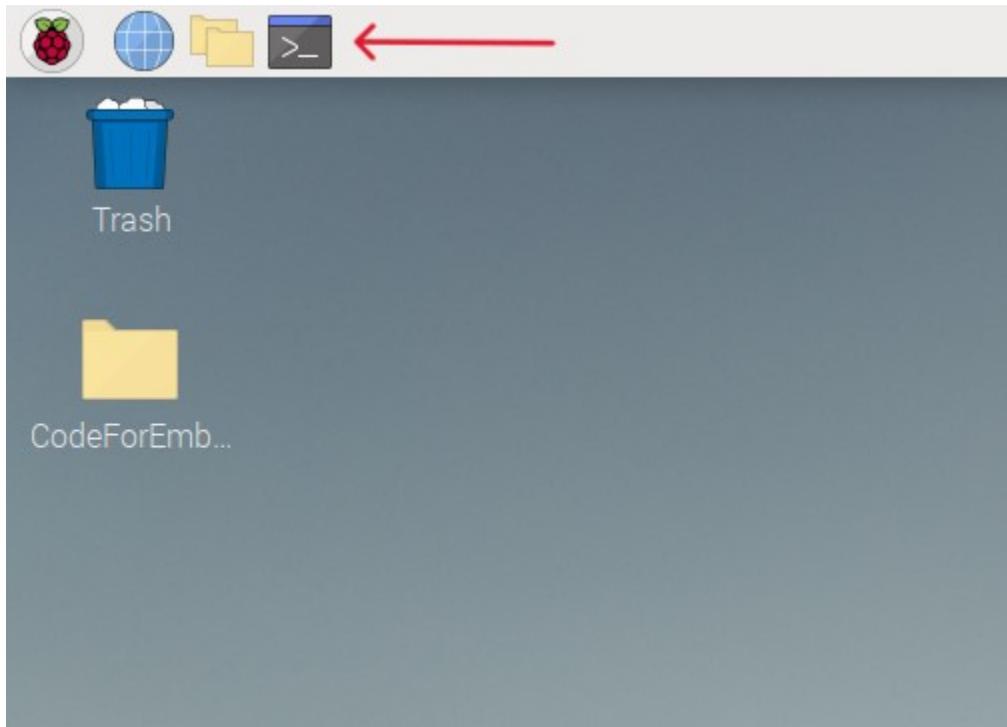


Assembly setup and hello world

To set up assembly you need an environment to code in, for this class I suggest using Visual Studio Code. To set up visual studio code on the raspberry pi is rather simple. The first step which should already be completed by this point is installing the raspberry pi OS and getting connected to the internet. Once connected to the internet you want to start with updating the Pi. This can be done through the terminal of the raspberry pi the icon can be located in the top left.



Once opened, type in the terminal box 'sudo apt update' once this is complete you should see something like what is in the second picture below.

```
cory@raspberrypi:~
```

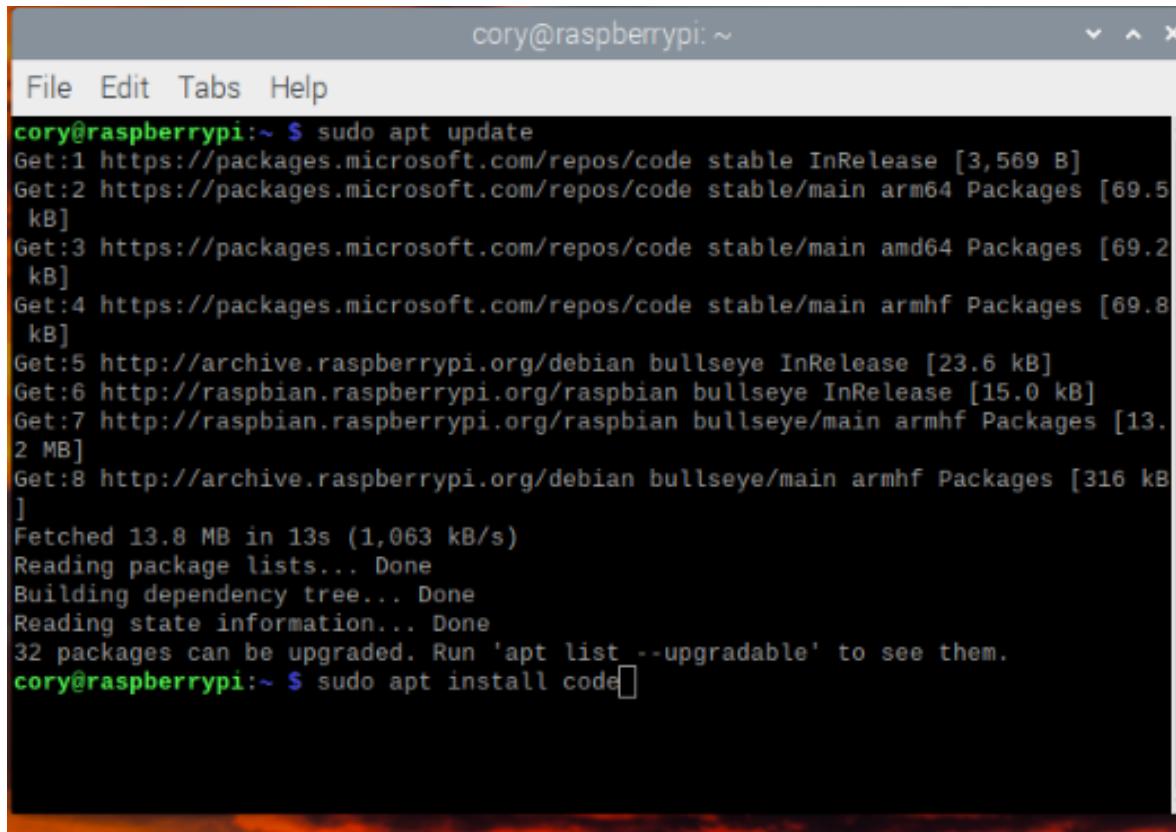
```
File Edit Tabs Help
```

```
cory@raspberrypi:~ $ sudo apt update
```

```
Get:1 https://packages.microsoft.com/repos/code stable InRelease [3,569 B]
Get:2 https://packages.microsoft.com/repos/code stable/main arm64 Packages [69.5 kB]
Get:3 https://packages.microsoft.com/repos/code stable/main amd64 Packages [69.2 kB]
Get:4 https://packages.microsoft.com/repos/code stable/main armhf Packages [69.8 kB]
Get:5 http://archive.raspberrypi.org/debian bullseye InRelease [23.6 kB]
Get:6 http://raspbian.raspberrypi.org/raspbian bullseye InRelease [15.0 kB]
Get:7 http://raspbian.raspberrypi.org/raspbian bullseye/main armhf Packages [13.2 MB]
Get:8 http://archive.raspberrypi.org/debian bullseye/main armhf Packages [316 kB]
Fetched 13.8 MB in 13s (1,063 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
32 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

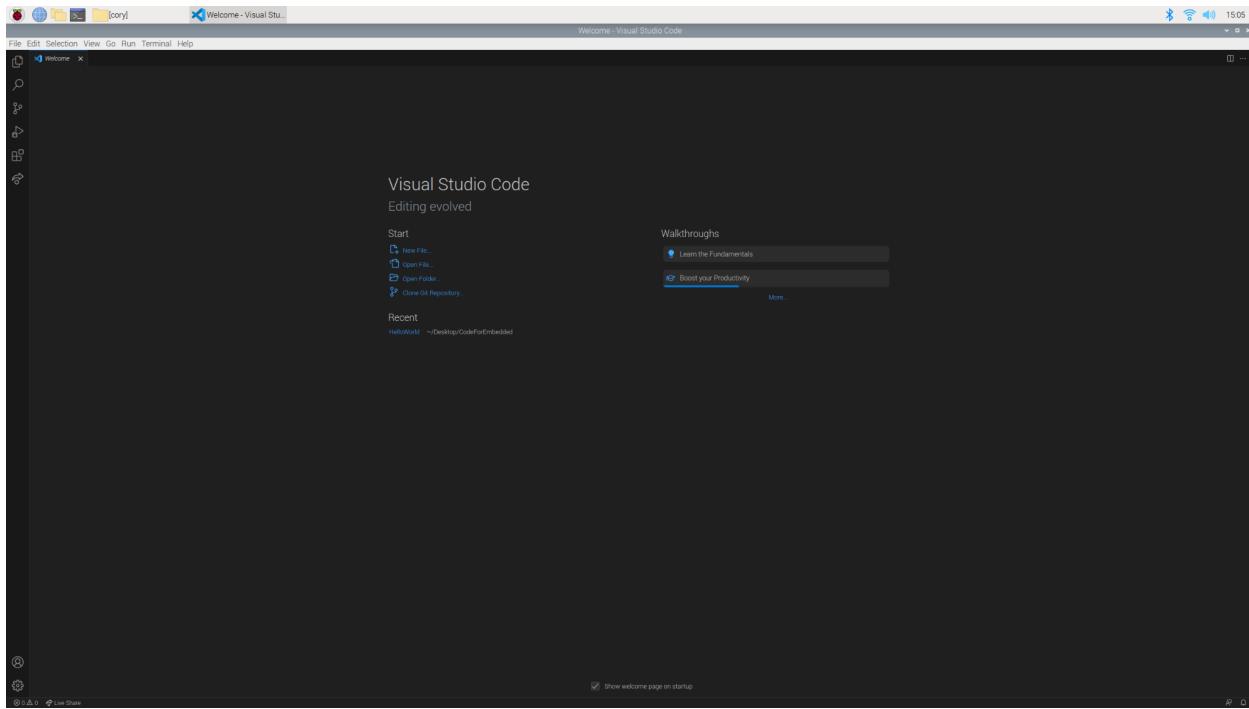
```
cory@raspberrypi:~ $
```

Once the Pi has been updated you will need to install visual studio code. To do this type in the terminal box again and type 'sudo apt install code'. This will install visual studio code.

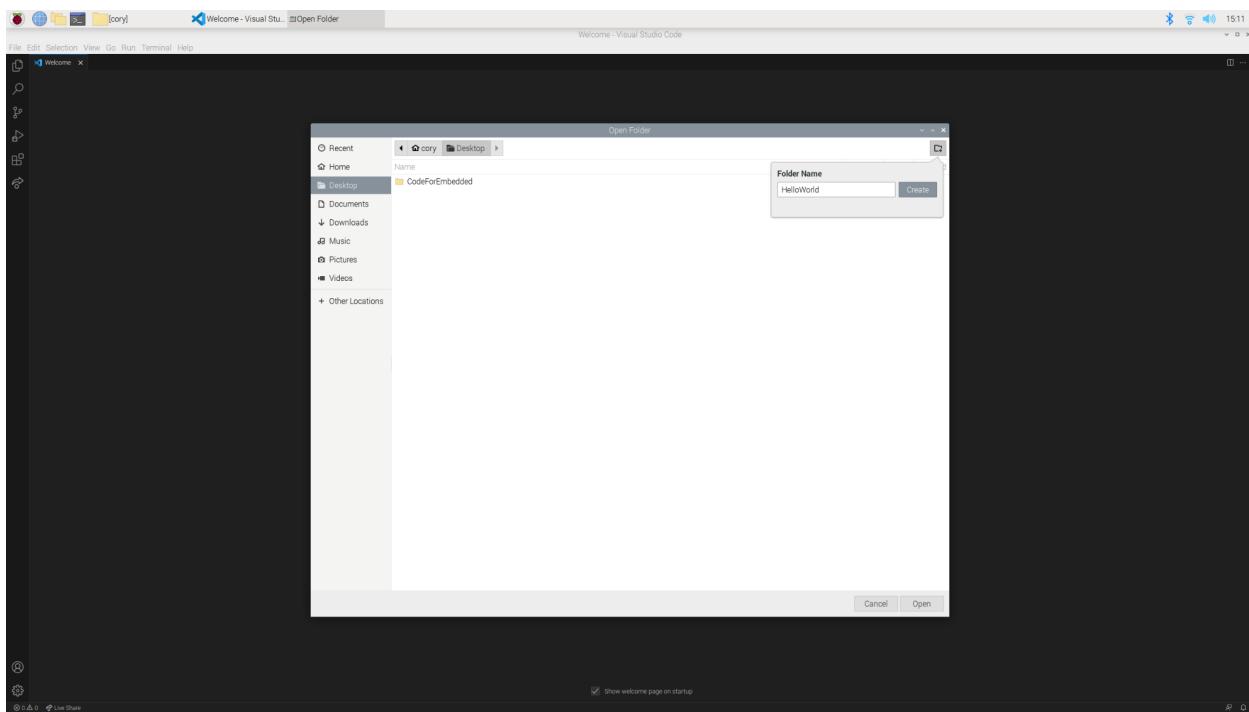
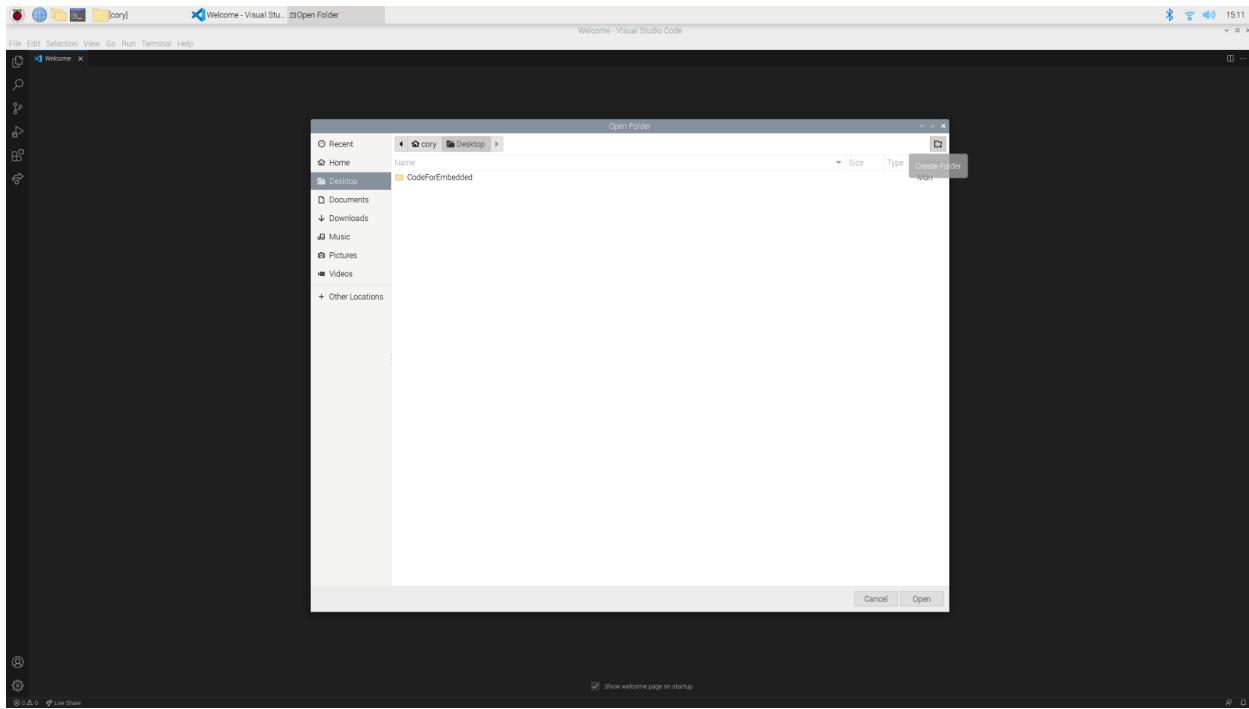


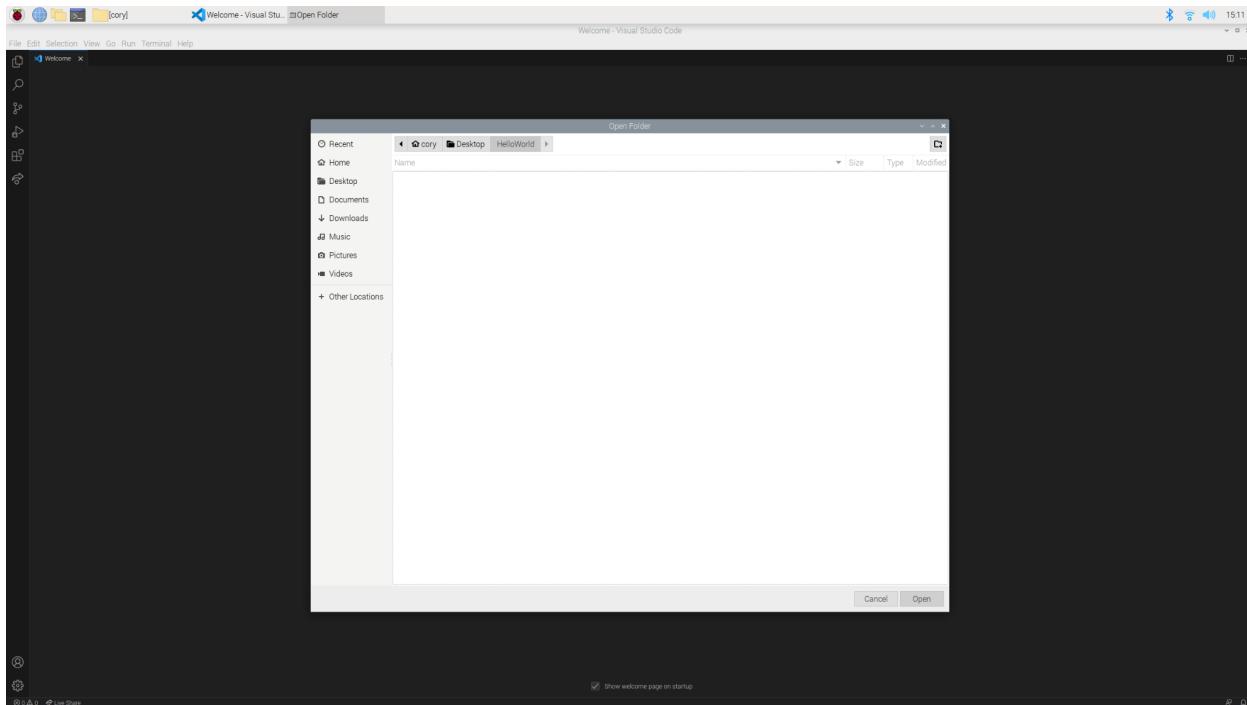
```
cory@raspberrypi:~$ sudo apt update
Get:1 https://packages.microsoft.com/repos/code stable InRelease [3,569 B]
Get:2 https://packages.microsoft.com/repos/code stable/main arm64 Packages [69.5 kB]
Get:3 https://packages.microsoft.com/repos/code stable/main amd64 Packages [69.2 kB]
Get:4 https://packages.microsoft.com/repos/code stable/main armhf Packages [69.8 kB]
Get:5 http://archive.raspberrypi.org/debian bullseye InRelease [23.6 kB]
Get:6 http://raspbian.raspberrypi.org/raspbian bullseye InRelease [15.0 kB]
Get:7 http://raspbian.raspberrypi.org/raspbian bullseye/main armhf Packages [13.2 MB]
Get:8 http://archive.raspberrypi.org/debian bullseye/main armhf Packages [316 kB]
Fetched 13.8 MB in 13s (1,063 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
32 packages can be upgraded. Run 'apt list --upgradable' to see them.
cory@raspberrypi:~$ sudo apt install code
```

To begin coding in assembly open visual studio code which you just installed. This can be found in the top left corner by clicking the raspberry icon, hovering over the programming icon and selecting visual studio code. This will open visual studio code.

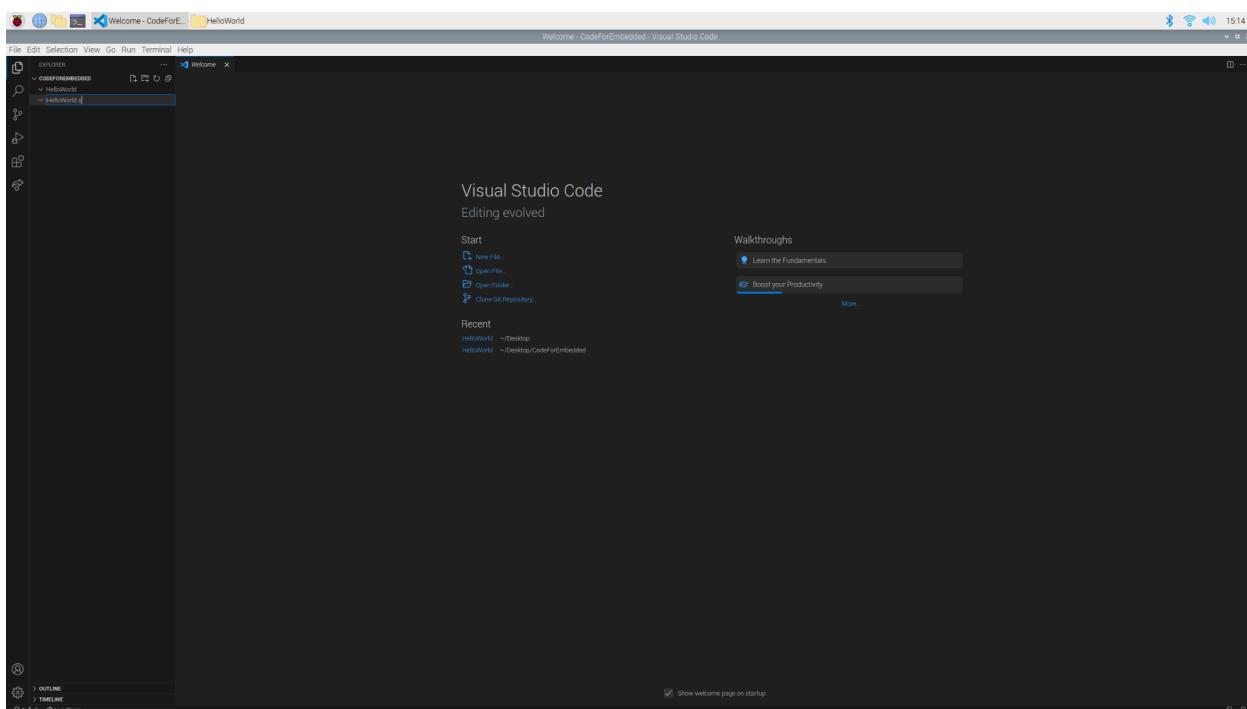


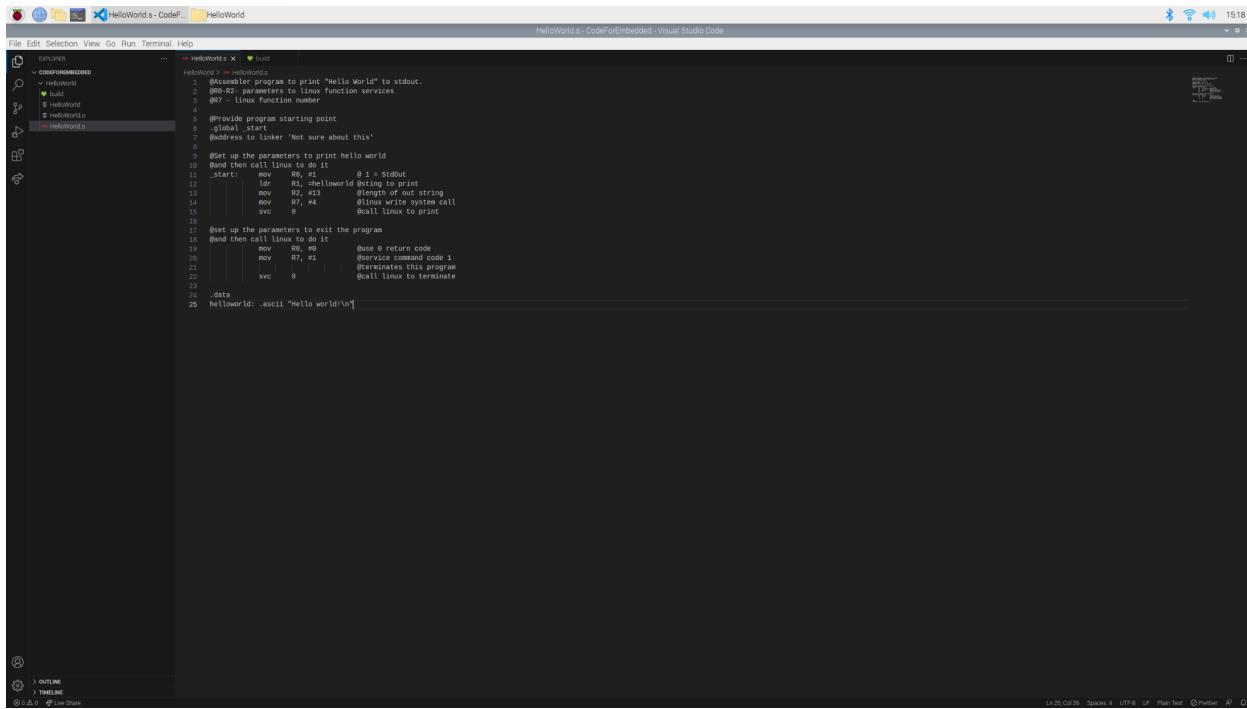
This will begin with prompting you to set it up and customize it, none of this is necessary and you can skip it. When you get past the initial setup you are brought to a screen where you can create a file or open a folder. We are going to open a folder. I suggest creating a general folder at first and name it 4230 Embedded or something similar. You can do this by clicking open folder -> desktop -> folder plus icon -> enter name. Inside this folder is where I would keep other folders for each lab and project individually. So repeat the process to create the HelloWorld folder and select open.





Now we can create the assembly file for Hello world. Hover over the folder name and select the page with a plus over it or click file and new file. Name the file HelloWorld.s this signifies to VS code that we are coding in assembly. Once this file is open enter the code included into the file and save it.

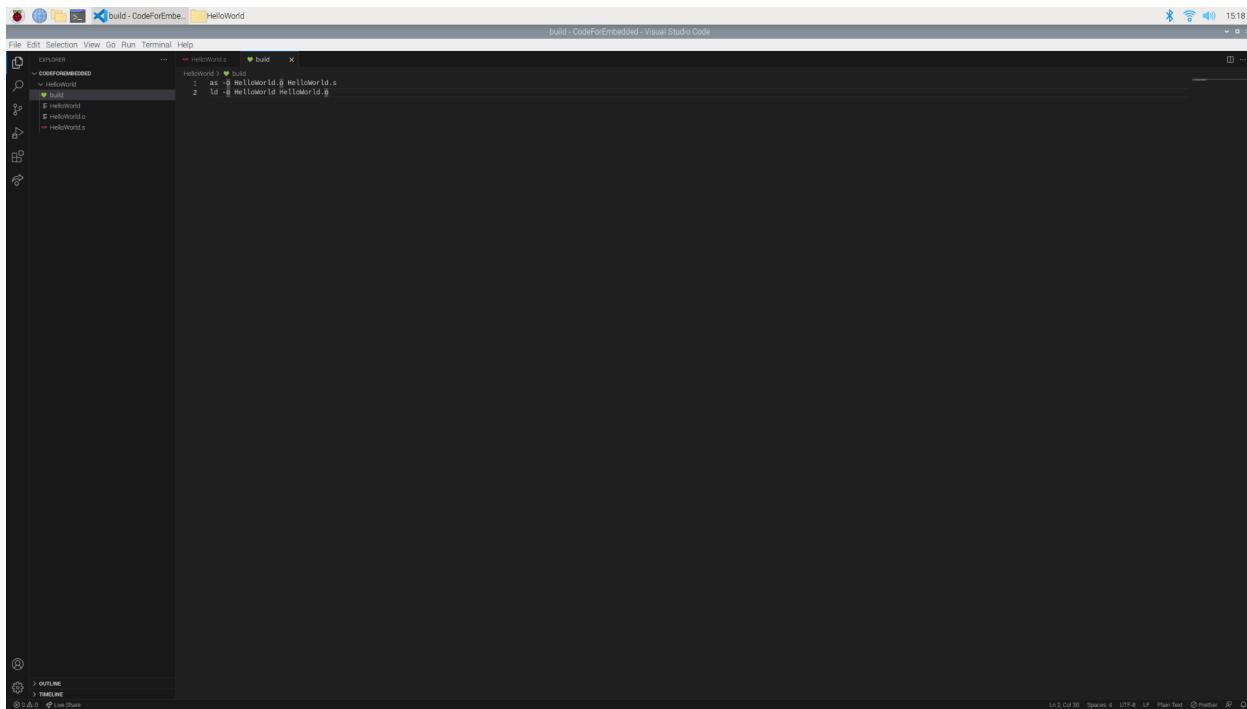




The screenshot shows the Visual Studio Code interface with the title "HelloWorld.s - CodeForEmbedded - Visual Studio Code". The main editor area displays the assembly code for the "HelloWorld.s" file. The code is as follows:

```
1  @Assembler program to print "Hello World" to stdout.
2  #define parameters to Linux function services
3  #define _NR - linux Function number
4
5  /*Provide program starting point
6  *global_start
7  *Address to linker "Not sure about this"
8
9  *Set up the parameters to print hello world
10 /*and then call Linux to do it
11 _start:    ldr    R1, =helloWorld @String to print
12         mov    R2, #13 @Length of string
13         mov    R3, #4 @String write system call
14         svc    0 @Call Linux to print
15
16 /*Set up the parameters to exit the program
17 /*and then call Linux to do it
18         mov    R0, #0 @Return code
19         mov    R7, #1 @Service command code 1
20         svc    0 @Terminates this program
21
22         svc    0 @Call Linux to terminate
23
24 .data
25 helloWorld: .ascii "Hello world!\n"
```

Then create another file called build using the code included

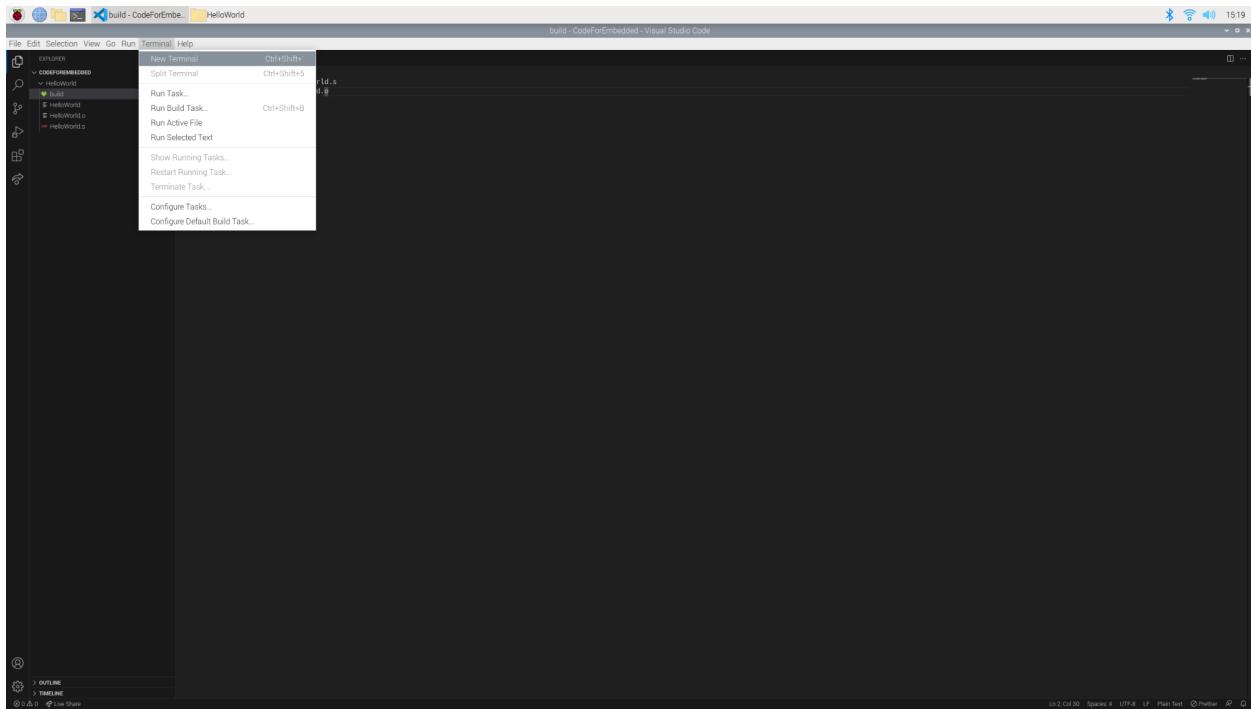


The screenshot shows the Visual Studio Code interface with the title "build - CodeForEmbedded - Visual Studio Code". The main editor area displays the build file, which contains the following assembly commands:

```
1  as -O HelloWorld.s HelloWorld.o
2  ld -O HelloWorld.o -o HelloWorld
```

The next step is to open the terminal, you can either open the linux terminal through raspberry pi and cd to the folder where you created the files or use the built in terminal with VS code which

is what I suggest. You can do this by clicking the terminal option in the top of the screen and clicking 'new terminal'.



Once the terminal is open it is time to compile and run hello world. To do this type in the terminal 'bash -x build'. This will compile the program you wrote and to run it you will type ./HelloWorld. This should print Hello World to the terminal screen.

The screenshot shows the Visual Studio Code interface with the following details:

- Explorer View:** Shows the project structure under "HelloWorld". It includes a "build" folder containing "HelloWorld.s", "HelloWorld.o", and "HelloWorld.s".
- Terminal View:** Displays the command-line session:

```
cory@raspberrypi:~/Desktop/CodeForEmbedded$ bash -x build
bash: build: No such file or directory
cory@raspberrypi:~/Desktop/CodeForEmbedded$ cd HelloWorld
bash: cd: HelloWorld: No such file or directory
cory@raspberrypi:~/Desktop/CodeForEmbedded$ cd HelloWorld
bash: cd: HelloWorld: No such file or directory
cory@raspberrypi:~/Desktop/CodeForEmbedded$ bash -x build
+ cd HelloWorld
HelloWorld:1: Assembler messages:
HelloWorld:1:1: warning: file not at end of a line; newline inserted
+ ld -o HelloWorld HelloWorld.o
cory@raspberrypi:~/Desktop/CodeForEmbedded$ ./HelloWorld
HelloWorld
cory@raspberrypi:~/Desktop/CodeForEmbedded$
```
- Status Bar:** Shows the date and time as "15:24".
- Bottom Status Bar:** Shows file counts and sizes: "1+2, 0+0", "Opened: 8", "177.0", "LF", "Plain Text", and "Editor".

Now that you have used the standard build, retry this with a make file. You can do this by deleting the .o file and the other HelloWorld file *do not delete HelloWorld.s. Using the make file included create a file called makefile. Once the file is created save it and use the terminal type make. This should create the HelloWorld.o file and allow you to run it by typing ./HelloWorld

References:

Raspberry pi assembly language programming by stephen smith

<https://code.visualstudio.com/docs/setup/raspberry-pi>