gccTitle : Project 3
Point Value : 100
Deadline : 4/24 at 11.59pm

Topics Covered:
Introduction to NASM, x86-64 instructions and architecture, Loops, System calls, ASCII

Setting Things Up:
For this project you will be using the NASM assembler to write x86-64 assembly. NASM is accessible on linux.gl.umbc.edu and you need to make sure your program compiles on it before submission.

You can reach it from anywhere with an internet connection

- You need to use your UMBC account
- ssh your-user-id@linux.gl.umbc.edu
- Enter your password (will not show you typing)
- The first time you are in create a CMSC313 folder
- mkdir cmsc313
- cd cmsc313

You can then use your favorite editor to write the program.

To assemble your program:

- Assemble it using: nasm –felf64 *filename*.asm
- Link to create executable: gcc –m64 –o *filename filename*.o
- Execute program: ./*filename* or *filename*

To transfer a file from your local computer to linux.gl.umbc.edu you can use the command below:
scp myFlie your-user-id@linux.gl.umbc.edu:cmsc313
This will put myFile in the cmsc313 directory on linux.gl.umbc.edu

To transfer from linux.gl to your computer:
scp your-user-id@linux.gl.umbc.edu:cmsc313/myFile /path/to/myFile

Project Description:

For this project we will be implementing a very basic encryption scheme known as a caesar cipher. The caesar cipher involves shifting every letter in the given string by a given number in the alphabet.

Write and submit a NASM assembly language program “caesar.asm” that:

- Asks the user for a location between 0-25.
    - You have to do error correction

- Asks the user for a string and displays the unedited string that the user entered.
    - The string from the user will be more than 30 characters long (spaces included)
    - You have to do error checking that the string is longer than 30 characters

- And finally displays an edited string to the user edited in the following way:
    - Shift every letter in the message by the indicated number of positions in the alphabet. When a letter gets shifted past the beginning or end of the alphabet, wrap around to the other end to continue. For example, a 6 would shift any 'e' characters to 'k'. The shift must be applied to all letters in the message. Do not change any non-alphabet characters in the message, and preserve all capitalization.

Below are 2 sample runs of the program:

Sample run 1:

Enter shift value: ***6***
Enter original message: ***This is the original message.***
Current message: This is the original message.
Encryption: Znoy oy znk uxomotgr skyygmk.

Sample run 2:

Enter shift value: 10
Enter original message: ***This is the original message.***
Current message: This is the original message.
Encryption: Drsc sc dro ybsqsxkv wocckqo.

Notes:

- Remember characters/numbers are read in ASCII.
    - You have to tackle double digit numbers

General Project (Hard) Requirements:

- **Code that does not assemble will receive 50% off**
- **We are programming for 64 bit Intel Architecture**
    - **Code written for 32 bit Intel will receive 20 points off**
- **You can work individually or with another person for this project. No more than 2 people can work together.**
- You must have a comment at the top of the code detailing what the code does.
- This comment should also include your full name and user ID.
    - If working with 2 people both names and user IDs should be included.
- You must use good coding style with respect to variable names, spacing, labels and comments.
- Submit **only the .asm file named firstname_lastname_casear.asm** to blackboard.
    - Late submissions will accrue a deduction of 10 points for every 6 hours it is late.
    - **If working with 2 people then the file should be named: lastname_lastname_caeaser.asm**
- You cannot use C/C++ function calls. **You have to use system calls**
- Any late submissions will incur a penalty of 10 points per every 6 hours they are late.

Grading Breakdown: (remember that code that doesn't compile will not get full credit)

- [75] Functionality
    - [20]Display prompts to user and read in values
    - [10]Display unedited string
    - [10]Display edited string
    - [20]Text edit functionality
    - [5]Use of system calls
    - [5]Program exits correctly (No Seg Faults)
    - [5]No extra new lines or spaces printed
- [25] Style (only if it compiles)
    - [5]Comments where necessary
    - [5]Code written for 64 bit Intel Assembly
    - [5]Comment at top of program with name, user id and explanation of code
    - [5]Code easy to read
        - Good variable names
        - Indentation

- [5]Submitted correctly