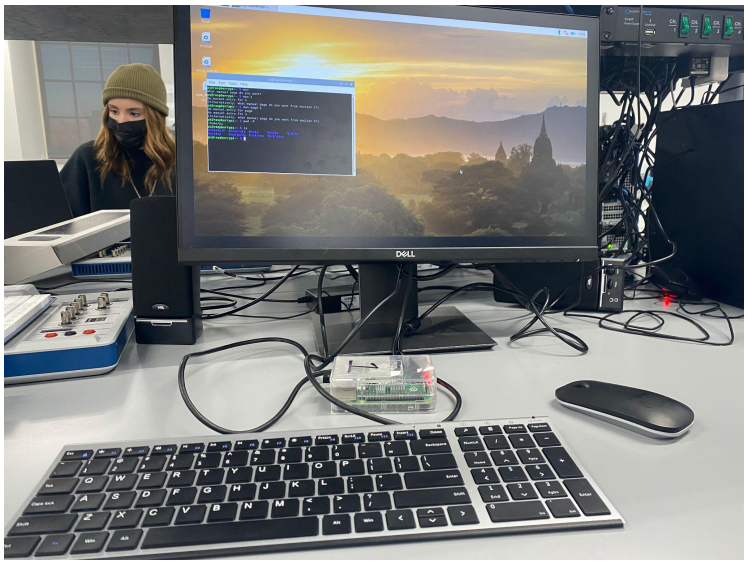
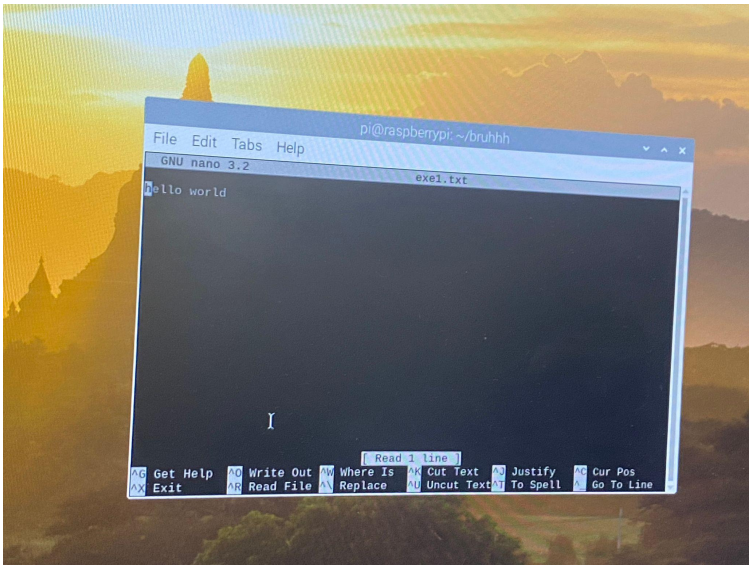
Lab #1 - Introduction to Raspberry Pi, Assembly Language, and Basic I/O

Roger Bennett & Matt Law

Exercise 1-



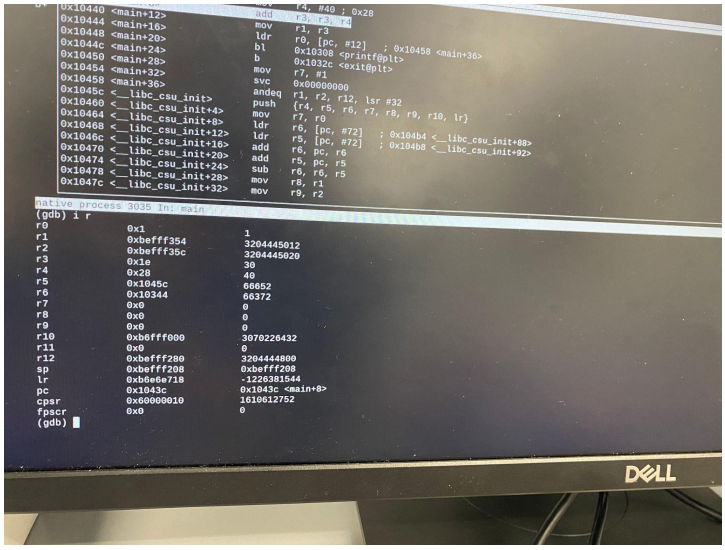


Using basic Linux commands, we created and edited a file, created a directory, and copied files.

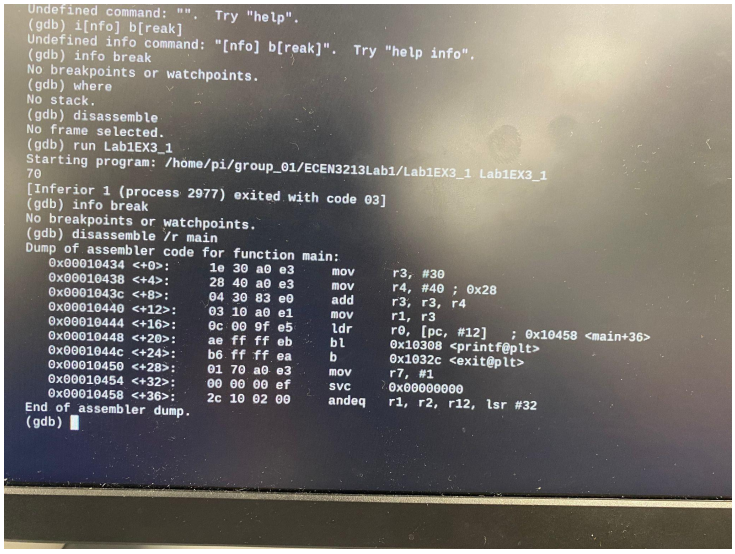
Exercise 2

We easily set up the Wi-Fi for the RPi through the steps listed in the documents, and set up the remote connection.

Exercise 3.1




```
Dump of assembler code for function main:
   0x00010610 <+0>:     00 48 2d e9     push    {r11, lr}
   0x00010614 <+4>:     04 b0 8d e2     add     r11, sp, #4
   0x00010618 <+8>:     08 d0 4d e2     sub     sp, sp, #8
   0x0001061c <+12>:    1e 30 a0 e3     mov     r3, #30
   0x00010620 <+16>:    08 30 0b e5     str     r3, [r11, #-8]
   0x00010624 <+20>:    28 30 a0 e3     mov     r3, #40 ; 0x28
   0x00010628 <+24>:    0c 30 0b e5     str     r3, [r11, #-12]
   0x0001062c <+28>:    0c 20 1b e5     ldr     r2, [r11, #-12]
   0x00010630 <+32>:    08 30 1b e5     ldr     r3, [r11, #-8]
   0x00010634 <+36>:    03 30 82 e0     add     r3, r2, r3
   0x00010638 <+40>:    0c 30 0b e5     str     r3, [r11, #-12]
   0x0001063c <+44>:    0c 10 1b e5     ldr     r1, [r11, #-12]
   0x00010640 <+48>:    10 00 9f e5     ldr     r0, [pc, #16]   ; 0x10658 <main+72>
--Type <RET> for more, q to quit, c to continue without paging--layout split
```



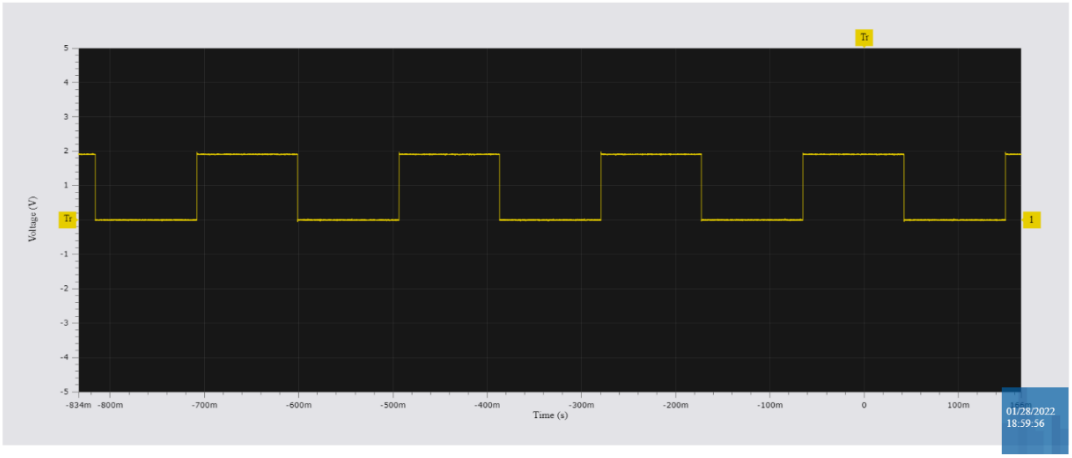
In this exercise we were introduced to assembly language and how to debug the code. We were given a code file and we were able to make some modifications so we could get the desired output. We then learned how to use gdb to debug the code and find out what the address and machine code for each line was. We then set breakpoints at each line.

Exercise 3.2

We added an addition operation to generate C++ from machine code.

Exercise 4.1

The goal for this exercise was to get an LED to blink from connecting it to the RPi and running a program. After we constructed the circuit. We fixed up the piece of code and watched as the light started blinking. We then connected it to an Oscilloscope to measure the voltage.



General Settings	
Trigger Type	Analog edge
Trigger Source	Channel 1
Trigger Slope	Rising
Trigger Level	0 V
Time Per Division	100 ms
Sampling Mode	Decimate
Repetitive Sampling Mode	Off

Advanced Trigger Settings	
Acquisition Delay	Disabled
Trigger Position	-333.7 ms

Channels						
Name	State	Coupling	Probe Attenuation	Vertical Offset	Vertical Position	Volt Per Division
Channel 1	Enabled	DC	1x	0 V	0 V	1 V
Channel 2	Disabled					
Channel 3	Disabled					
Channel 4	Disabled					

Reference Channels						
Name	State	Reference Mode	Source File Name	Source File Channel	Vertical Position	Volt Per Division
Reference 1	Disabled					
Reference 2	Disabled					
Reference 3	Disabled					
Reference 4	Disabled					

Reference Channels							
Name	State	Reference Mode	Source Circuit	Source Probe	Trigger Source Checked	Vertical Position	Volt Per Division

Additional Channels - FFT	
Name	FFT
State	Disabled

Additional Channels - Math	
Name	Math
State	Disabled

Channel Measurements Data				
Channel Name	VPP	RMS	Frequency	Period
Ch 1	2.037 V	1.298 V	4.66 Hz	214.6 ms

Exercise 4.2

In this exercise, we constructed a more complicated circuit where it included a button switch that, when pressed, would turn on the LED. We did this by completing the code we were given and executing the code.

Supplemental Questions

To summarize the goal of this overall lab, we were taught the basics of Raspberry Pi setup, basic functions and commands of Linux, code implementation and debugging, and some basic circuit setups with wires and resistors.

Machine Language is used by computers to execute a program, unfortunately it is very difficult for humans to understand and program in. Assembly language is a step above Machine Language. It is a basic language that is easy to program in. The problem with Assembly language is that it is a very low level language and it has its limitations especially with organization. C/C++ is a very high level programming language that is very easy to learn, understand, and program in. There are more steps that need to be taken in order to take the code that is written in C/C++ to a language that the computer can understand such as Machine Language.

After creating a C/C++ source program it will go to the next step of Modified Source Program, from there it will be translated into an Assembly Program and then Relocatable object Program, until finally becoming an Executable object Program. When writing a C/C++ Program you will need to compile the code using a gcc or g++ command line and then finally run the executable program.