COIS 2300H Lab 8

Today we are going to build a simple assembly program (using visual studio).

The goal here is not to have you writing x86 assembly code of any great value, that’s well beyond the scope of 1 hour tutorials. The goal is to get you to start to see the link between hardware and software (more on this for the remainder of the course), and understand the basic principles of debugging in assembly. A program you write in any high-level language is compiled into a low level assembly language, the available instructions in assembly depend on what the hardware implements. While ARM is hugely popular, and has a very similar instruction set to MIPS, x86 remains hugely popular, and has a nightmarishly large instruction set.

For these tutorials you will need Visual Studio of some sort. Every version is slightly different but Kip Irvine (of kipirvine.com) has written a number of books on x86 assembly programming and provides tutorials for each version of visual studio on his website.

I tested this on VS 2017 and it works fine, here are guides for older versions of Visual Studio

VS 2015  [https://web.archive.org/web/20190823192330/http://kipirvine.com/asm/gettingStartedVS2015/index.htm](https://web.archive.org/web/20190823192330/http:/kipirvine.com/asm/gettingStartedVS2015/index.htm)

VS 2012 [https://web.archive.org/web/20190807141151/http://kipirvine.com/asm/gettingStartedVS2012/index.htm](https://web.archive.org/web/20190807141151/http:/kipirvine.com/asm/gettingStartedVS2012/index.htm)

[http://kipirvine.com/asm/gettingStartedVS2015/index.htm - for VS 2015](http://kipirvine.com/asm/gettingStartedVS2015/index.htm%20-%20for%20VS%202015), specifically we are way down “creating a project from scratch”

[http://kipirvine.com/asm/gettingStartedVS2013/index.htm - for VS 2013](http://kipirvine.com/asm/gettingStartedVS2013/index.htm%20-%20for%20VS%202013)

<http://kipirvine.com/asm/gettingStartedVS2012/index.htm> This is for VS 2012 which the lab machines have. Some students have had success and some haven’t… following the same steps. I’m working on it, but sick, so you success may vary.

<http://kipirvine.com/asm/gettingStartedVS2010/index.htm> -For VS 2010

Etc.

As with last week we are going to make a Visual Studio project as though it’s C++ but we’re not actually going to write any C++ code. (C# compiles to a .NET framework first then to x86 so there’s an extra layer of instructions getting added, and C++ supports the full C language so we can write simple programs).

This is basically copied from the kipirvine VS 2015 tutorials

# 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 (This section in Gray is optional or not available in all versions of VS)**

Visual Studio supports 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.

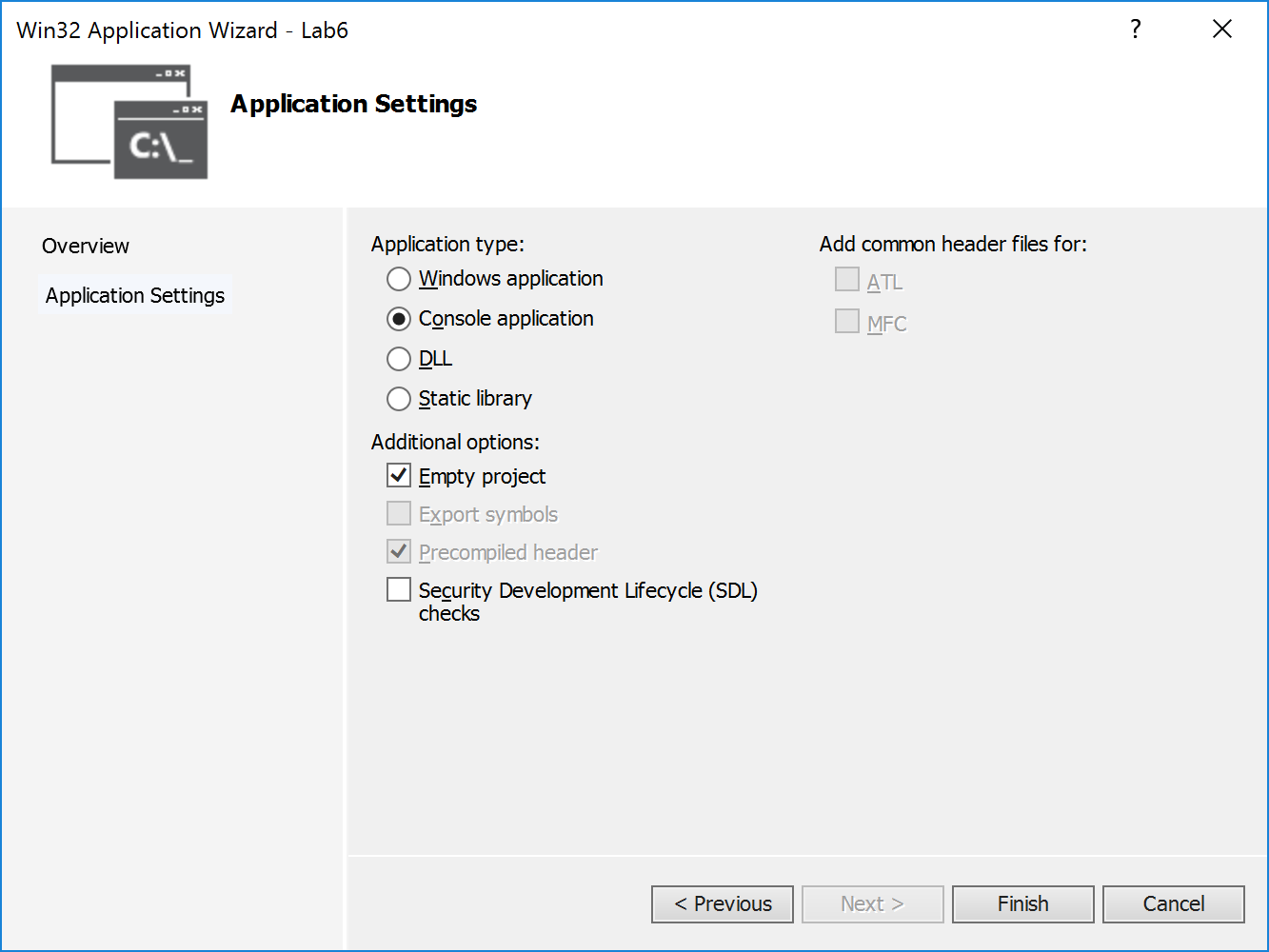
## Tutorial: Building a 32-Bit Assembly Language Program

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

Create a new C++ console application project. Make it an empty project

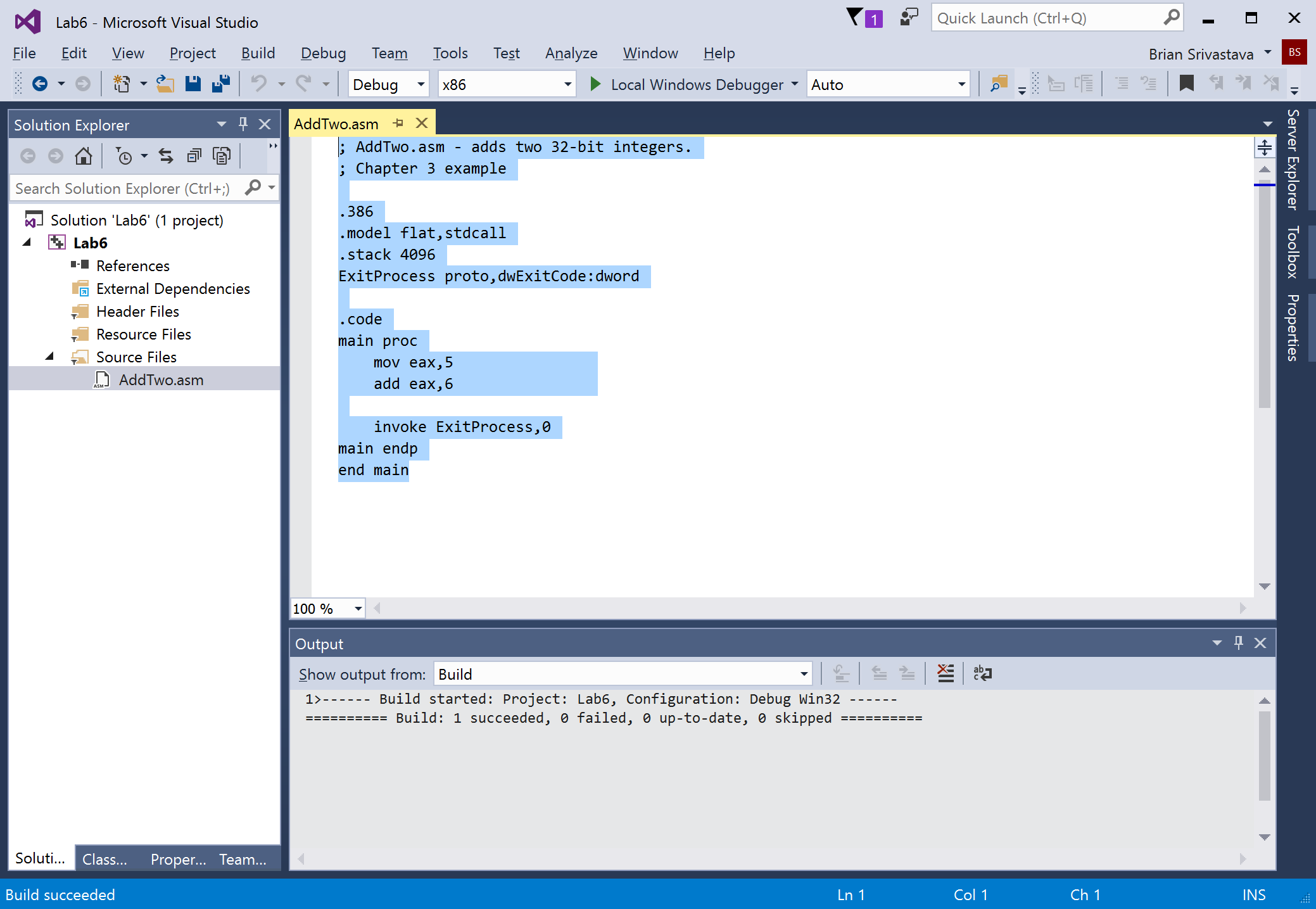


In the visual studio project directory create a file named AddTwo.asm.

In Visual Studio right click on the project (Lab6 in the screenshot below) and select add existing item, and select the file you just created, which is empty.

Paste in the following assembly code.

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

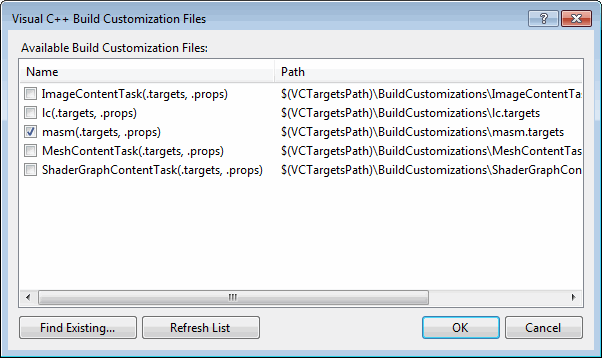


Try and build your program (it should work like the screenshot above).

Try and run it and…. It doesn’t work. There’s no executable file to run.

That’s because we need to actually compile the asm file. Only about 7 more steps (yes really)

Next, right click your project name (on the screenshot above that is “**Lab6”**), select **Build Dependencies**, then 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.

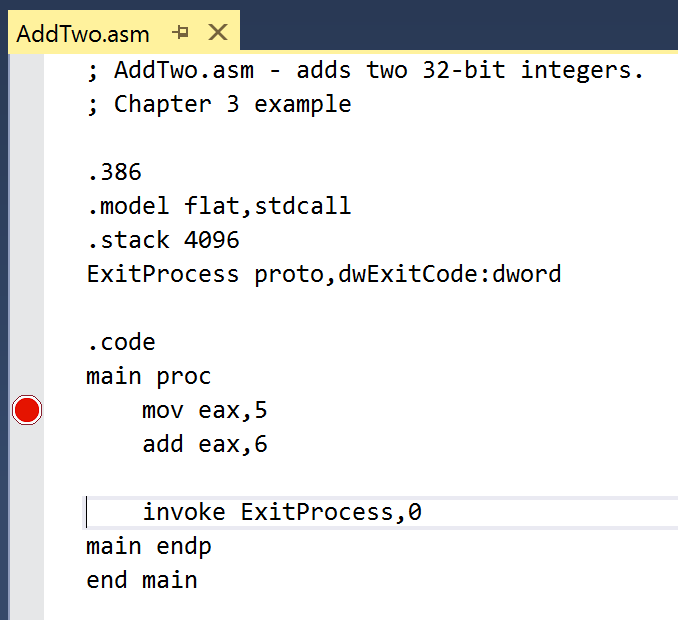


You're almost ready to set the assembler properties. But first, you must help Visual Studio associate source code files having the .asm extension with the Microsoft Macro Assembler. To do this, right-click the .asm file in the Solution Explorer window, select Properties from the popup window, click on Item Type in the right-hand pane, scroll down the list, and select Microsoft Macro Assembler. Click OK to close the dialog window.

At this point we deviate from the tutorial on the web because we’re not depending on any of the actual examples from Irvine.

At this stage you should be able to compile your program. It doesn’t actually do anything except add two numbers together.

Set a breakpoint by click on the vertical bar beside your program, set the breakpoint at the move eax, 5 l



Run your program by pressing F10, and keep pressing F10…. Notice that you uh… still can’t see anything

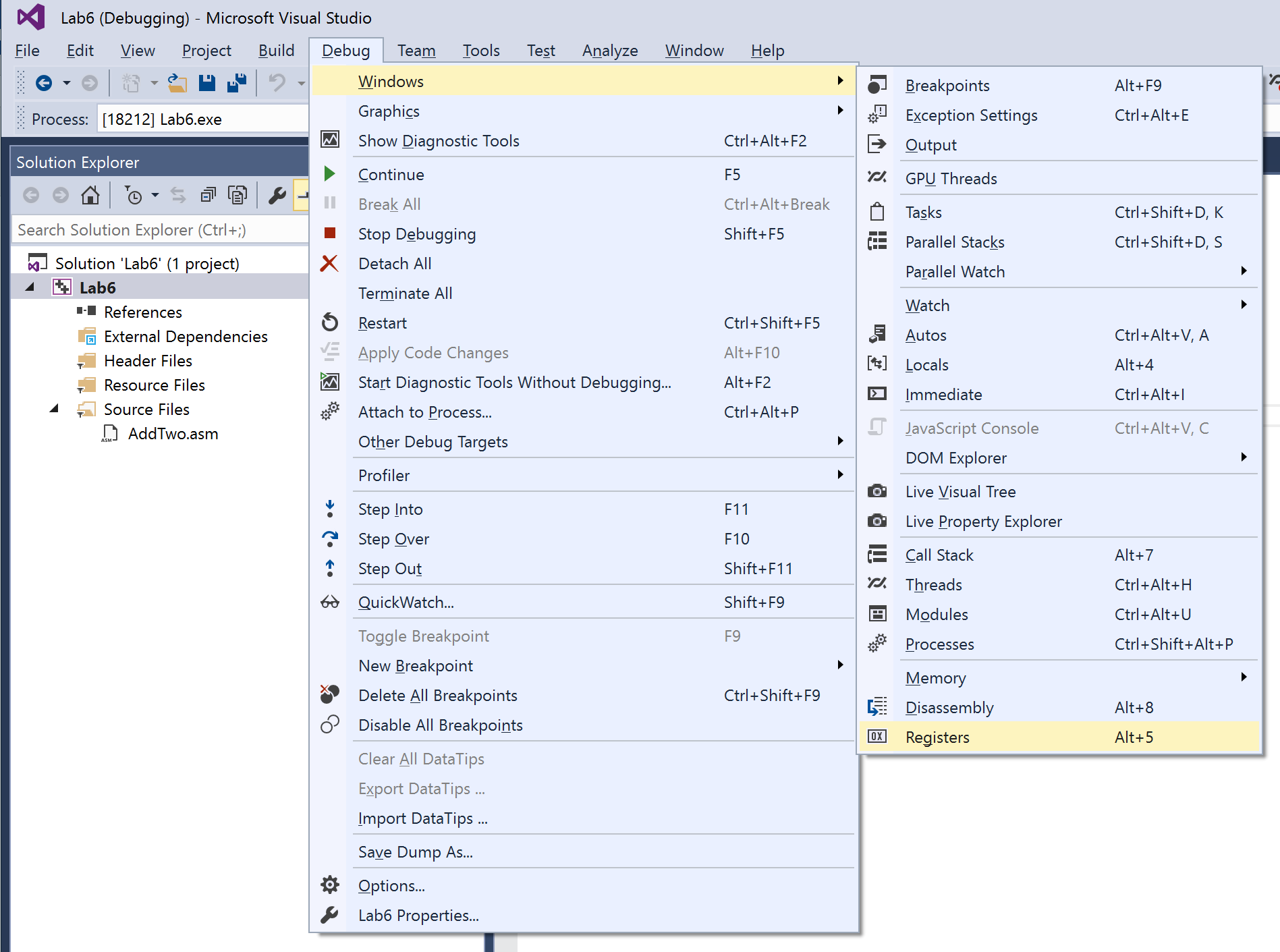
Link 221 Error: you need to set the properties of the project to use the subsystem for windows. (Depending on exactly how you created this project will determine if this is set when it’s created or not)

One last thing before you do something interesting on your own

#### 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.

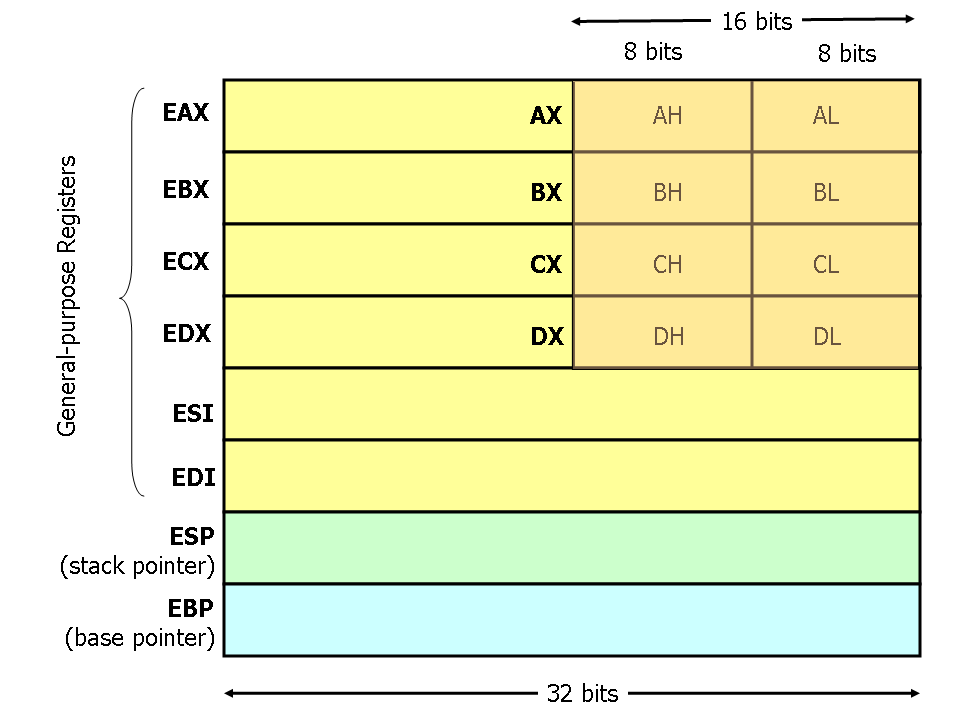
You can interrupt a debugging session at any time by selecting *Stop Debugging* from the Debug menu. You can do the same by clicking the maroon-colored square button on the toolbar. To remove a breakpoint from the program, click on its red dot to make it disappear.



And select registers.

Notice now you have a window that tells you the (hexidecimal) value of each register. If you run your program multiple times notice that the EAX register starts at a different value each time.

# A very simple guide to x86 Registers



Source: <http://www.cs.virginia.edu/~evans/cs216/guides/x86.html>

Ok so there is one important part to that, which you’re going to see, notice in the EAX register there are two 8 bit parts and the end AH and AL (A register high, A register Low). This is an example of an instruction option that is quite a bit different from ARM/MIPS

Copy the following code into your ASM file

|  |
| --- |
| ; S; Copying a String (CopyStr.asm)  ; This program copies a string.  .386  .model flat,stdcall  .stack 4096  ExitProcess PROTO, dwExitCode:dword  .data  source byte "Sri",0  target byte SIZEOF source DUP(0),0  .code  main proc  mov esi,0 ; index register  mov ecx,SIZEOF source ; loop counter  L1:  mov al,source[esi] ; get a character from source  mov target[esi],al ; store it in the target  inc esi ; move to next character  loop L1 ; repeat for entire string  invoke ExitProcess,0  main endp  end main |

Change the line for “source byte “Sri”, 0” to have your first name rather than Sri

Place a breakpoint on the line mov esi, 0.

**Run your program several times and record what happens to the EAX register (Try 5 runs) and show us it all running**

**Again the point of this lab is just to see how you could write an actual assembly program if you knew how and wanted to, not to make you actually do it. (Think of this more as an active participation in a lecture than a skill you need to actually develop)**