Assembly Language Lab: Encrypting Text

CSC 242 Lab 3
Handed Out: see course calendar
Due Date: see course calendar

Background:
Cryptography, from the Greek word kryptos, meaning "hidden", deals with the science of hiding a message from eyes you do not want to understand the contents of the message. The desire to do this has existed ever since humankind was first able to write. There are many ways to keep something secret. One way is to physically hide the document. Another way is to encrypt the text you wish to remain secret. Some people use this to keep others from understanding the contents of the files in their computer. Encrypting a file requires an input message, called the "plain text," and an encryption algorithm. An encryption algorithm transforms "plain text" into "cipher text." Just like a door needs a key to lock and unlock it, an encryption algorithm often requires a key to encrypt and decrypt a message. Just like a homeowner can selectively give the key to the front door to only those he wants to allow unaccompanied access, the author of a message can selectively give the encryption key to only those he wants to be able to read the message. In order for someone to read the encrypted message, he has to decrypt the cipher text, which usually requires the key.

For example, suppose the plain text message is HELLO WORLD. An encryption algorithm consisting of nothing more than replacing each letter with the next letter in the alphabet would produce the cipher text IFMMP XPSME. If someone saw IFMMP XPSME, he would have no idea what it meant (unless, of course, he could figure it out, or had the key to decrypt it.) The key to encrypt in this case is "pick the next letter in the alphabet," and the decryption key is "pick the previous letter in the alphabet." The decryption algorithm simply replaces each letter with the one before it, and presto: the plain text HELLO WORLD is produced.

Instructions:
Implement, in LC-3 assembly language, an encryption/decryption program that meets the following requirements:

Input
Your program should prompt the user for three separate inputs from the keyboard, as follows:

1. The prompt: IF YOU WANT TO ENCRYPT, TYPE E; IF YOU WANT TO DECRYPT, TYPE D:

> · The user will type E or D. A real world program should also detect any other character typed and respond with THAT IS AN ILLEGAL CHARACTER. PLEASE TRY AGAIN. We will assume in this assignment that the user can type an E or D correctly.

· Your program will accept that character, store it in x3100, and use it, as we shall see momentarily.

2. The prompt: ENTER THE ENCRYPTION KEY (A SINGLE DIGIT FROM 1 TO 9):

· The user will type a single digit, from 1 to 9. Again, we will assume the user is not an Aggie, and can successfully hit digit keys on the keyboard.

· Your program will accept this digit, store it in x3101, and use it to encrypt or decrypt the message.

3. The prompt: INPUT A MESSAGE OF NO MORE THAN 20 CHARACTERS. WHEN DONE, PRESS <ENTER>

· The user will input a character string from the keyboard, terminating the message with the <ENTER> key.

· Your program will store the message, starting in location x3102. Since the message is restricted to #20 characters, you must reserve locations x3102 to x3115 to store the message. (Note that #20 = x14.)

· One constraint: Messages must be less than or equal to #20 characters. (Recall: # means the number is decimal.)

Algorithm

The encryption algorithm is as follows. Each ASCII code in the message will be transformed as follows:

1. the low order bit of the code will be toggled. That is, if it is a 1, it will be replaced by a 0; if it is a 0, it will be replaced by a 1.

2. The key will be added to the result of step 1 above.

· For example, if the input (plain text) is A and the encryption key is 6, the program should take the ASCII value of A, 01000001, toggle bit [0:0], producing 01000000 and then add the encryption key, 6. The final output character would be 01000110, which is the ASCII value F.

The decryption algorithm is the reverse. That is,

1. Subtract the encryption key from the ASCII code.

2. Toggle the low order bit of the result of step 1.

· For example, if the input (cipher text) is F, and the encryption key is 6, we first subtract the encryption key (i.e., we add -6) from the ASCII value of F, 01000110 + 11111010, yielding 01000000. We then toggle bit [0:0], producing 01000001, which is the ASCII value for A.

· The result of the encryption/decryption algorithm should be stored in locations x3116 to x3129.

Output

Your program should output the encrypted or decrypted message to the screen. Note that the encryption/decryption algorithm stored the message to be output starting in location x3116.

```
- The starting location is to be x3000
- The buffers have been expanded to 21 memory locations each so that
you may store <ENTER>
x3100 <- E/D
x3101 <- Encryption key
x3102 through x3116 <- input buffer x3117 through x312A <- output
buffer
- The <ENTER> key is mapped to the line feed character on the LC-2
(ASCII x0A)
- Acceptable inputs to be encrypted are any ASCII characters within the
range x20 to x5A
- You may assume the user will only type an upper-case E or D
- The prompts are
Prompt 1: (E)ncrypt/(D)ecrypt:
Prompt 2: Encryption Key:
Prompt 3: Input Message:
```

Hints and suggestions
To continually read from the keyboard without first printing a prompt on the screen, use TRAP x20. That is, for each key you wish to read, the LC-3 operating system must execute the TRAP x20 service routine. If you follow TRAP x20 with the instruction TRAP x21, the character the user types will be displayed on the screen.

Writing and testing your program
Your code should have a comment block at the beginning of the file containing your name, your email, student number, and a brief description of the program. Your description should serve as a general summary of your program's approach to the problem and will aid in grading. It is in your best interest to make all of your ideas clear through this summary and through commenting within your code.

IMPORTANT NOTE: The first line of your program must specify the memory address of where you want your program to be placed (using the .ORIG pseudo-op). We request you place your program at x3000.

When and how to submit: The programs will be submitted in class by turning a hard copy of the programs with your full name and date. The program will be evaluated during class. The instructor or TA will evaluate documentation and logic outside of class.

Grading: 50% for technical approach to be illustrated through the program description and comments in the code**,** 30% properly executing; 20% for formatting and programming style.