What is the secret message?

COMP 273 Assignment 2 - Fall 2021

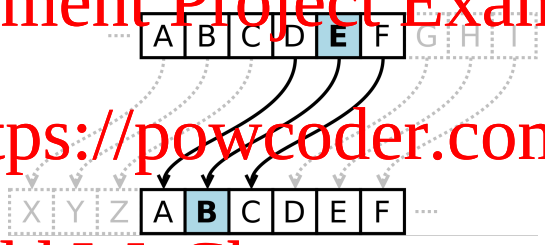
QEB QEOBB IXTP LC OLYLQFZP
CFOPQ IXT: X OLYLQ JXV KLQ FKGROB X ERJXK YBFKD LO, QEOLRDE FKXZQFLK, XIILT X ERJXK YBFKD QL ZLJB QL EXOJ. PBZLKA IXT: X OLYLQ JRPQ LYBV QEB LOABOP DFSBK FQ YV ERJXK YBFKDP BUZBMQ TEBOB PRZE LOABOP TLRIA ZLKCIFZQ TFQE QEB CFOPQ IXT. QEFOA IXT: X OLYLQ JRPQ MOLQBZQ FQP LTK BUFPQBKZB XP ILKD XP PRZE MOLQBZQFLK ALBP KLQ ZLKCIFZQ TFQE QEB CFOPQ LO PBZLKA IXT.
FPPXZ XPFJLS EXKAYLLH LC OLYLQFZP 56QE BAFQFLK, 2058 X.A.

Submission instructions

All work must be your own, and must be submitted by MyCourses. **Include your name and student number in a comment at the top of your source file**. Also use comments at the top of your code to provide any other information you would otherwise include in a README file. Submit only one file, `a2.asm`. Do not use a zip archive. **Check your submission** by downloading your submission from the server and checking that it was correctly submitted. You will not receive marks for work that is incorrectly submitted.

Overview

In this assignment, you will encrypt and decrypt messages where letters are *shifted* by a given number, which is called the *key*. For instance, a key of -3 change "D" to "A". At the end of the alphabet, we wrap back to the beginning, so a shift of -3 also changes "A" to "X".



We will assume that the text is in ASCII and we will work only with capital letters. The key is an integer between -10 and 10. Where the plain text contains spaces and punctuations, we will leave the spaces and punctuations unchanged, and skip to the next letter in the text. For instance "LET'S GO TO THE CAT MUSEUM!" with key "3" will becomes "OHW'V JR WR WKH FDW PXVHXP!"

Decryption is done by simply shifting by a negative amount of the key. For example, key=3 means shift the letter by -3 when decrypting a message.

Provided code and files

You are provided some code to help you get started, along with a small collection of test files of encrypted text (`messages0.txt` - `messages4.txt`). Once you've completed the objectives of the assignment, you should be able to discover the original message for these encrypted texts. There are no marks associated for discovering the original messages, but feel free to discuss your discovery with your classmates!

The provided code `a2.asm` implements a simple command menu to let you read a text file into a buffer, print the text in the buffer, encrypt the buffer, decrypt the buffer, write the buffer to a file, and quit. The code handles reading from and writing to files and a few other basic procedures to let you focus on the job of writing procedures for encrypting text, decrypting text, and guessing keys. Here follows an example session.

```
Commands (read, print, encrypt, decrypt, write, quit):r
Enter file name:plain0.txt
LET'S GO TO THE CAT MUSEUM!

Commands (read, print, encrypt, decrypt, write, quit):e
```

```
Enter key (upper case letters only):3
OHW'V JR WR WKH FDW PXVHXP!

Commands (read, print, encrypt, decrypt, write, quit):w
Enter file name:out.txt

Commands (read, print, encrypt, decrypt, write, quit):p
OHW'V JR WR WKH FDW PXVHXP!

Commands (read, print, encrypt, decrypt, write, quit):d
Enter key (upper case letters only):3
LET'S GO TO THE CAT MUSEUM!

Commands (read, print, encrypt, decrypt, write, guess, quit):q
```

Test your program

We provide a few files for you to test your program.

- `plain0.txt` contains a simple sentense. You can try to encode it and then decode it. You should get exactly the same sentense back
- `messages0.txt` - `messages4.txt` are messages encoded with key = -3.

Implementations

Note that MARS will open files by default in the directory in which it is started. You can place your files in that location for easy reading and writing, or specify the full path, or likewise specify the directory in which to start the MARS jar.

Extra MIPS instructions

Here are two extra MIPS instructions that will help you with this assignment:

```
lbu $DistReg offset($AddrReg)  # load a byte from offset($AddrReg) to $DistReg
sb  $DataReg offset($AddrReg)  # store a byte from $DataReg to offset($AddrReg)
```


The above two instructions are similar to `lw` and `sw`. Insteading of loading/storing a *word*, `lbu` and `sb` loads/stores a *byte* (to/from the least significant byte of the register).

EncryptMessage

Write the MIPS assembly code to encrypt the messages, and place your code right after the label `EncryptMessage`. You can assume that `$a0` contains the address to the *message* stored in the memory, `$s1` contains the number of characters in the messages, and `$a1` contains the *key*. You only need to encrypt the upper case letters. The maximum size of the messages is 10000 characters.

DecryptMessage

Write the MIPS assembly code to decrypt the messages, and place your code right after the label `DecryptMessage`. You can assume that `$a0` contains the address to the buffer, `$s1` contains the number of characters in the messages, and `$a1` contains the *key*. You only need to decrypt the upper case letters. The maximum size of the messages is 10000 characters.