CMSC 313 — Spring 2017
Project 1 — ROT13 Encryption

Assigned	Tuesday, February 21st
Program Due	Tuesday, February 28th by 11:59pm
Updates:	None yet.

Objectives

Through this project, you will practice writing loops, indexing through strings, and using conditional jumps in assembly language.

Background

ROT13 is a substitution cipher where every character in the text is replaced with the letter 13 positions later in the alphabet, with wraparound. The following description is taken from the Wikipedia page:

Applying ROT13 to a piece of text merely requires examining its alphabetic characters and replacing each one by the letter 13 places further along in the alphabet, wrapping back to the beginning if necessary. A becomes N, B becomes O, and so on up to M, which becomes Z, then the sequence continues at the beginning of the alphabet: N becomes A, O becomes B, and so on to Z, which becomes M. Only those letters which occur in the English alphabet are affected; numbers, symbols, whitespace, and all other characters are left unchanged. Because there are 26 letters in the English alphabet and 26 = 2 × 13, the ROT13 function is its own inverse.
[...]
The transformation can be done using a lookup table, such as the following:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
NOPQRSTUVWXYZABCDEFGHIJKLMnopqrstuvwxyzabcdefghijklm
```

For example, in the following joke, the punchline has been obscured by ROT13:

```
Why did the chicken cross the road?
Gb trg gb gur bgure fvqr!
```

Transforming the entire text via ROT13 form, the answer to the joke is revealed:

```
Jul qvq gur puvpxra pebff gur ebnq?
To get to the other side!
```

A second application of ROT13 would restore the original.
[END OF WIKIPEDIA SECTION]

ROT13 is not a very good cipher--it is defeated very easily; however, encryption was never the real intent. ROT13 was/is used to obfuscate text so that the original text is temporarily hidden from the reader. It was used for purposes like hiding

punchlines to jokes, or spoilers in movie reviews, so that the reader would not accidentally see the real message.

Assignment

Your assignment for this project is to write an assembly language program that computes the ROT13 of an input.

Your program will print out a prompt, then read an input string from the user. Note that the "read" system call does not null-terminate the input string--you must use the returned byte count.

The main body of your program will then loop over the characters in the input string, converting the alphabetic characters to their ROT13 equivalents. All non-alphabetic characters (digits, punctuation symbols, etc.) must be left as-is.

The program must then output the translated string. The output must be EXACTLY of the form:

```
Enter string: Why DID the chicken?
Original: Why DID the chicken?
Convert:  Jul QVQ gur puvpxra?
```

Your program must then exit cleanly using the exit system call.

Sample Run

Your program input/output should look exactly like the following (user input is underlined):

```
acw3f@ubuntu:~$ ./rot13
Enter string: abcdef
Original: abcdef
Convert:  nopqrs
acw3f@ubuntu:~$ ./rot13
Enter string: AbCdEfGhI
Original: AbCdEfGhI
Convert:  NoPqRsTuV
acw3f@ubuntu:~$ ./rot13
Enter string: NoPqRsTuV
Original: NoPqRsTuV
Convert:  AbCdEfGhI
acw3f@ubuntu:~$ ./rot13
Enter string: 1 + 1 = Two
Original: 1 + 1 = Two
Convert:  1 + 1 = Gjb
acw3f@ubuntu:~$ ./rot13
Enter string: 1 + 1 = Gjb


Original: 1 + 1 = Gjb
Convert:  1 + 1 = Two
```

Implementation Notes

It would probably be very helpful to start from the source for toupper.asm, as that already has code to do simple input and output system calls, as well as templates for defining .data and .text sections. However, you will have to restructure the main loop substantially to do the ROT13 character conversion.

You can download the source for toupper with the following command on GL:

```
cp -r /afs/umbc.edu/users/p/a/park/pub/cmsc313/fall16/code/toupper .
```

Note the '.' at the end of the line--it is important. Also note that the "toupper" directory contains both the source for toupper.asm as well as a Makefile, both of which you should adapt to develop your project. Make sure you remember to rename the program "rot13.asm".

What to Submit

Use the UNIX `submit` command on the GL system to turn in your project. You should submit a single assembly language program file, named `rot13.asm`. The UNIX command to do this should look something like:

```
submit cs313_park proj1 rot13.asm
```

We will be testing it on the GL machines--make sure your program assembles and loads without ***any*** errors or warnings!