit.asm** from the ch03 folder in the book's example programs. Select **Build 16-bit ASM** from the Tools menu. The following command window should appear, showing the successful execution of the assembler and linker, followed by a listing of all files related to this program:



Press a key to close the window. Next, you will run the program. Select **Run 16-bit ASM** from the Tools menu. The following window will appear, although the contents of all registers except EAX will be different:



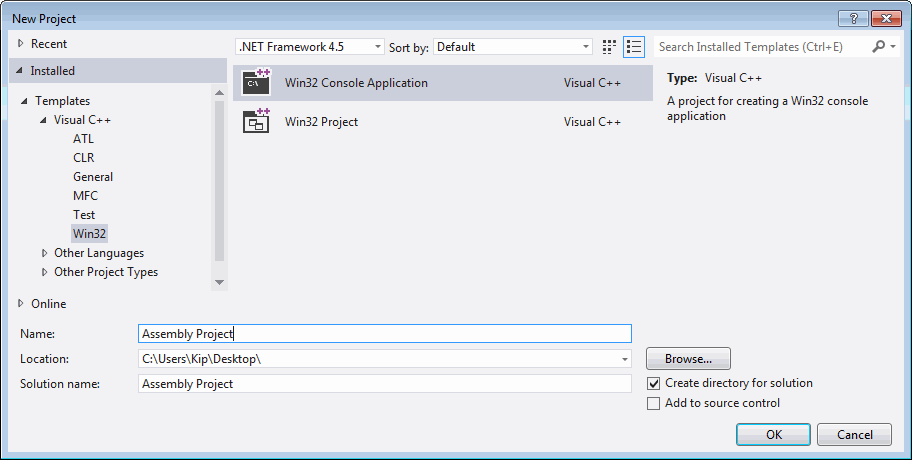
Press a key to close the window.

You have completed the setup for building and running 16-bit assembly language programs.

## Creating a Project From Scratch

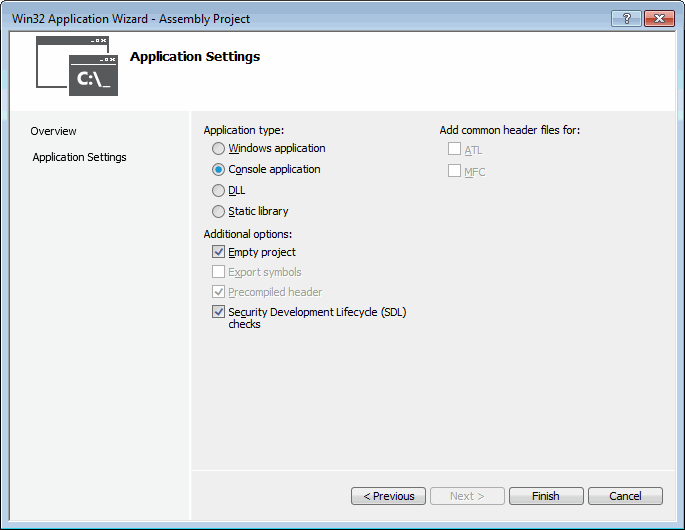
Visual Studio makes it possible (in 12 easy steps) to create an Assembly Language project from scratch. In the first step, you will create a Win32 Console application designed for C++, and just modify the custom build rules.

Step 1: Select New from the File menu, then select Project. In the New Project window, select **Win32** under Visual C++ in the left panel, and select **Win32 Console Application** in the middle panel. Give your project a suitable name (near the bottom of the window):



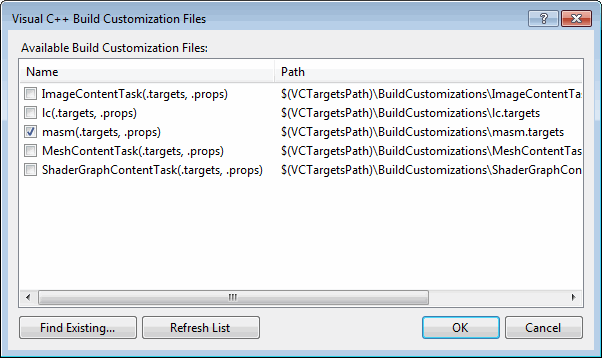
(Depending on your Visual Studio configuration, you might have to find Visual C++ under the "Other Languages" category in the left panel.)

Step 2: Click the **OK** button to continue. The Win32 Application Wizard window will appear. Select **Application Settings**, and then select the **Empty project** check box.

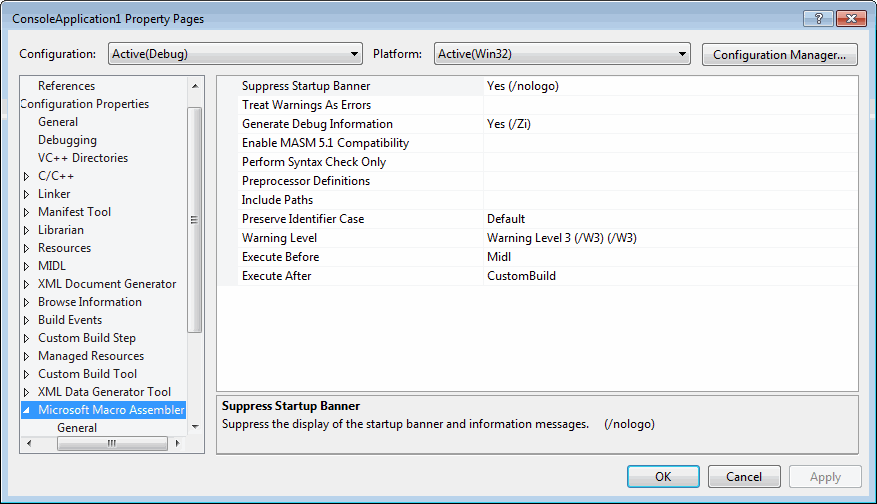


Step 3: Click the **Finish** button to save the new project.

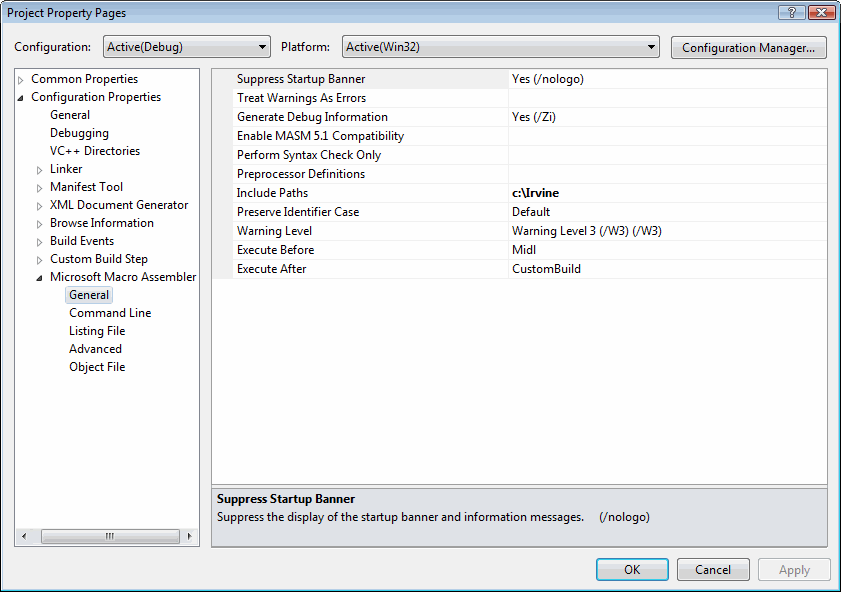
Step 4: In the Solution Explorer window, delete the Header Files, Resource Files, and Source Files folders. They are not necessary. In the same window, right click your project name and select **Build Customizations** from the popup menu. When you see this window, select the check box next to **masm** and click the OK button to close the dialog.



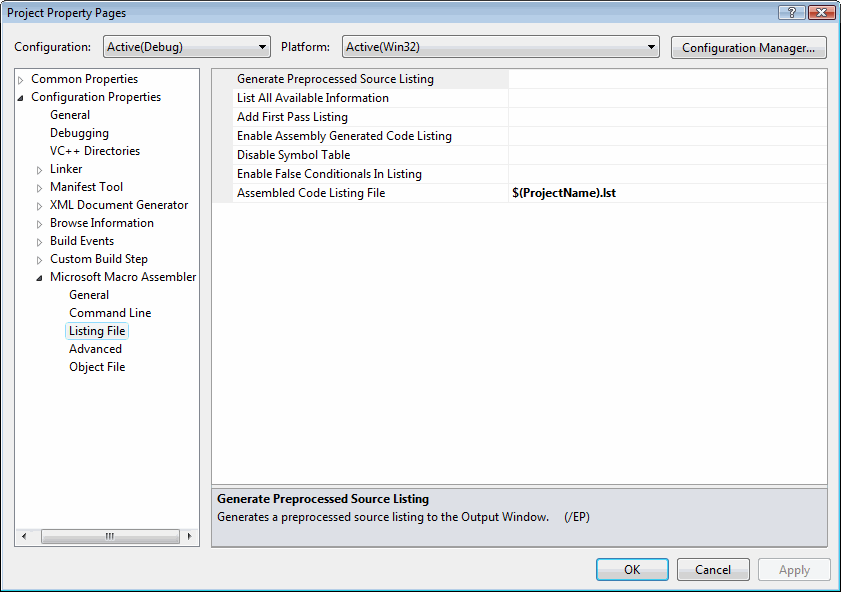
Step 5: Select **Properties** from the Project menu. Expand the entry under **Configuration Properties**. Then expand the entry named **Microsoft Macro Assembler**. This is what you should see:



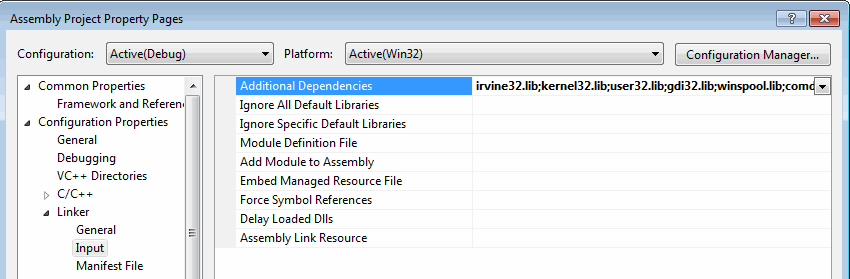
Step 6: Modify the **Include Paths** option so it equals "C:\Irvine". This tells the assembler where to find files having a filename extension of ".inc". Here is a sample:



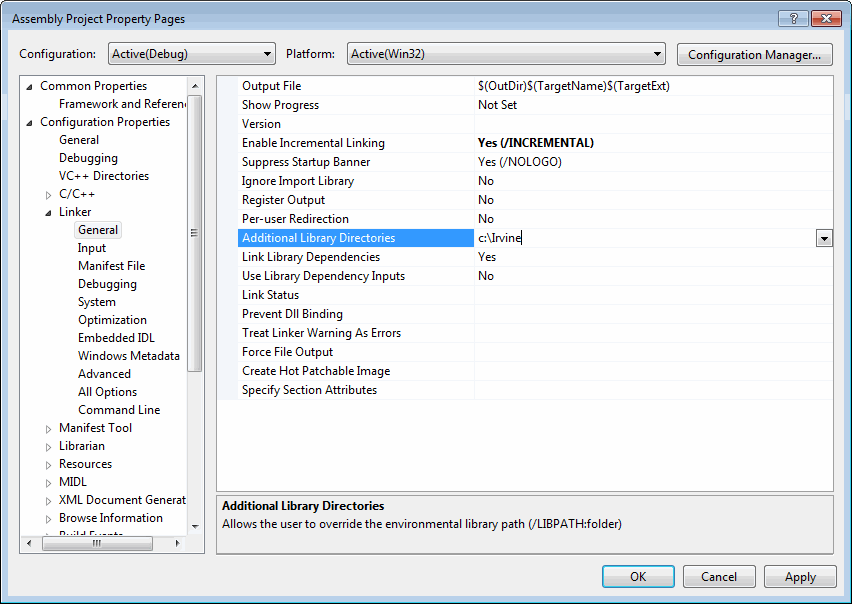
Step 7: Next, select the **Listing File** entry, also in the Microsoft Macro Assembler group. Modify the Assembled Code Listing File entry (shown below) so it contains **$(ProjectName).lst.** This uses a built-in variable to identify the name of the source input file, with a file extension of .lst. So, if your program were named main.asm, the listing file would be named main.lst:



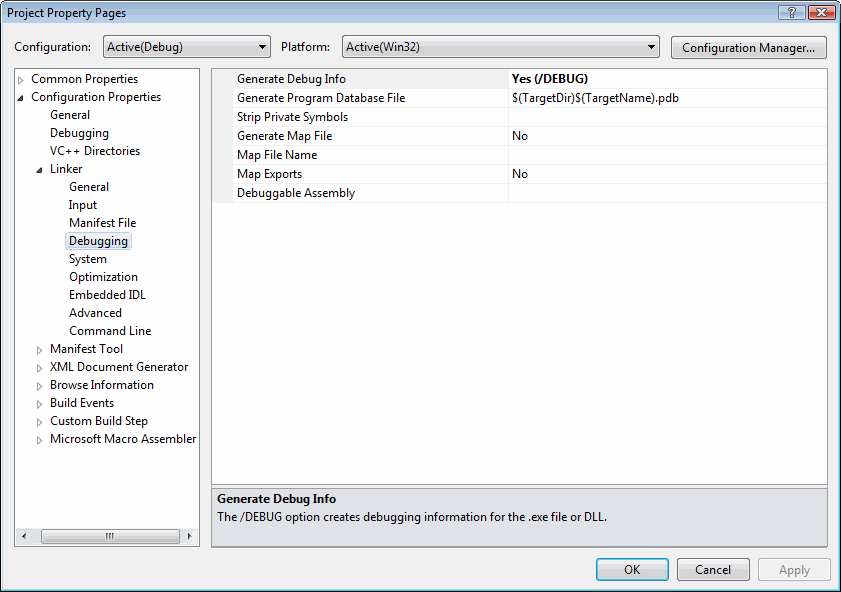
Step 8: Find the Linker entry under **Configuration Properties**. Select the **Input** entry, and insert **irvine32.lib;** at the beginning of the **Additional Dependencies** entry. The **irvine32.lib** file is the link library file supplied with this book. The filenames must be separated by semicolons.



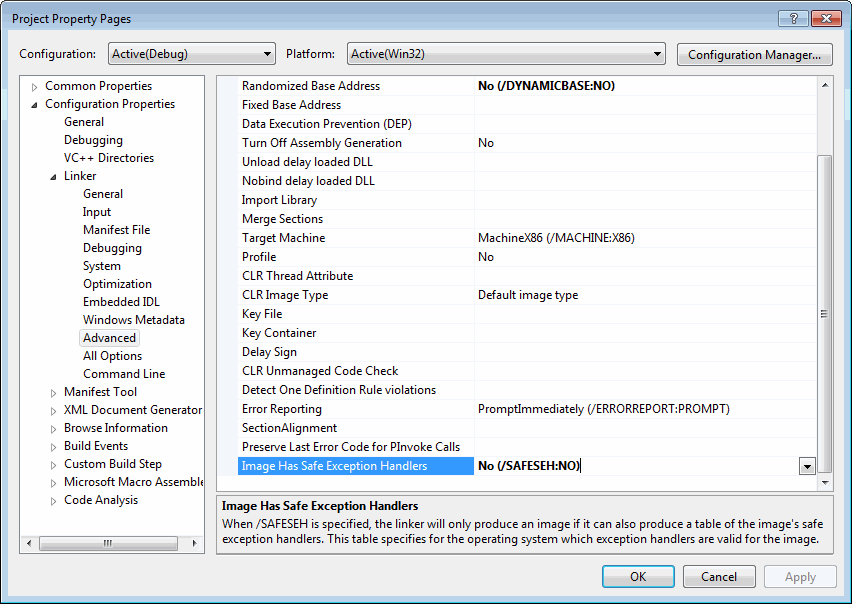
Step 9: Select **Linker** under Configuration Properties, and then select **General**. The **Additional Library Directories** entry must contain **c:\Irvine** so the linker can find the Irvine32.lib library file:



Step 10: Select **Linker** under the **Configuration Properties** and select **Debugging**. Verify that the **Generate Debug** **Info** option is equal to **Yes:**



Step 11: Select **Advanced** under the **Linker** entry. Set the Image Has Safe Exception Handlers option to **No**.



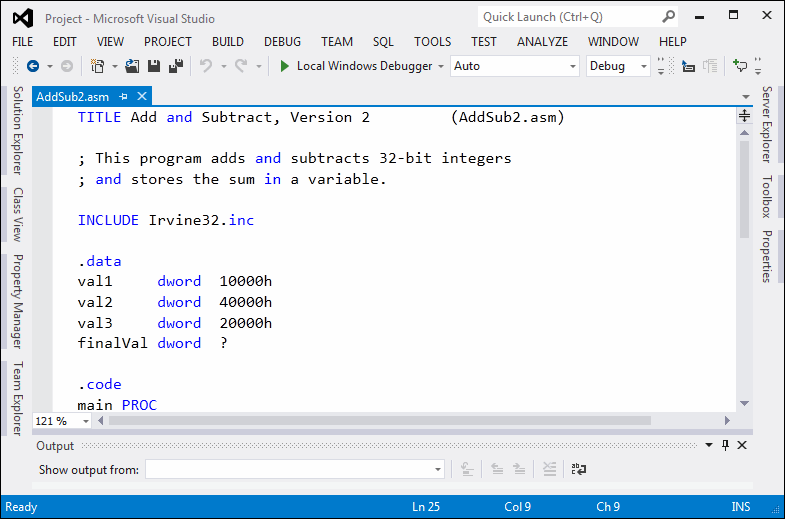
Step 12: Click the OK button to close the Property Pages window. Verify that your project has been created correctly by doing the following: (1) Add the AddTwo.asm program from the Chapter 3 examples folder to the project; (2) Build and debug the program as you did earlier in this tutorial.

### Generating a Source Listing File

Open the project. From the menu, select **Project**, select **Properties**. In the list box, select **Microsoft Macro Assembler**, then select **Listing File**. Set the **Assembled Code Listing file** option to **$(InputName).lst**.

### MASM syntax highlighting

When a text editor uses syntax highlighting, language keywords, strings, and other elements appear in different colors. Visual Studio highlights MASM reserved words and strings, as shown in the following example:



This won't happen automatically, but you can create a syntax definition file named Usertype.dat that contains MASM keywords. Then when Visual Studio starts, it reads the syntax file and highlights MASM keywords.

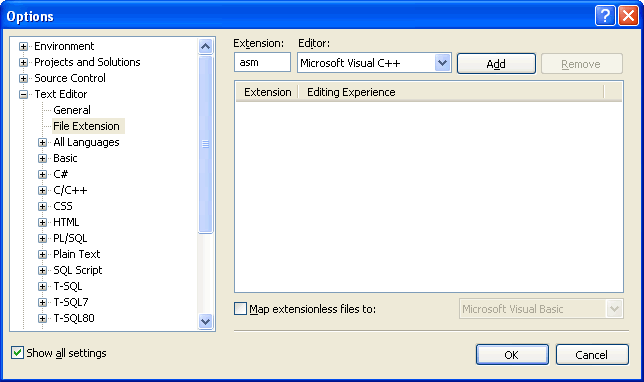
Here are the required steps to set up MASM syntax highlighting:

1) Download the [Usertype.dat](http://kipirvine.com/asm/gettingStartedVS2013/usertype.zip) file (enclosed in a ZIP file) given here to a folder in which you have read/write permissions. If you are using Windows 7, download to My Documents, or C:\temp, or any folder that doesn't have security restrictions.

2) Copy Usertype.dat to the C:\Program Files\Microsoft Visual Studio 12.x\Common7\IDE folder. *In 64-bit windows, Program Files will be named Program Files(x86).*

Windows will display a verification dialog before copying the file.

3) Open Visual Studio, select **Options** from the Tools menu, select **Text Editor**, and select **File Extension**. On the right side of the dialog (shown below), enter **asm** as the extension, select **Microsoft Visual C++** from the Editor list, and click the **Add** button. Click the **OK** button to save your changes.



Close Visual Studio and restart it. Open your project and display an ASM file. You should see syntax highlighting in the editor. There is a glitch in the highlighting--assembly language comment line starts start with a semicolon, which C++ doesn't recognize. But this is a simple workaround: add an extra // right after the semicolon, like this, which will cause the comments to appear in their usual green color:

;// AddTwo.asm - adds two 32-bit integers.

;// Chapter 3 example

## Assembling, Linking, and Debugging a 32-bit Program with a Batch File

Many people like to use a *Windows batch file* to assemble and link programs. A batch file is a text file containing a sequence of commands that execute as if they had been typed at the command prompt. In fact, they are powerful enough to contain variables, loops, IF statements, and so on.

The easiest way to run a batch file is to first open a Command window and then type the name of the batch file (along with arguments) at the command prompt. To open a Command window, you must execute a program named **cmd.exe**. We will make that easy for you.

**Step 1:** [Download a ZIP file](http://kipirvine.com/asm/gettingStartedVS2013/BatchSampleVS2013.zip) containing the following items:

* **A shortcut to cmd.exe,** which opens a Command window in the current directory
* **asm32.bat**, a batch file for assembling and linking programs
* **main.asm**, a sample assembly language program

**Step 2:** Extract the ZIP file into the c:\Irvine\Examples directory on your computer.

**Step 3:** If you're running 64-bit Windows, open the file with a text editor (such as NotePad++) and find this line:

set VS\_HOME=C:\Program Files\Microsoft Visual Studio 12.0

And change it to the correct path for Visual Studio:

set VS\_HOME=C:\Program Files (x86)\Microsoft Visual Studio 12.0

**Step 4:** Do the following:

* Copy asm32.bat to any directory on your system path. By doing this, you make it possible for MS-Windows to recognize **asm32** as a valid command when typed at the MS-Windows command prompt.(If you want to find out which directories are on the current system path, type **path** and press Enter at the system command prompt.)
* Double-click the shortcut to **cmd.exe**. A Command window should appear.
* At the command prompt in this window, type **asm32** and press Enter. This will execute the asm32 batch file and display help information.

|  |
| --- |
| This file assembles, links, and debugs a single assembly language  source file. Before using it, install Visual Studio 2013 in the following  directory:  C:\Program Files\Microsoft Visual Studio 12.0  Next, install the Irvine link libraires and include  files in the following directory: C:\Irvine  Finally, copy this batch file to a location on your system path.  We recommend the following directory:  C:\Program Files\Microsoft Visual Studio 12.0\VC\bin  Command-line syntax:  asm32 [/H | /h | -H | -h] -- display this help information  asm32 filelist -- assemble and link all files  asm32 /D filelist -- assemble, link, and debug  asm32 /C filelist -- assemble only  <filelist> is a list of up to 5 filenames (without extensions),  separated by spaces. The filenames are assumed to refer to files  having .asm extensions. Command-line switches are case-sensitive. |

Type the following command to assemble and link a source file named **main.asm**:

asm32 main

You should see the following messages:

|  |
| --- |
| Assembling: main.asm  The file main.obj was produced.  ..................................  Linking main.obj to the Irvine32, Kernel32, and User32 libraries.  The file main.exe was produced.  .................................. |

In fact, several files were produced.

* main.obj - the object file
* main.ilk - incremental link status file
* main.pdb - debug symbol file

If there were syntax errors in your program, you would see error messages generated by the assembler. Here is an example:

Assembling: main.asm  
main.asm(9) : error A2008: syntax error : myMessage  
main.asm(15) : error A2006: undefined symbol : myMessage

You would then open the main.asm file with a text editor (such as Notepad), fix the errors, and run the asm32 batch file again.

|  |
| --- |
| Although we used a file named main.asm in this example, the asm32.bat batch file will work for any assembly language file, regardless of the name. The only requirement is that your assembly language source file have a **.asm** filename extension. |

#### Assembling Programs in Other Directories

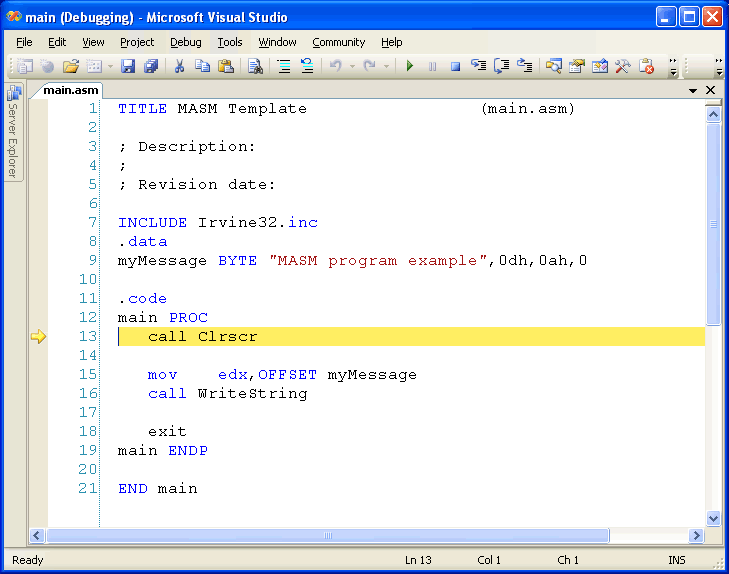
No doubt, you will want to assemble programs in various different disk folders, not just the batch\_sample folder used in the foregoing example. All you need to do is copy the **cmd.exe** shortcut we gave you to your working directory, where your assembly language source files are located. When you double-click to run the shortcut, it will open a Command window in the current folder.

#### Assembling, Linking, and Debugging

In addition to assembling and linking, you can use the asm32.bat file to launch your program in the Visual Studio debugger. Try the following command:

asm32 /D main

If the program assembles and links with no errors, your program should load in Visual Studio. The first time you do this with a new program, the source code will not appear. All you have to do is press the **F10 key** to begin debugging, and your program should appear with a yellow band across the first executable line:



This page contains the latest versions of the Irvine16 and Irvine32 link libraries, as well as the book's example programs and library source code.

|  |  |
| --- | --- |
| **Date** | **Filename** |
| 8/15/2016 | [Updated 32-bit Visual Studio project for VS2015](http://kipirvine.com/asm/examples/Project32_VS2015.zip) |
| 8/15/2016 | [Updated 64-bit Visual Studio project for VS2015](http://kipirvine.com/asm/examples/Project64_VS2015.zip) |
| 5/8/2015 | [Example programs and link library source code for the Seventh Edition](http://kipirvine.com/asm/examples/Irvine_7th_Edition.msi) |
| 12/20/2012 | [Example programs and link library source code for the Sixth Edition](http://kipirvine.com/asm/examples/Irvine_6th_Edition_VS2012.msi) |
| 12/11/2011 | [Example programs and link library source code for the Sixth Edition](http://kipirvine.com/asm/examples/Irvine_6th_Edition_VS2010.msi) |
| 12/16/2009 | [Curtis Wong's Linux/NASM port of the Irvine32 library at SourceForge.net](http://along32.sourceforge.net/) |

Editions prior to the 6th Edition are no longer supported here. Instructions for downloading example files: For files having a .zip extension, right-click the appropriate file's hyperlink and save to your computer's download folder. Next, create a new folder named c:\Irvine and extract the zip file into this new folder.

If the file has an .msi extension, it is a Microsoft installer file. After downloading this type of file, double-click to launch the Microsoft installer.