CSCI 2450 Assembler Programming Project I

File Encryption

Deadline: April 9th, 2015 by 6:00PM [You may partner up in groups of two]

Submit your .ASM code file + your .EXE program by emailing me at azorani1@uno.edu along with the hard copy in class. Follow the submission rules outlined in the syllabus. Submissions without the hard copy will not be graded.

**Assignment**

Write an ASM program to encrypt a file using a key of 16 HEX bytes in length generated randomly. Your program should ask the user to input the file name that contains the text to be encrypted and the output file where the cipher text will be stored. The 16-byte, i.e. 128-bit HEX key (password) to encrypt the text will then be automatically and randomly generated, displayed to the user on the screen and stored in a variable so that the decryption can later take place. Remember that the XOR operator is commutative, meaning that the same key will reverse the text cipher and retrieve you the original text. Your program should be intuitive and must not crash!!! After the program finishes execution, ask the user whether or not to run it again.

YOU HAVE TO USE PROCEDURES TO IMPLEMENT THIS ASSIGNMENT. Divide your code into logical procedural units. Also, do not submit assignment that do not compile!!!

A sample run of my cipher program would be:

Please enter the name of the file you want to encode/decode > **myDocument.txt**

Name of the output file > **myCipher.txt**

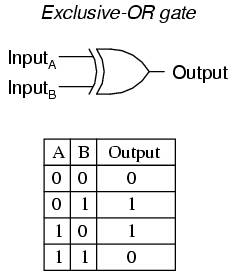
Your random 128-bit key is > 81 f8 10 a0 66 56 b8 c6 f6 e1 a9 6c 4e 9c 67 57

myDocument.txt succesfully XORed with your key into myCipher.txt

Do you wish to run the program again? (y/n) > **n**

Exiting.......

XOR Operator has a very neat commutative property most routinely used in cryptology. Remember the XOR table given inputs A and B:



Given a message **M** and an encryption key **K** , **M XOR K = C** . **C** is a ciphered message **M** . It also stands that **C XOR K = M** , therefore **M XOR K = C XOR K.**

Therefore, having a message M = "b" and key K = "t", the encrypted text C = " ►". Why?

Looking at the ASCII table a HEX of "b" is 62h which in binary is  0 1 1 0 0 0 1 0

An ASCII hexadecimal value for "t" is 74 which translated to binary is  0 1 1 1 0 1 0 0

Thus, if we apply an XOR operator on the two binary value we get  0 0 0 1 0 1 1 0

In other words, a hexadecimal 16h, which according to the ASCII table is equivalent of ► .

Think of the following example:

Given a string of text "Assembler is the best!" and a five letter secret key "world", an encryption should take place in the following manner:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Txt: | A | S | S | E | M | B | L | E | R |  | I | S |  | T | H | E |  | B | E | S | T | ! |
| Key: | W | O | R | L | D | W | O | R | L | D | W | O | R | L | D | W | O | R | L | D | W | O |
| ASCII | 6 | ↔ | ☺ | ○ | ○ | § | ♥ | ↨ | ▲ | D | ▲ | ∟ | R | ↑ | ♀ | ↕ | O | ► | ○ | ↨ | ♥ | N |

ASCII represents your newly encrypted string. If you were to run the ASCII string through your XOR loop, XORing the string with your secret key, you would get your original text back.