

CS 131 Assignment 8

Encryptor / Decryptor

Note: This is a group assignment with one to three students in a group. The group leader shall be responsible for uploading the deliverable items.

Objectives: To learn how to program in the LC-3 assembly language, including:

- assembly language instructions
- pseudo-ops
- TRAPS
- basic I/O

1. 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.

2. Assignment

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

2.1 Input

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

Prompt 1: **Enter E)ncrypt or D)ecrypt:**

- Echo and validate the user input. If neither 'E' or 'D' are entered, redisplay the prompt.

Prompt 2: **Enter Encryption Key (1-9):**

- Echo and validate the user input. If the entered key is not between 1 and 9 inclusive, redisplay the prompt.

Prompt 3: **Enter Message (<20 char, press <ENTER> when done):**

- The user will input a character string from the keyboard, terminating the message with the <ENTER> key. The entered message should be echoed back to the display.
- Your program will store the message in memory.
- Two input constraints:
 - Entered messages must be less than or equal to 20 characters. No need to range check for this exercise, but make sure you do not enter more than 20 characters.
 - Acceptable inputs to be encrypted are any ASCII character within the range of x20 – x5A (no lower case characters). No ASCII range check need be performed in the code, but make sure you do not enter characters that are outside the above range.

2.2 Algorithm

Encryption: The encryption algorithm transforms each ASCII code in the message as follows:

1. Toggle* the low order bit (bit 0, Least Significant Bit / LSB) of the ASCII code.
2. Add the Encryption Key to the result in step 1.

* Toggle means if the bit is a 1, replace the bit with a 0; if the bit is a 0, replace the bit with a 1.

For example, if the input (plain text) is 'A' (ASCII code 0x41) and the Encryption Key is 6, the program first toggles bit 0, producing 0x40. The program then adds the Encryption Key (6) to 0x40, yielding 0x46 – which is the ASCII code for 'F'.

Decryption: The decryption algorithm is the reverse of the encryption algorithm:

1. Toggle the low order bit (least significant bit / LSB) of the ASCII code.
2. Subtract the Encryption Key from the result in step 1.

For example, if the input (cipher text) is 'F' (0x46) and the Encryption Key is 6, the program first subtracts the Encryption Key (6), producing 0x40. The program then toggles bit 0, producing 0x41 – which is the ASCII code for 'A'.

Hint #1: Remember to convert the ASCII Encryption Key to an integer before performing arithmetic with the key.

Hint #2: Use code from the previous assignment to help you with the inputs and outputs on this assignment.

Hint #3: Implement the encryption and decryption portion first and then worry about the command and key validation.

Hint #4: For reasons that will become clear to you later, **do not use R7** to store anything for this assignment.

2.3 Output

Your program should **echo all user inputs** and output the encrypted or decrypted message to the screen using the following format:

- Encrypted Output: **Encrypted Message: {the encrypted message}**
- Decrypted Output: **Decrypted Message: {the decrypted message}**

2.4 Format. Your program must be a single text file of LC-3 assembly code (i.e., a .asm file).

- Include a header block at the top of your source code with all of the pertinent information (see Assignment 5, Source Code Format for further details).
- **Add comments to “blocks” of your code (you do not need to comment every line).**

2.5 Program Testbench

A testbench is used to verify the correctness or soundness of a product. For a program testbench, you create a list of inputs and expected outcomes. For this program, the testbench is (**user inputs are in blue bold**):

#	Inputs			Expected Outcome	Comments
1	E	3	HANCOCK	LCREQEM	Successfully encrypted
2	D	3	LCREQEM	HANCOCK	Successfully decrypted
3	Z			Reprompt Encrypt/Decrypt	'Z' is an invalid command
4	E	0		Reprompt Key	'0' is an invalid key
5		6	CS 131 ROCKS	K[*9;9*\WKS[Successfully encrypted
6	D	7	K[*9;9*\WKS[BR#020#UNBJR	Mismatch due to different keys
7	D	6	K[*9;9*\WKS[CS 131 ROCKS	Successfully decrypted

See the Example Testbench Output at the end of this assignment for clarification. As part of a complete testbench, many of the edge / corner / boundary cases should be exercised. Can you think of other edge / corner / boundary cases that might be applicable for testing?

2.6 Extra Information

- The starting location shall be x3000.
- The <ENTER> key is mapped to the ASCII line feed character (ASCII x0A) in the Windows LC-3 simulator.
- The input buffer must be preallocated (no writing data to unallocated memory locations).
- Your program must loop and run correctly multiple times in a row.
- Name your assembly file: `encryptor.asm`
- Your program **may NOT use subroutines** (no JSR, JSRR instructions).
- Please make sure **each student's name is in the header block** of the source code.

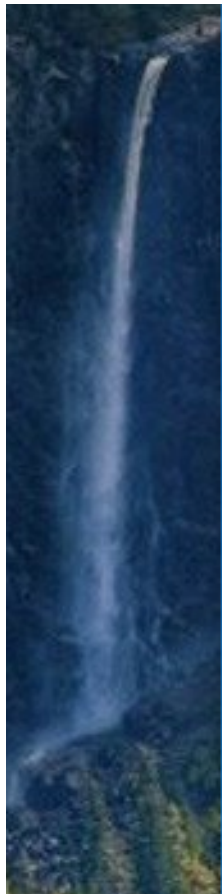
3.0 Deliverables

Run your program, execute the program testbench (see Section 2.5), and then have your Group Leader submit the following files on Canvas:

1. A screen capture of your program's output following the testbench run (the console window).
2. Your source code (`encryptor.asm` file).

Please examine the symbol table (.sym) and listing file (.lst) generated by the LC-3 assembler. Make sure you understand the relationship between the two files.

Example Testbench Output



```
Enter E)ncrypt or D)ecrypt: E
Enter Encryption Key (1-9): 3
Enter Message (<20 char, <ENTER> when done): HANCOCK
Encrypted Message: LCREQEM
Enter E)ncrypt or D)ecrypt: D
Enter Encryption Key (1-9): 3
Enter Message (<20 char, <ENTER> when done): LCREQEM
Decrypted Message: HANCOCK
Enter E)ncrypt or D)ecrypt: Z
Enter E)ncrypt or D)ecrypt: E
Enter Encryption Key (1-9): 0
Enter Encryption Key (1-9): 9
Enter Message (<20 char, <ENTER> when done): CS 131 ROCKS
Encrypted Message: K[*9;9*\WKS[
Enter E)ncrypt or D)ecrypt: D
Enter Encryption Key (1-9): 8
Enter Message (<20 char, <ENTER> when done): K[*9;9*\WKS[
Decrypted Message: BR#020#UNBJR
Enter E)ncrypt or D)ecrypt: D
Enter Encryption Key (1-9): 9
Enter Message (<20 char, <ENTER> when done): K[*9;9*\WKS[
Decrypted Message: CS 131 ROCKS
Enter E)ncrypt or D)ecrypt: █
```