Department of Electrical and Computer Engineering  
University of Texas at Austin

EE 306  
Fall, 2003  
Instructor: Tony Ambler  
TAs: Sundar Balasubramaniam, Ming Ding, Ashwin Gunasekar, Aamir Hasan, Harshal Ingole, Ritu Kar, Paroma Sen, Sujata Sharma, Raghuveer Simha

Programming Assignment 3  
Due: Midnight – 10th November, 2003

Instructions  
Name your file program3.asm. Submit the .asm file, not the .obj file.

Write your name at the beginning of the program file you submit. You will be penalized if you don’t. Be sure to write it as a comment, like so

; Last name, First name

No late submissions will be accepted. So make sure you start early.

Programs you submit must be your own work. You may discuss algorithms with your classmates. You may not discuss your program with others. Do not show your code to any of your fellow students. Plagiarism of any kind will not be tolerated.

Encryption/Decryption

The ASCII character set represents each character with an integer. It is therefore possible to perform some sort of arithmetic operations on these integer values to get a value that represents a different integer. The process of “encryption” takes a string of characters, String1, and performs some operations on the characters so that you end up with a different string, String2, which hides the original message. The process of encryption should be reversible – that is, if you perform the inverse operation on String2, you should be able to get the original message String1. This reverse process is known as decryption. Note: The arithmetic operations take two operands – one will be the character you are encrypting, and the second operand is a “key” which you can select.

Description

In this assignment you will input a string of characters from the keyboard, encrypt or decrypt it and print it back to the screen. Encryption and decryption are to use 2 keys. Details follow.
The Encryption process should have two operations:

1. Addition of a number (the addition key) to the character’s ASCII value.
2. Flipping a single bit in the result of part 1 – the second key specifies which bit to toggle.

The decryption process will involve the exact inverse, that is

1. Flipping a bit specified by the second key
2. Subtraction of the addition key

Specifications
You are required to use the LC3 input/output and string storage features (TRAP, .STRINGZ) to print out prompts asking the user for the necessary information and then read in data. You should input:

1. A single letter, either ‘e’ or ‘d’, depending on which you will either encrypt or decrypt the text you enter. You may assume that these are always entered in lower case, though you can try to make it case insensitive, purely as entertainment (??).
2. The addition key, a single integer between 0 and 9.
3. The toggle key, a single integer between 0 and 7 (since ASCII codes are 8 bit values). The bits are numbered using the standard convention. Bit 0 is the LSB and bit 7 is the MSB.
4. The text to encrypt or decrypt as a stream of characters. You should stop taking in character input when you hit the Enter key (ASCII x000A or “line feed” in the standard ASCII table)

Example display on console (your prompts should be exactly these)
Enter ‘e’ or ‘d’ to select encrypt or decrypt: e
Enter addition key: 4
Enter bit toggle key: 1
Enter text to encrypt: Hello World!
Encrypted text: Nkrrq&Yqtrj’

Example encryption and decryption

Encryption: Consider the letter H

ASCII – x48 0100 1000 ;This is the 8 bit code – LC3 sign extends it to 16 bits
Add key 4 0100 1100
Toggle bit 1 0100 1110 ;Bits numbered using standard convention B[7:0]
Encrypted value x4E ;ASCII code for the letter N

**Decryption:** Consider the letter N

ASCII – x4E 0100 1110
Toggle bit 1 0100 1100
Subtract key 4 0100 1000
Decrypted value x48 ;ASCII code for the letter H

Store the resulting string (after encryption or decryption) from location x4000 onwards and use TRAP x22 to print it out to the console.

**Hints**

Use .stringz to store the prompts that ask for user input on the console.

Toggling a single bit can be done in 2 ways

1. The X-OR function

X-ORing a bit with a 1 flips the bit. To toggle a given bit in a number, X-OR it with a number that has 0s in all but the bit position to toggle.

E.g. To toggle bit 4 in 01100110
     X-OR with 00010000
     to get 01110110

2. Masking

First determine whether the bit to toggle is 1 or 0. If it is a 1, make it a 0 by ANDing with the appropriate mask. If it is a 0, make it a 1 by ORing with the appropriate mask. If you think a little, you can use one ADD instruction (instead of OR) to change a 0 to a 1.

E.g. To toggle bit 4 in 01100110
     AND with 00010000
     to get 00000000

Result is all zeros, so bit 4 must have been a zero

In the original, change bit 4 to a 1, using OR, to get 01110110