Objective:
Understanding the basics of parallel input and output using the programming language Assembly.

Summary:
In part one, we programmed a one second delay subroutine by using nested loops. The inner loop held a variable called Counter L that would start at 0xF5 and decrement by one every iteration until it reached 0x00. Once Counter L reached 0x00, the outer loop would decrement its own variable called Counter H starting at 0xFF. After setting PORTB to be a digital output, we used a loop that would alternate the value of PORTB between 0x05 and 0x0A with a one second delay. This would alternate the on and off states of the LEDs connected to PORTB. We then connected the logic analyzer to show the waveform of one of the LEDs to ensure it had a half period of one second.

In part two, we programmed PORTA to be a digital input and PORTB to be a digital output. We connected input switches to PORTA and used an infinite while loop to mask the least significant three bits before outputting the value through PORTB. The output can be seen visually by three LEDs connected to PORTB which would mirror the input switches.

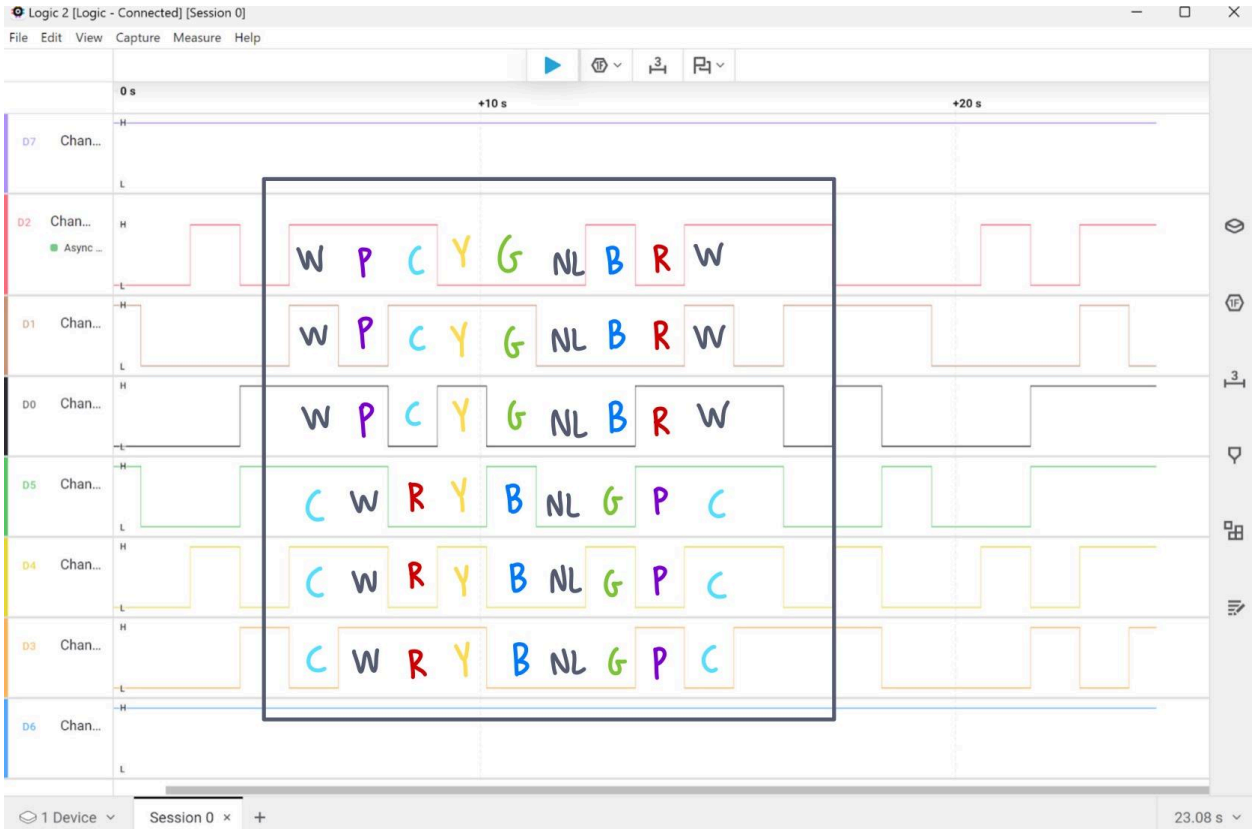
In part three, we used the code from part two and switched the output to PORTC and connected an RGB LED to PORTC. This would allow us to control the color of the RGB LED using the input switches to cycle through eight different colors.

In part four, we made a for loop that will cycle through values from 0x00 to 0x07. This was accomplished by setting a location in storage that would hold the color value and another space to hold the loop count. The color value will be sent to the RGB LED, wait for one second, then increment the color value and decrement the loop count. When the loop count reaches 0x00, a while loop will reset the values of the color value to 0x00 and the loop count to 0x08 and restart the for loop.

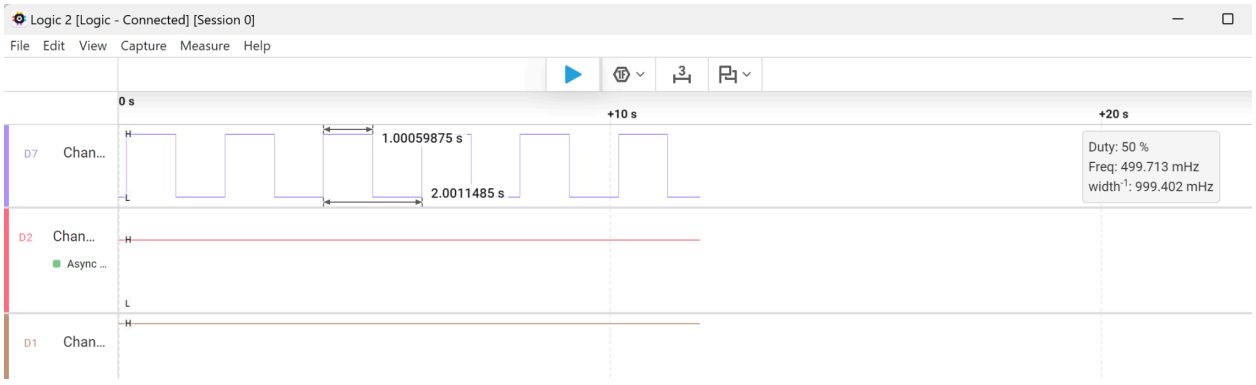
In part five, we loaded the color values in a certain order into consecutive storage locations, then set the value of FSR0 to the address of the first location. We repeated this with a second order of the same color values and used FSR1. We used a loop that would send the color value in the address being pointed to by FSR0 and send it to the first RGB LED and send the color value pointed to by FSR1 to the second RGB LED. The loop would then increment FSR0 and FSR1 to point to the next address and the loop would repeat.

Data Collected:

Annotated Waveform of RGB LED 1 and RGB LED 2



Waveform of PORTB Bit 0



Conclusion:

Using assembly to program a PIC18F4620 for digital input and output was effective. While preparing for this lab we learned about indirect addressing mode and how to utilize it to create an array with a variable index. We also learned about subroutines when creating a one second long delay. Pin connections were very important in this lab because we needed to use specific pins to create the correct waveforms. We also needed to shift values in one part because we used the most significant bits as opposed to the least significant ones that we used in the rest of the lab. A combination of loops and subroutines were a good way to control digital input and output using assembly.