# Lab 4

Madiba Hudson-Quansah

**Question 1**

What is the main difference between using unconditional jumps (j) and loop structures (while, do-while)?

***Solution:*** Unconditional jumps only jump to a specific location in the code for the number of times specified, but loops will continue to execute until a certain condition is met

**Question 2**

When should you use j instead of a loop? Provide a real-world example where unconditional jumps are useful.

***Solution:*** A jump should be used instead of a loop when you want to jump to a specific location to continue an alternate path of execution. An example is when you want to prematurely exit a while loop while the loop condition is still met.

**Question 3**

How does jal (jump and link) differ from j (jump)? Why is jal useful for calling functions?

***Solution:*** jal saves the address of the next instruction in the link register which allows the program to return to the function's caller after the function has been executed.

**Question 4**

Why do we use jr $ra at the end of a function? What would happen if we didn't include it?

***Solution:*** jr $ra is used to return to the function's caller where $ra is the link register. It is used as a way to return to the caller of a jal instruction and if it was not included the program would not continue its designated path of execution.

**Question 5**

In high-level languages like C or Python, we don't explicitly write j instructions. What happens at the assembly level when a function is called?

***Solution:*** When a function is called the program jumps to the address of the function saving the address of the next instruction in the link register. The function then executes and when it is done it jumps back to the address saved in the link register.