Objective:
Understanding the basics of arithmetic, logical and branching instructions using assembly programming language.

Summary:
In part one, we made a program to take a four bit input from a DIP switch bank and had each switch inverted and then displayed with LEDs as an output. We used two declared variables to hold the value from the input switches and the result which would be displayed in the output LEDs. We used the logical and operation to mask the upper four bits and we used the complement operation to invert the input values before masking a second time and displaying the results on the LEDs

In part two, we added an LED to display if the result had zero as a value. If the result was zero, the new LED would turn on and if the result was any other value, the LED would turn off. To accomplish this, we added a branching instruction to turn on the LED if the Z flag was also set. If the Z flag was not set, the branching instruction would be skipped and a second branching instruction would turn off the LED before restarting the loop.

In part three, we declared a new variable that would take input from a second DIP switch bank. This time the program would add the values from the two switches before displaying the value in binary using LEDs. We used the same process for masking the values as in part one and the same zero value check as in part two.

In part four, we used the same code as part three and replaced the logical and operation with the logical inclusive or operation. This program would take the input of one switch and logically or each bit with the bits of the second input bit. The result would be displayed by the output LEDs along with the zero value check LED.

In parts five and six, we repeated the process in part four but the logical and was used in part five and the logical exclusive or was used in part six. Both of these parts took the input from the two switch banks and displayed their respective outputs on the LEDs.

In part seven, we took the input from only one switch bank and converted it to a BCD number. To accomplish this we would compare the value of the input with a value of nine and add six if the value exceeded nine but kept it the same otherwise. We added six to adjust the hexadecimal value to show a decimal value in the output.

In part eight, we made a combination of all previous parts and controlled the operation performed with a third input switch. We utilized calling subroutines for each operation. We first had to name and define all the subroutines before calling them in the main loop. We defined the subroutines after the main loop and altered some of the names to account for reused code like the z flag check.

Data Collected:
No Data was collected.

Conclusion:
Assembly is an effective language to control arithmetic and logical operations as well as branching instructions. Conditional branching instructions were helpful in implementing the z flag check using only a few lines to properly check and define a subsequent action. The arithmetic and logical operations were a longer process in assembly compared to a language like C.