

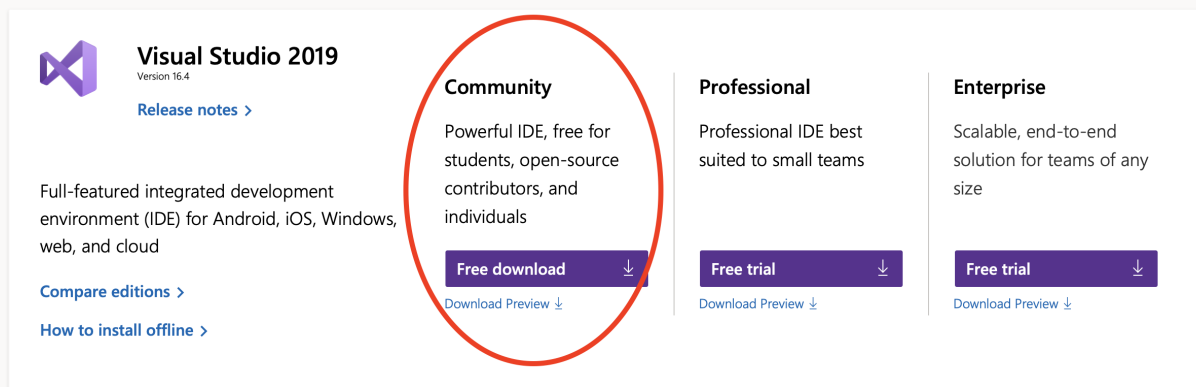
Installing Visual Studio Community 2019 for C++

Note: If you have a mac, you will need to dual boot your device so that you can run the Windows operating system. Microsoft recently came out with Visual Studio for macs, but this will not allow you to run the assembly programs for the class - you need to be using Windows. If you have never done this before, the following guide can offer some assistance: <https://www.laptopmag.com/articles/dual-boot-windows-os-x-mac>

The following steps demonstrate how to install Visual Studio 2019 onto your own Windows machine.



1. Go to: [Download Visual Studio 2019 for Windows](#) and choose *Community*.



Downloads





Visual Studio 2019
Version 16.4
[Release notes >](#)

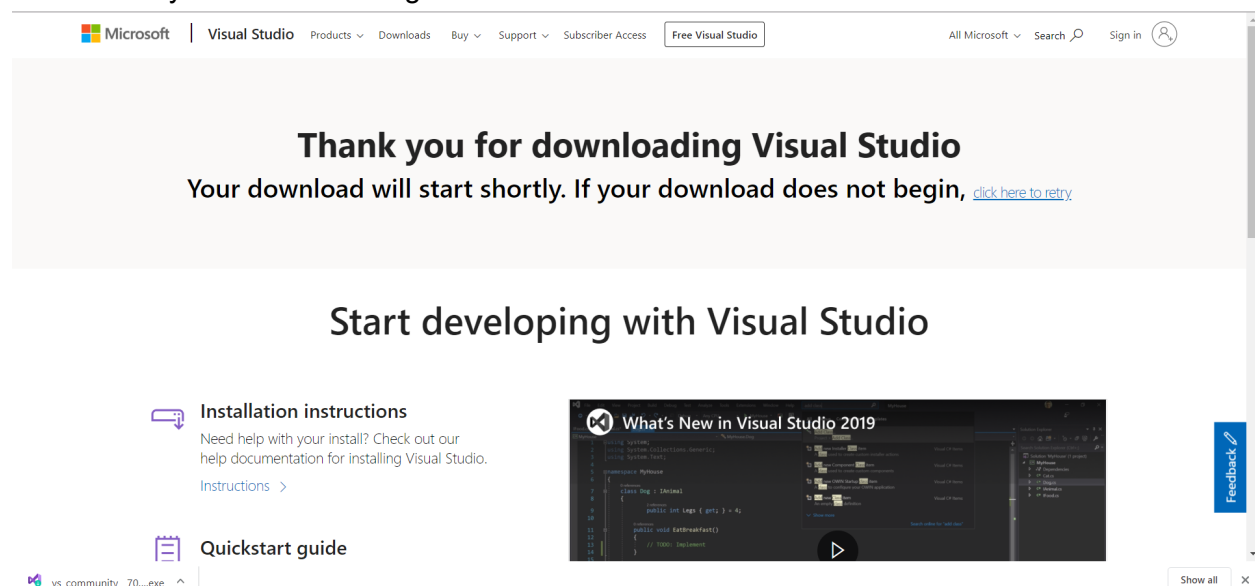
Full-featured integrated development environment (IDE) for Android, iOS, Windows, web, and cloud
[Compare editions >](#)
[How to install offline >](#)

Community
Powerful IDE, free for students, open-source contributors, and individuals
[Free download](#) 
[Download Preview](#) 

Professional
Professional IDE best suited to small teams
[Free trial](#) 
[Download Preview](#) 

Enterprise
Scalable, end-to-end solution for teams of any size
[Free trial](#) 
[Download Preview](#) 

2. Clicking on the **Free download** button will take you to the following page, and a .exe file will automatically start downloading.



Thank you for downloading Visual Studio
Your download will start shortly. If your download does not begin, [click here to retry](#).

Start developing with Visual Studio

Installation instructions
Need help with your install? Check out our help documentation for installing Visual Studio.
[Instructions >](#)

Quickstart guide

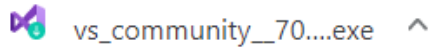
What's New in Visual Studio 2019

vs_community_70...exe

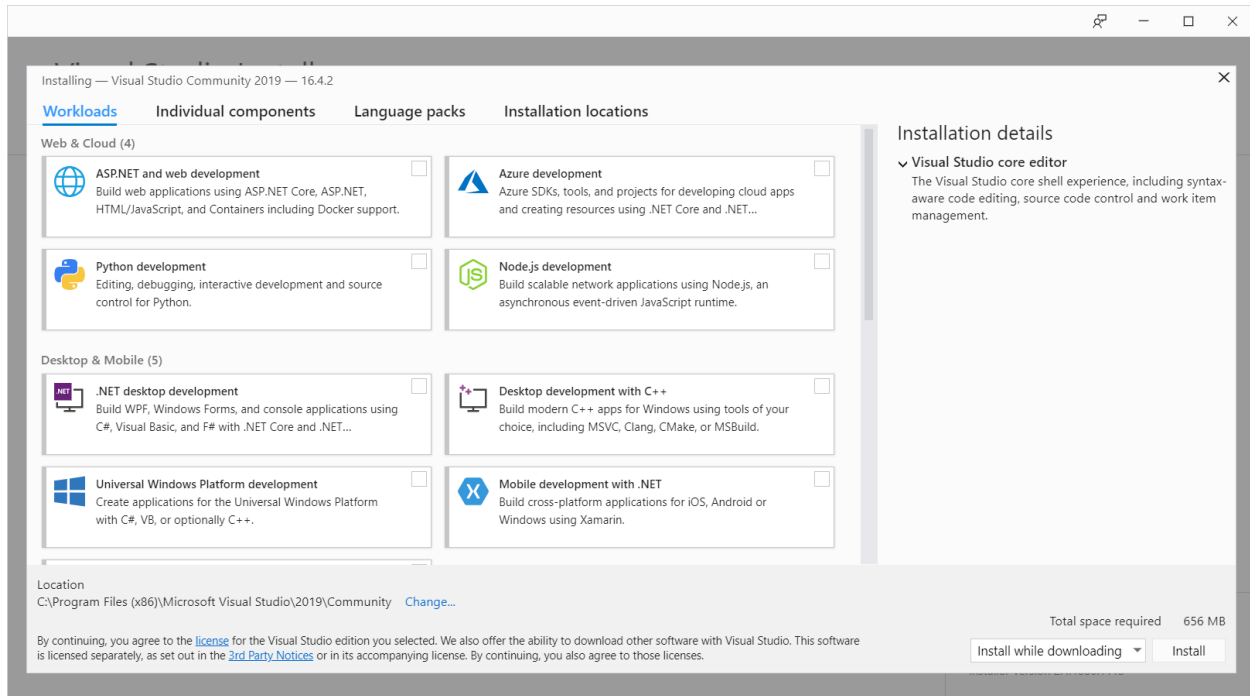
[Feedback](#)

Show all X

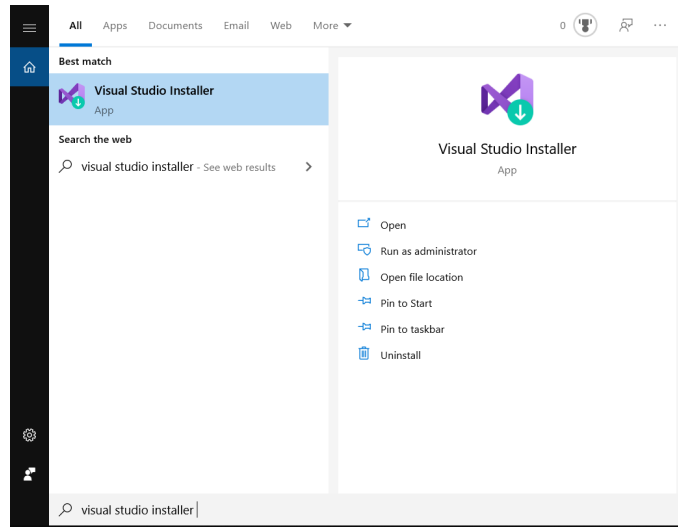
3. Once the .exe finishes downloading, click on it to launch the installer.



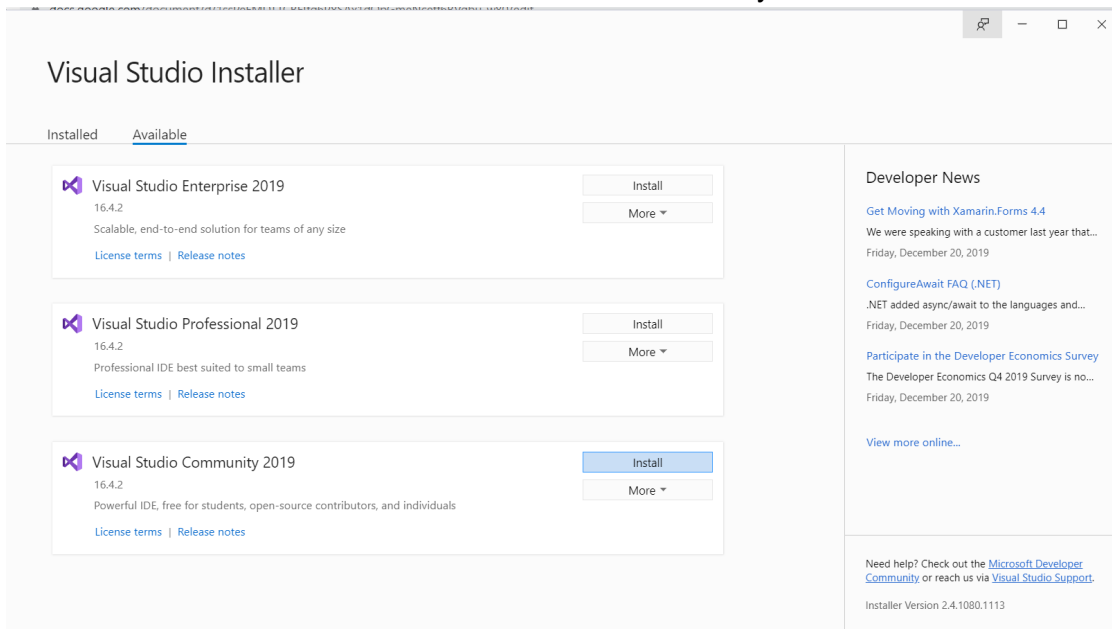
4. When the installer finishes, the **Visual Studio Installer** should automatically pop up and you should see a screen similar to the one below.



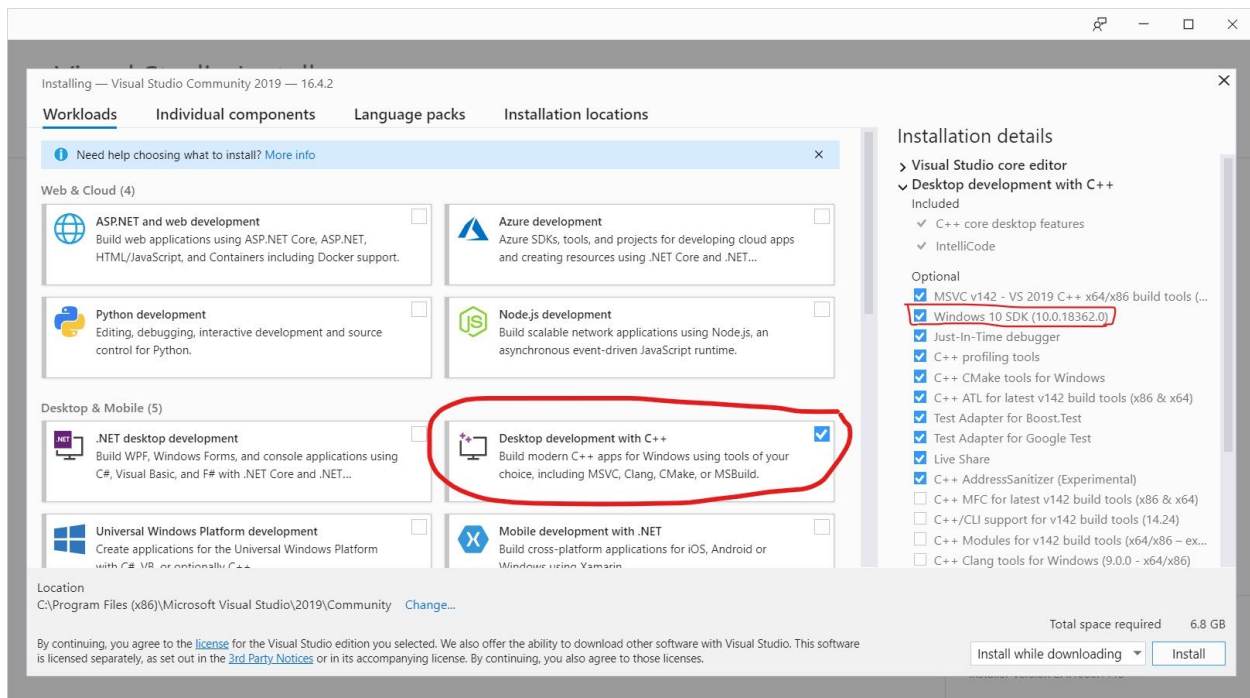
If, for whatever reason, the installer does not automatically pop up, search for **Visual Studio Installer** in the start menu search bar to launch the installer.



And then click the **Install** button in the Visual Studio Community 2019

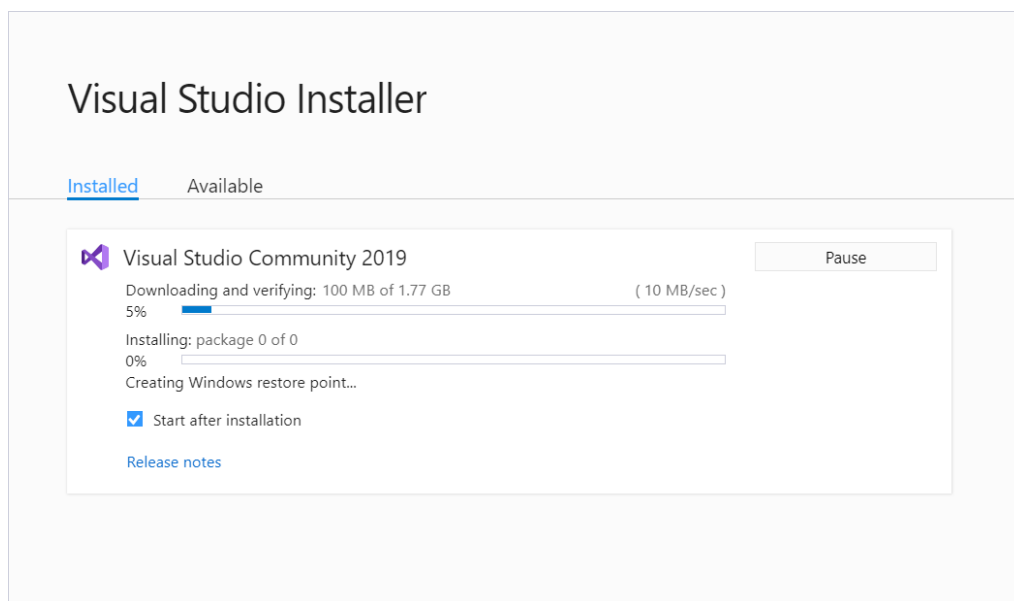


5. Make sure that *both* **Desktop development with C++** and a **Windows SDK** are selected. The following example comes from a computer using Windows 10, so the Windows 10 SDK is selected.

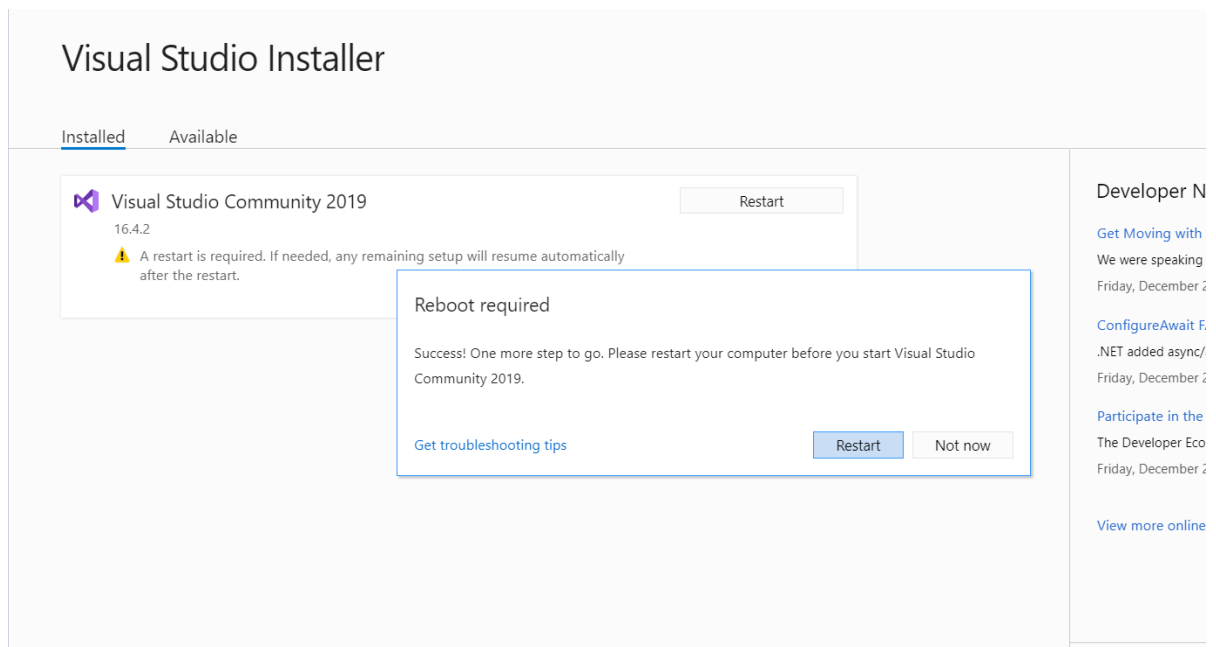


Once you are sure both are selected, click **Install**.

6. You will now see Visual Studio Community 2019 begin downloading and installing onto your computer. This will take some time.

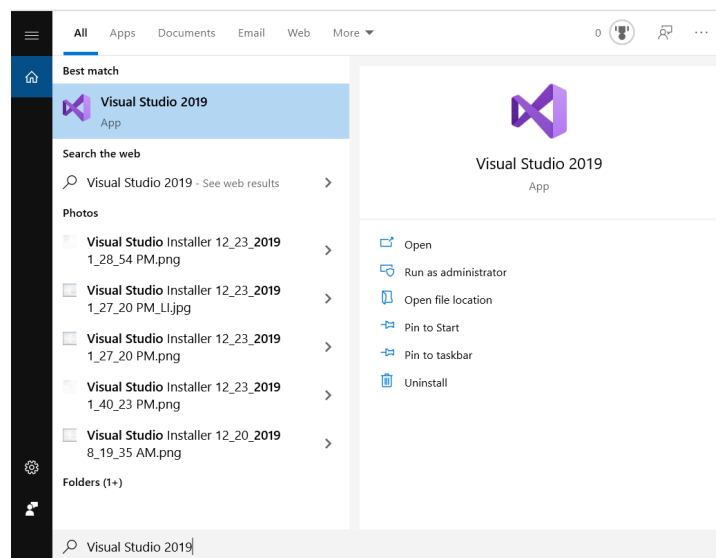


7. Once the installer finishes, you will see a message indicating that a reboot is required.

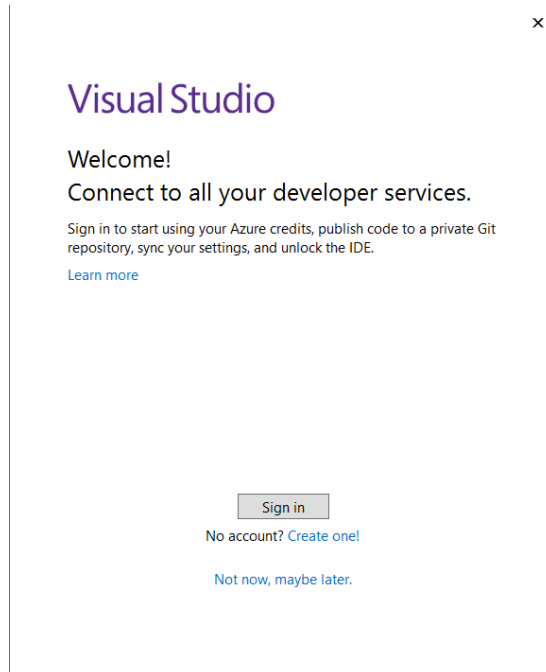


Click on **Restart**

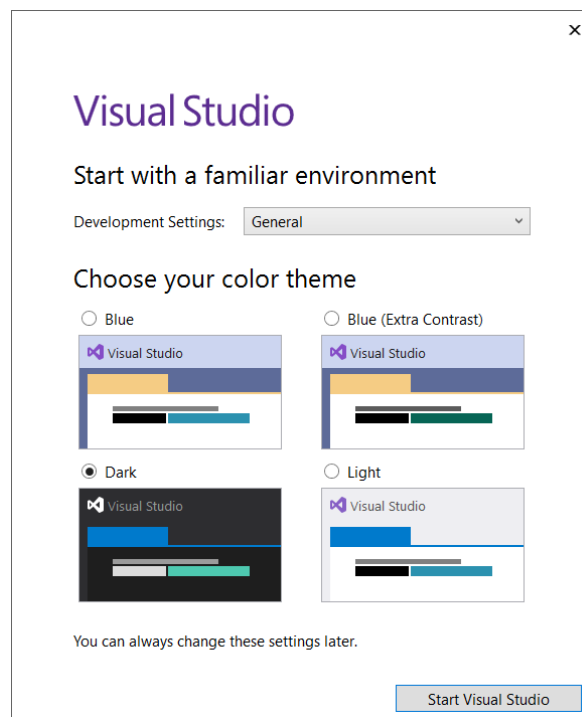
8. After your machine restart, search for **Visual Studio 2019**, and click on it to launch VS



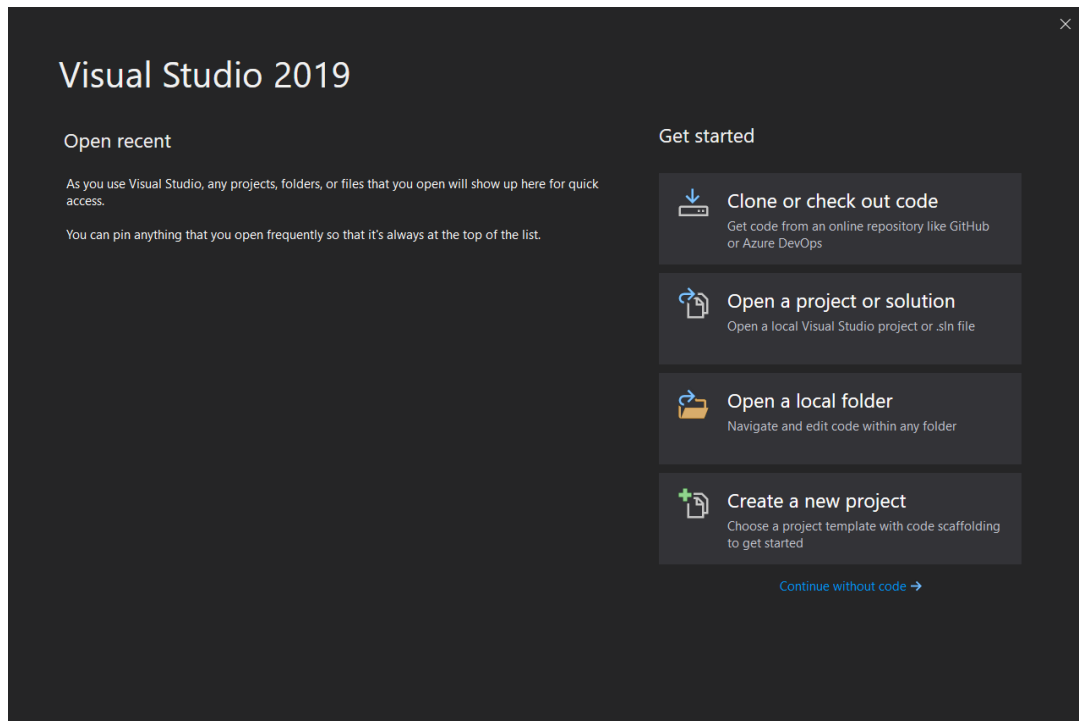
The *first time* you launch Visual Studio, you will be prompted to sign in. If you have a Microsoft account, you can sign in; otherwise, choose **Not now, maybe later**



You will also be asked to choose a color theme. Pick one and click **Start Visual Studio**. You can always change the color theme later in the settings.



10. If you see the Visual Studio homepage, this means that you have successfully installed Visual Studio Community!

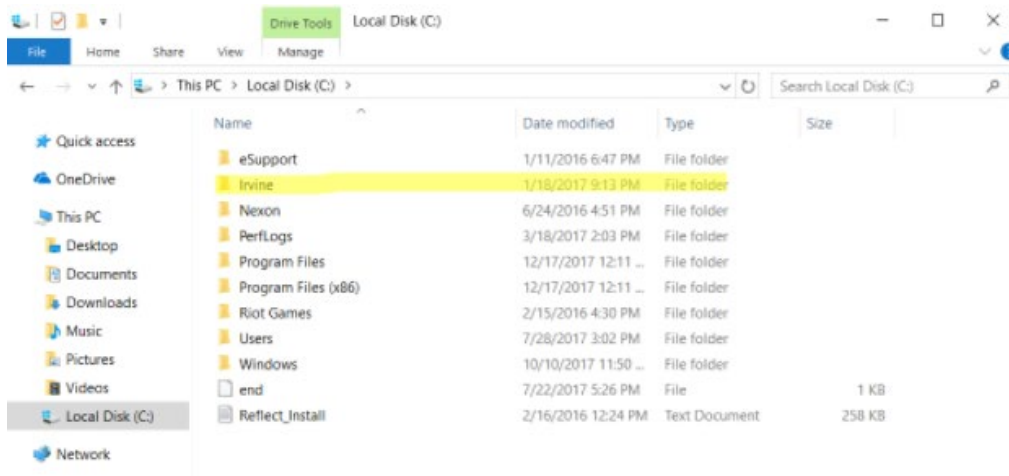


Running *hello.asm*

Now that Visual Studio is successfully installed on your device, let's move on to the standard hello world program.

First, you will need to download the Irvine library. Clicking the following link will start the download immediately: <http://asmirvine.com/gettingStartedVS2019/Irvine.zip>

Once the .zip file finishes downloading, extract the folder and place it onto your C: drive. (Make sure that it is on your C: drive or your programs will not run!).



Next, download the Project file

<https://drive.google.com/drive/folders/1zGLAUe5pCm7WEE95TUdIkYeMf6qi5tWK?usp=sharing>

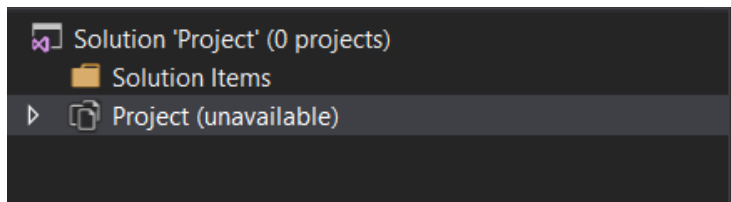
The project file does not to be in any particular location, but you will be continually using it to complete your assignments throughout the quarter, so I recommend you place it somewhere easily-accessed.

If you are on the school computers, the Project File is located in the C: drive in the Irvine folder. Copy and paste the folder onto your Z: drive so that it does not get deleted. (Remember that the school computers will delete anything on your desktop once you log out).

Open the file and click on the solution (.sln). If you are prompted, choose to open the file with Visual Studio. In the below image, the .sln file is highlighted.

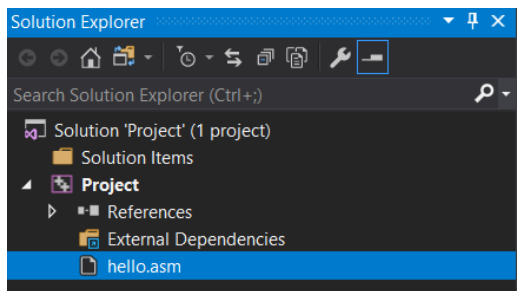
Name	Date modified	Type	Size
hello	8/15/2017 11:36 A...	ASM File	1 KB
Project.lst	8/15/2017 11:36 A...	VisualStudio.lst.12.0	27 KB
Project	8/15/2017 11:29 A...	SQL Server Compa...	384 KB
Project	8/15/2017 11:29 A...	Microsoft Visual St...	2 KB
Project.vcxproj	8/15/2017 11:36 A...	VCXPROJ File	5 KB
Project.vcxproj.user	8/15/2017 11:29 A...	VisualStudio.user.5...	1 KB

Once Visual Studio has opened the file, look at the solution explorer on the left-hand side of the screen. If you see something like the following:

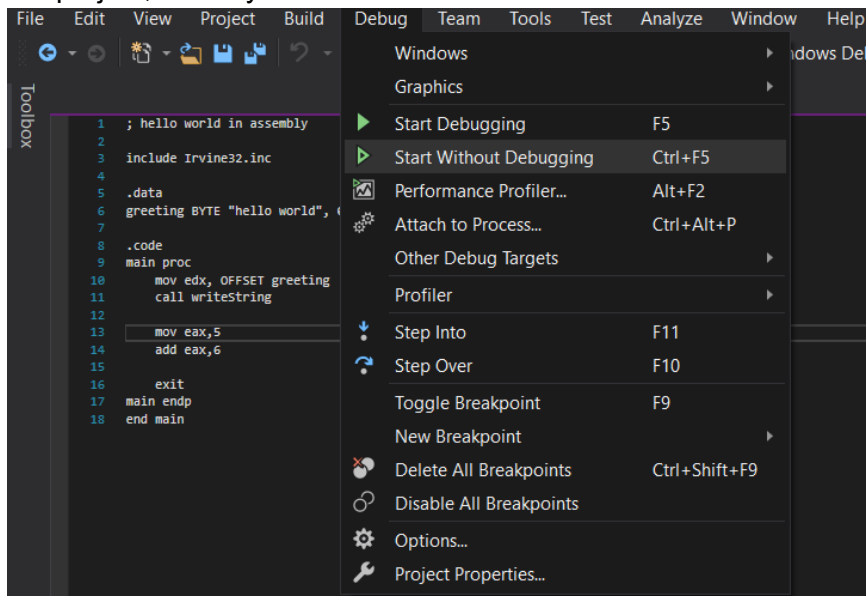


Right-click on the “Project (unavailable)” and select “Install Missing Feature(s)”. Install whatever you are missing and try again. (This means you probably forgot to click on “desktop development with c++”)

Once the missing features finish installing, look at the solution explorer and double click on “hello.asm”



Once you see that the file has opened up, go to “Debug” and click on “Start Without Debugging” (or press Ctrl + F5). If you get a message saying the project is out of date and a prompt to build the project, select yes.



At this point, you should see the console pop up and display “hello world”. If this happened, then you will be able to run assembly programs for the class on your device!

If you encounter any errors, feel free to post a message on Canvas for help.

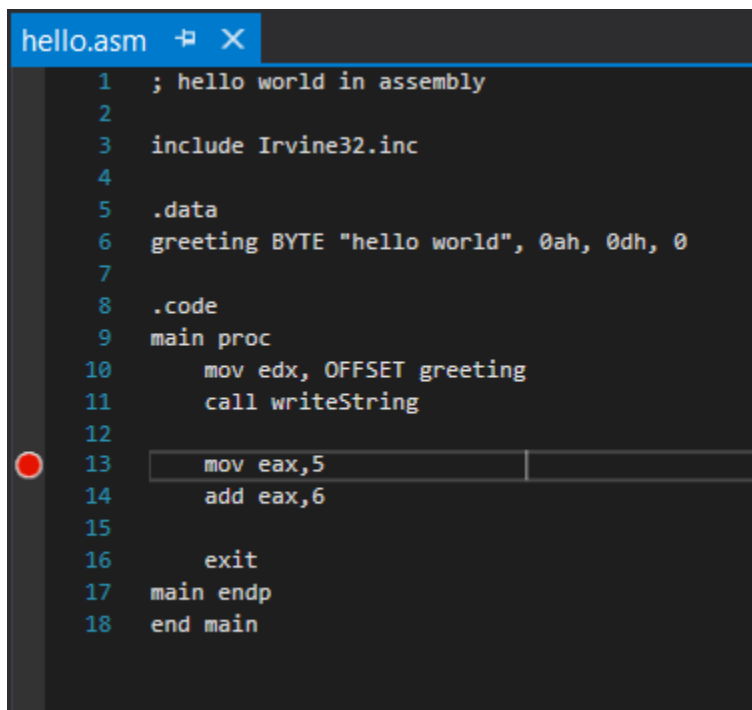
Throughout the quarter, use the hello.asm file as a template for your assignments. The hello.asm file has several settings needed to make the program run correctly.

Debugging in Visual Studio

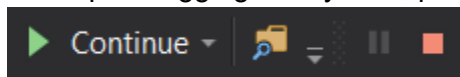
Visual Studio provides a debugger that is extremely useful for finding pesky errors in code. This section will cover some features of the debugger that you will use with your programs throughout the quarter.

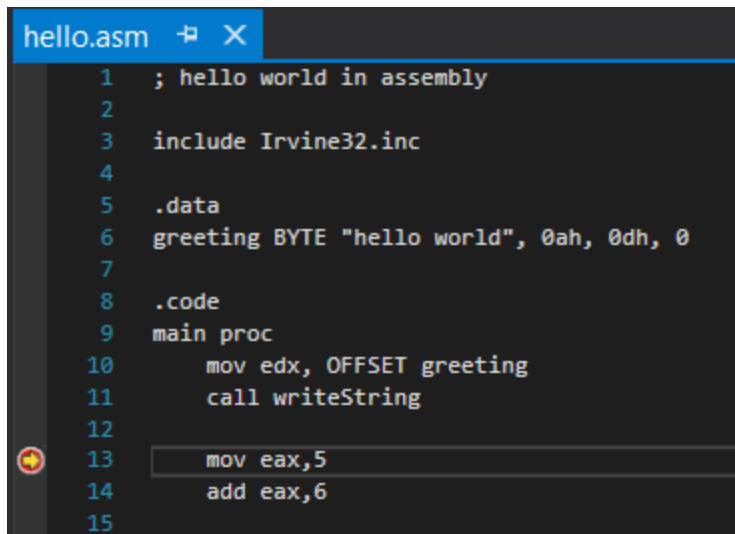
Setting a breakpoint

When debugging, we can set something known as a breakpoint. Setting a breakpoint will cause the program run up to the line where the breakpoint was set. To set a breakpoint, click on the bar to the left of the line numbers. Once you do this, you should see a red dot appear where you clicked.



Once you have set a breakpoint, you can go to “Debug” -> “Start Debugging” (or press F5). To stop debugging at any time, press the red square near the top of the screen.





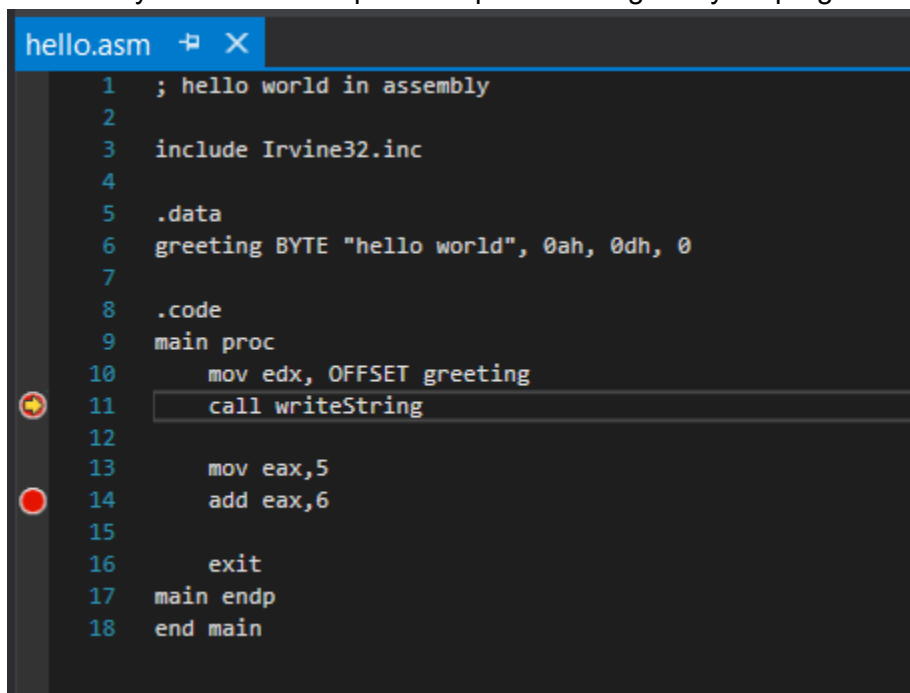
The screenshot shows a debugger window titled 'hello.asm'. The assembly code is as follows:

```
1 ; hello world in assembly
2
3 include Irvine32.inc
4
5 .data
6 greeting BYTE "hello world", 0ah, 0dh, 0
7
8 .code
9 main proc
10     mov edx, OFFSET greeting
11     call writeString
12
13     mov eax, 5
14     add eax, 6
15
```

A yellow arrow icon on the left margin points to line 13, indicating the next instruction to be executed.

This is what your screen should look like once you have set a breakpoint and started debugging. Notice that there is a yellow arrow pointing at line 13. The yellow arrow represents the *next* instruction to be executed (it has not yet been executed).

Note that you can set multiple breakpoints throughout your program.



The screenshot shows the same debugger window with two breakpoints set. The assembly code is as follows:

```
1 ; hello world in assembly
2
3 include Irvine32.inc
4
5 .data
6 greeting BYTE "hello world", 0ah, 0dh, 0
7
8 .code
9 main proc
10     mov edx, OFFSET greeting
11     call writeString
12
13     mov eax, 5
14     add eax, 6
15
16     exit
17 main endp
18 end main
```

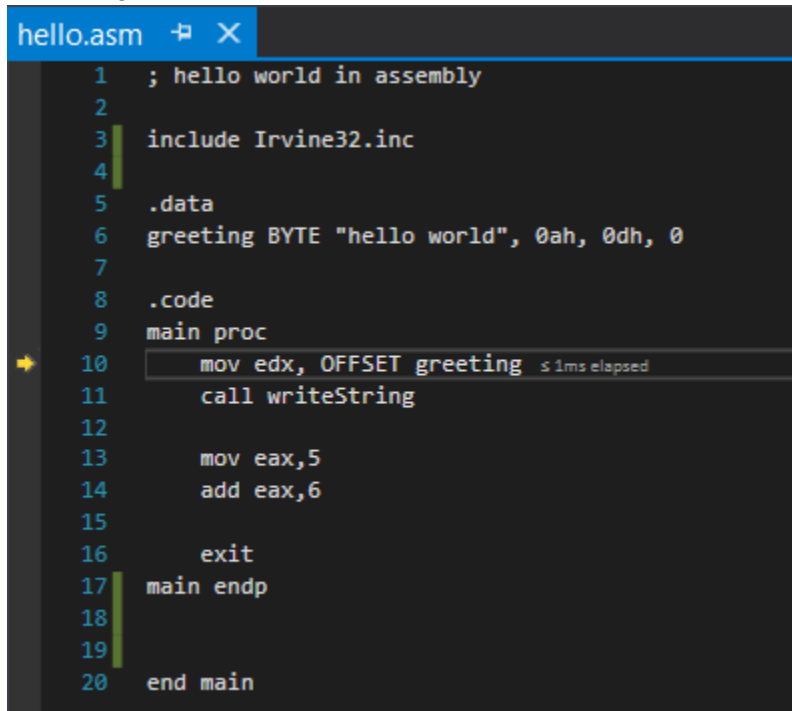
Two red circular icons on the left margin indicate breakpoints: one at line 11 and another at line 14.

Pressing F5 once will take me to the first break point. Pressing it once again will jump to the second.

Stepping into and stepping over

In addition to setting breakpoints, we can step into and over our code. This essentially lets us step through our code, line-by-line.

To Step over your code, go to “Debug” -> “Step Over” (or press F10). You will see a screen that says “Source not Available”. Press F10 again and you will once again see the yellow arrow indicating the next line that will be executed.



```
hello.asm
1 ; hello world in assembly
2
3 include Irvine32.inc
4
5 .data
6 greeting BYTE "hello world", 0ah, 0dh, 0
7
8 .code
9 main proc
10 mov edx, OFFSET greeting ≤ 1ms elapsed
11 call writeString
12
13     mov eax,5
14     add eax,6
15
16     exit
17 main endp
18
19
20 end main
```

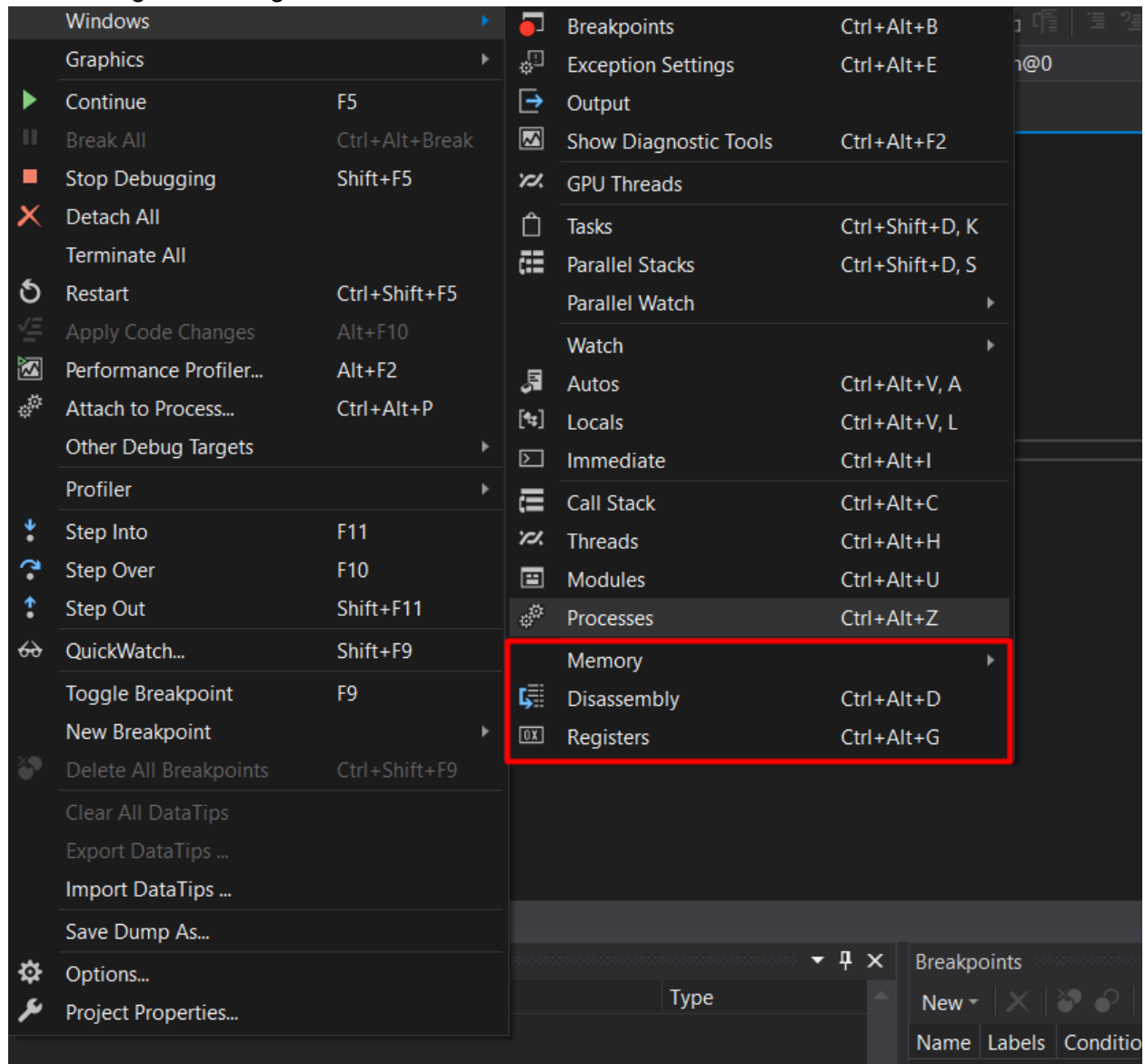
Every time you press F10, the next instruction will be executed. The difference between “Stepping Over” (F10) and “Stepping Into” (F11) is that stepping into will also step into procedures. If you have a procedure call in main and are using F10 to step through your program, the procedure call will be treated as a single instruction. If you use F11, you will see the arrow actually moves into the procedure and allows you to step through all instructions inside. This may not have much worth to you now, but it will be helpful to keep this in mind once you start creating your own procedures.

You are able to combine breakpoints with the step commands. For example, imagine I know that everything in my program up to line 13 works. I could then set a breakpoint at line 13 to have my program run up to that point, and then use F10 or F11 to step through the rest of my program.

Viewing Registers, Memory, and Flags

One of the main reasons why we want to debug our programs is so that we can see what each instruction is doing. Most instructions will modify registers or memory, so it is useful to be able to view these as they are being changed, line-by-line. To view the registers, memory, and flags, you **must** be in debugging mode.

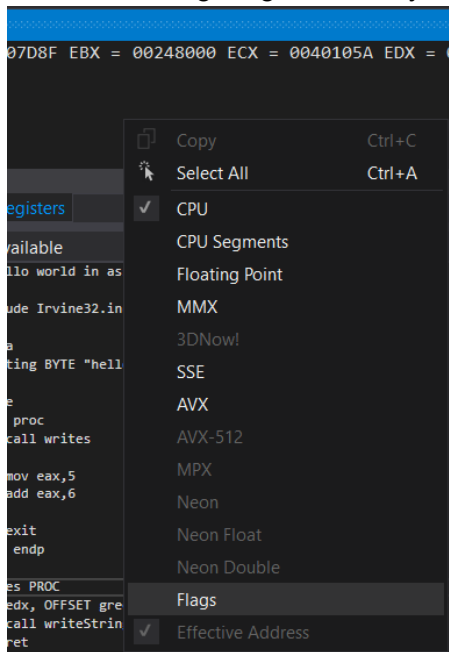
To enter debugging mode, either press F10, or set a breakpoint and press F5. Once you have done this, go to “Debug” -> “Windows”



Here, you can choose to view the registers and portions of memory.

To view the registers, all you need to do is click on “Registers”. Once you do this, you will see a bar appear above your code with the current state of the registers.

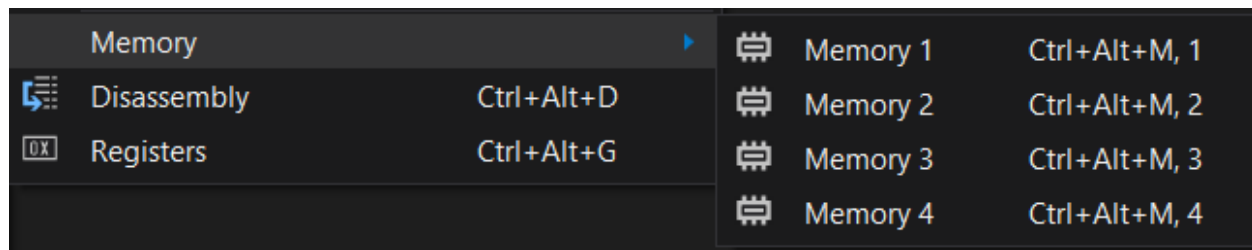
To view the flags, right-click anywhere on the registers window and select flags.



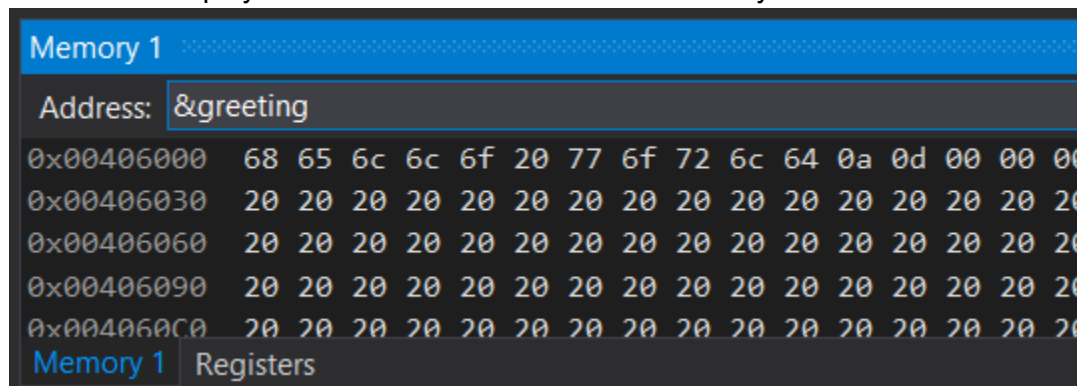
You should now see all the flags listed with the current state of each one. Here is a table for the flags.

Overflow	OV
Direction	UP
Interrupt	EI
Sign	PL
Zero	ZR
Auxiliary	AC
Parity	PE
Carry	CY

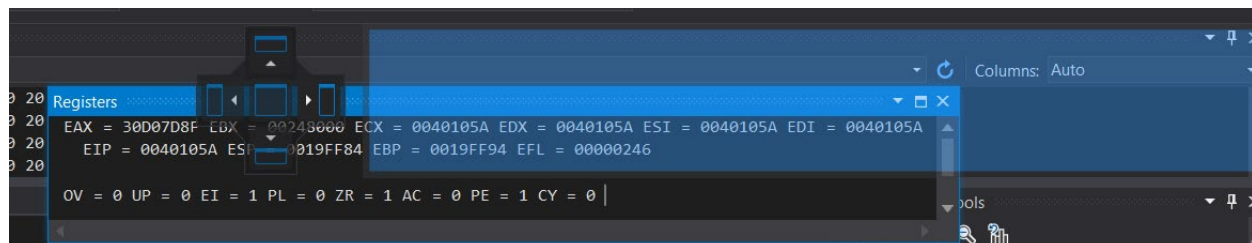
To view the contents of a variable in memory, go back to “Debug” -> “Windows” -> “Memory”



You can click on any memory you choose. Once you do, you will see a box appear. To view the contents of a specific variable, go to the address bar and type `&variableName` (the name you input here is case-sensitive). For example, if I wanted to view the contents of the “greeting” variable in the `hello.asm` file, I would type `&greeting`. Once you do, press enter and the window should now display the contents of that variable in memory.

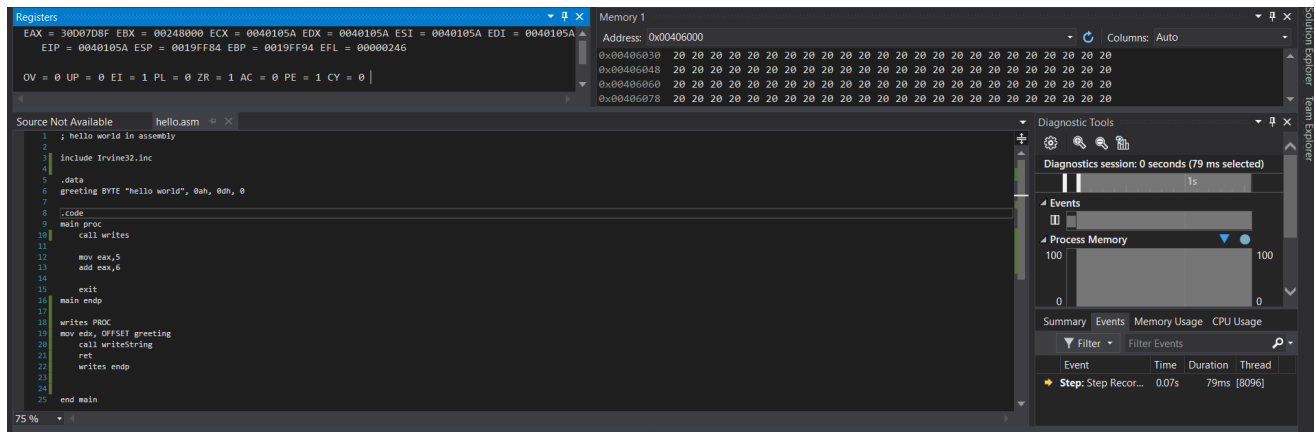


Note: You can rearrange the windows to appear in a format that easy for you to work with. To move a window, click and hold onto the top bar, then move it to the location where you want to place it on the pad that appears.



For example, in the above image, I wanted to move the registers window so I dragged the top bar (where it says “Registers”) to the right until I saw the pad appear (with the rectangles and arrows). Once I saw the pad, I put my cursor over the appropriate location where I wanted to place the window.

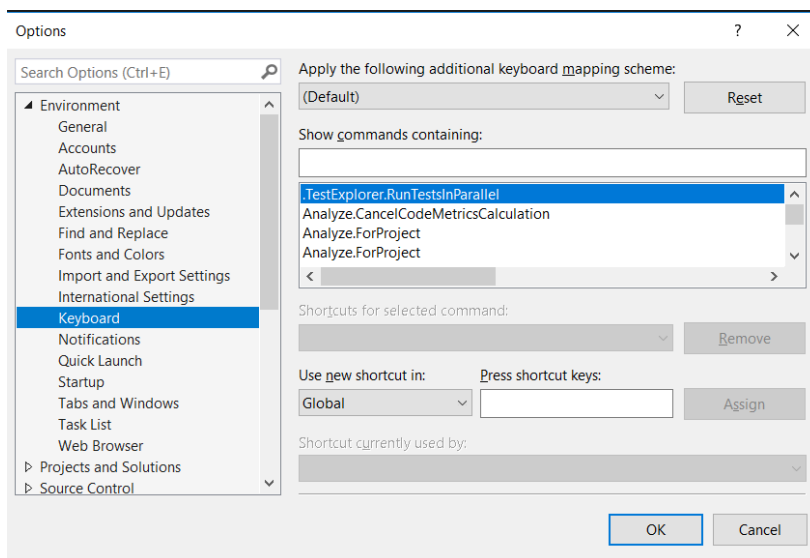
Once you do this, you can create a debugging screen that allows you to keep track of both the memory and registers.



There are several other debugging features that Visual Studio offers, but these are the ones that you will mainly be using.

Setting Hotkeys

Most computers will have hotkeys pre-set on the most commonly used commands. For example, the step-into command is bound to the F11 key by default. If for whatever reason you wish to change or add a hotkey, you can do so by going to “Tools” -> “Options” -> “Environment” -> “Keyboard”.



Once here, go to the “Show commands containing box” to find a certain command. Once you find it, click on it, then go to the “Press shortcut Keys” box and press the keys that you want to bind the command to. Usually, you want to enter a combination of ctrl and some key, or ctrl + shift + some key. Once you do this, press assign, and the hotkey will be set. Make sure that you do not override a hotkey that you are already using. You can check for this by looking at the “Shortcut Currently Used By” box at the bottom.

