Course:	COMP 2131	**Programming Assignment-3**
Weight:	**100 marks = 10% of final grade**	**Working with Assembly loops**

Step 1. Write a program in GNU assembly language that uses macros to print the following messages on the screen [20 marks]:

Hello, programmers!
Welcome to the world of,
Linux assembly programming!

Step 2. Using the C Programming language, write four version of a function that contain a loop. Each function should accept two numbers and calculate the sum of all numbers between the first number and last number (inclusive of the first and last number). Tip: Try to use the same number of variables and almost the same logic when writing the C code for all four functions**.** Once each function is written and tested for correct output, generate an assembly language version of the function using the command: *$gcc -O1 -S filename.c.* (where *filename.c* is the C program containing the function). Next, analyze and compare the assembly language version of each function. [50 marks]

a. Write a version of the function using a *for* loop
b. Write a version of the function using a *while* loop
c. Write a version of the function using a *do..while* loop
d. Write a version of the function using a *goto* loop
e. Is the assembly language version of each loop function the same or different? If different, identify the differences. Your comparison should be based on:
    - Number of registers used
    - Number of jumps (iterations)
    - Total number of operations

Step 3. Using the C Programming language, write a program that sums an array of 50 elements. Next, optimize the code using loop unrolling. Loop unrolling is a program transformation that reduces the number of iterations for a loop by increasing the number of elements computed on each iteration. Generate a graph of performance improvement. Tip: Figure 5.17 in the textbook provides an example of a graph depicting performance improvements associated with loop unrolling. [30 marks]

Submission:

Make a Word document that contains all the programs that you wrote for Step 1, Step 2 and Step 3.

- For Step 1, include only the assembly code.
- For Step 2, include the C code for the 4 functions and the assembly code that you generated for each function. Be sure to include your analysis of the assembly code generated from each function.
- For Step 3, include the C code and the assembly code that you generated for each step of the loop unrolling process.

Submit the word document to the Open Learning Faculty Member for grading.

Step	Jobs Done	Marks
I	Write a program in GNU assembly language using macros.	20
II(a)	Write a function in C containing a *for* loop	5
II(b)	Write a function in C containing a *while* loop	5
II(c)	Write a function in C containing a *do ..while* loop	5
II(d)	Write a function in C containing a *goto* loop	5
II(e)	Generate the assembly language version for each of the four functions; analyze and compare the assembly language code for the four functions.	30
III	Optimize the code for an array of 50 elements using loop unrolling. Prepare a graph that show the performance improvements	30
Total		100