Description

**Subroutines:**

MODULUS – Takes R0 and R1 and performs R0 mod R1 and the output is stored in R2.

XOR – Takes R6 and R1 and performs R6 XOR R1 and the output is stored in R3.

ASCIItoBinary – Store 3 digits in ASCIIBUFF and it will store the output binary value at x4500.

Start – Prompts the main menu and takes the input of X or D or E.

Test – Tests for the input if it is X or D or E and call the corresponding subroutines.

StoreIn – Stores the input of X or D or E at x6000 for later use.

Exit – Prompts exit message and halts.

tenChar – takes the 10 character string input to encrypt and store in edToSt

Vigenere – performs XOR on the string with Key, character by character and store in edToSt

Ceaser – performs the sum of string and key character by character and performs modulus with 128 and stores in MESSAGE.

reverseCeaser - performs the negative sum of string and key character by character and performs modulus with 128 and stores in MESSAGE.

Decryption – OUTPUT the main text if the key matches.

Assumptions:

* User inputs second Digit in the to be either 0 or non-numeric ascii from the keyboard.
* User inputs last 3 digits of the key to be less than or equal to 128.
* Although my algorithm supports the functionality of adding string of less than or equal to 10 when pressed <enter>, but to encrypt it is preferred to enter a 10 character input.

Questions:

Question (during your design): what do you echo when the user types in the key?

* It echoes the key. If automatically proceeds to the next subroutine when the user inputs the 5 character key.

Question: How would you change the program if instead of at most 10 messages, you want to allow strings of any length?

* I have implemented this in my algorithm. When you press <enter> after, lets say 4 characters you press <enter> it will store the ASCII value of 10 in the memory, it will simultaneously check for this that if the user has input a character <enter> it will terminate the loop and proceed to the next step.

Question: Figure out how decryption works.

* You reverse engineer the encryption method. In this case you will back trace the method. It will perform the inverse of the methods. In this case it will subtract the Key from the value of string and mod it will 128 in Ceaser Cipher, next it will XOR it with the key to get the original string. XOR properties allow us to do this.

Question: How do you decrypt a message that has been encrypted using this scheme?

* You redo/perform the XOR on it with the same key again and it will give you the original string back again.

Question: Why are we using modulo 128 ?Should we be using 128 or a different value if the characters have to be alpha numeric or special characters (i.e., they have to be visible characters from the keyboard –cannot be space, tab etc.).

* I believe we are using 128 because there are 128 ASCII characters on normal table that can be viewed. It is better to do this since it will give us a number in range to identify.

Question: How do you shift left in LC3? How do you right shift in LC3?

* You take BITWISE AND of two numbers to shift bit left.
* You divide the binary by 2. Since there is no division we repeatedly subtract in a loop to right shift in LC3.

Pseudo code

Start code at x3000

Prompt the start message

Start( )

Prompt for input X or D or E

Store input int register 1

Test( )

Store the input at location charSt

Call StoreIn subroutine to store the input

If input = X

Clear the MESSAGE

Halt exit

If input = E

Go to Key Input EDprompt

Go to tenChar to take 10 character input to encrypt

If input = D

Go to key input EDprompt

Take MESSAGE string

Decrypt

Else

Start( )

StoreIn( )

Store the input at x6000

Exit ( )

Prompt the exit

Clear the MESSAGE

HALT

EDprompt( )

Prompt the input key criteria

Take 5 character input key

Use cloop to iterate 5 times

Store it in KeySt

If 5 character input

exLoop

RET

tenChar( )

prompt 10 char input message

take 10 character input

loop it 10 times and iterate

if cloop1 == 10 or <enter> is pressed

exLoop1

else

take character input

store it in space address

cloop1++

Vigenere( )

Take the second digit of Key stored in KeySt

Store it in R1

R4 is set to loop 10 times or when <enter> found in MESSAGE

If xloop = 10 or <enter> found

endString

else

iterate through MESSAGE

take the char store it in R0

call subroutine XOR

R0 XOR R1

Store it in MESSAGE

Xloop++

Ceaser( )

Take the last 3 digits of keySt

Store them in ACIIBUFF

Call subroutine ASCIItoBinary

Store the output binary at x4500 int

Read from MESSAGE

Use bloop to iterate through the MESSAGE

If bloop==10 or <enter> found

Go to next and store it in MESSAGE

Else

Call subroutine modulo

(keySt[ ] + int) modulo 128

Store it in MESSAGE

Modulo( )

Used from HW5

Takes modulus of 2 numbers

ASCIItoBinary( )

Taken from Textbook Calculator Example

Converts ascii to binary for addition of numbers

XOR( )

Taken from HW5

Takes XOR of 2 ascii values

Diagram

Description automatically generated