Sanjana Madhu
10/9/2020
I pledge my honor that I have abided by the Stevens Honor System

In the main function, the main purpose is to initialize the values and register I'm working with. By the end of the main, x1 holds the value of n, d2 holds the value of x, and d0 is initialized to contain the final answer of the approximation.

The loop function checks if we are done calculating the approximation. It checks if the value of n is 0 and if it is, it branches to the end label where I print out the string. If not, it stores the link register and calls the function calc.

In calc, I initialize x19 and d20 so they can hold the values of n! and x^n after they have been computed. Finally in calc, in order to prevent changing the true value of n, I move the value of n in x1 to x2, and x2 is manipulated further in the next label.

In fact, I recursively calculate the value of n! and x^n.

In calc_term, I calculate the nth term of the approximation. I convert the value of n! into a double. Then I divide n! by 1 and multiply it by x^n; this is stored in d9. Then I add this term to d0. Then I subtract the true value of n and call br x30, which will then go to line 31. In loop_call, I load the value of x30 onto the stack and then call loop.

In end, I simply store my values on the stack, call printf to print out the string, and then load values from stack and call br x30.

At the end of the program, the link register will have the same value as it did at the beginning of the program!