Seeam Khan
11/19/22

Project 4 Report

The program begins by printing the proper messages and getting the user input to determine whether to encrypt, decrypt, or exit. The program determines the input based on subtracting the hex value of ‘E’, ‘D’, and ‘X’ with the hex value of the input character. If the input is not one of these three, the program goes to an invalid subroutine where it prints the invalid message and goes back to the start of the program.

If input = ‘E’:

- If the input is ‘E’ the program goes into a subroutine to get the key, the subroutine stores the key as saveKey, which is x4500.
- It gets z1 first and checks if it is the range of 0 and 7 by subtracting 30 (hex value of 0) from the input for z1. If the value is positive it goes to check if the value is below 7 by subtracting 37 (hex value of 7). If the value is greater than 7 or less than 0 it goes to the invalid subroutine.
- It does the same thing for x1 and y1-y3 but with their respective ranges in hex.
- After checking the validity the 3 digit value of y is calculated by subtracting 48 from all the y’s to get the actual value of the input character in decimal form to. Y1 is multiplied by 100, y2 by 10, and y1 by 1, they are then all added together and if the value is greater than or equal to 128 we JSRR to the invalid subroutine
- After we RET to main and then JSRR to the subroutine where we get the string input, the subroutine is a loop that stores the value at R0 (after getc) at x4A00 + i. i is incremented after each loop until its greater than 9.
- We then RET back into main we then JSRR to the vignere encryption. Here we get the value at x4A00 + i, XOR it with X1 which we hold in R1, then store it in R2 then str R2 into address MESSAGE + i. This loops 10 times with a counter that starts at -10 and is incremented after every loop, when the register == 0, BRz done. When I coded the vignere subroutine I saved my R4 and R5 values at address so I could use the registers while performing the actual XOR, I loaded them back in after the XOR was complete, the same thing was done when decrypting.
- We RET back to main and then JSRR into the caesar encrypt. Here we get MESSAGE + i and add the value at that address with the y value we computed earlier when we got the input. We loop though the whole string adding y to every character.
- After this we then loop again from the begging and MOD each value with 128. (Value at MESSAGE + i) MOD 128. The resulting answer is placed back into the same address it was taken from. The MOD calculation itself adds k to (the value at saveStr + i), then subtracts 128 from a temp value (which equals the Value at

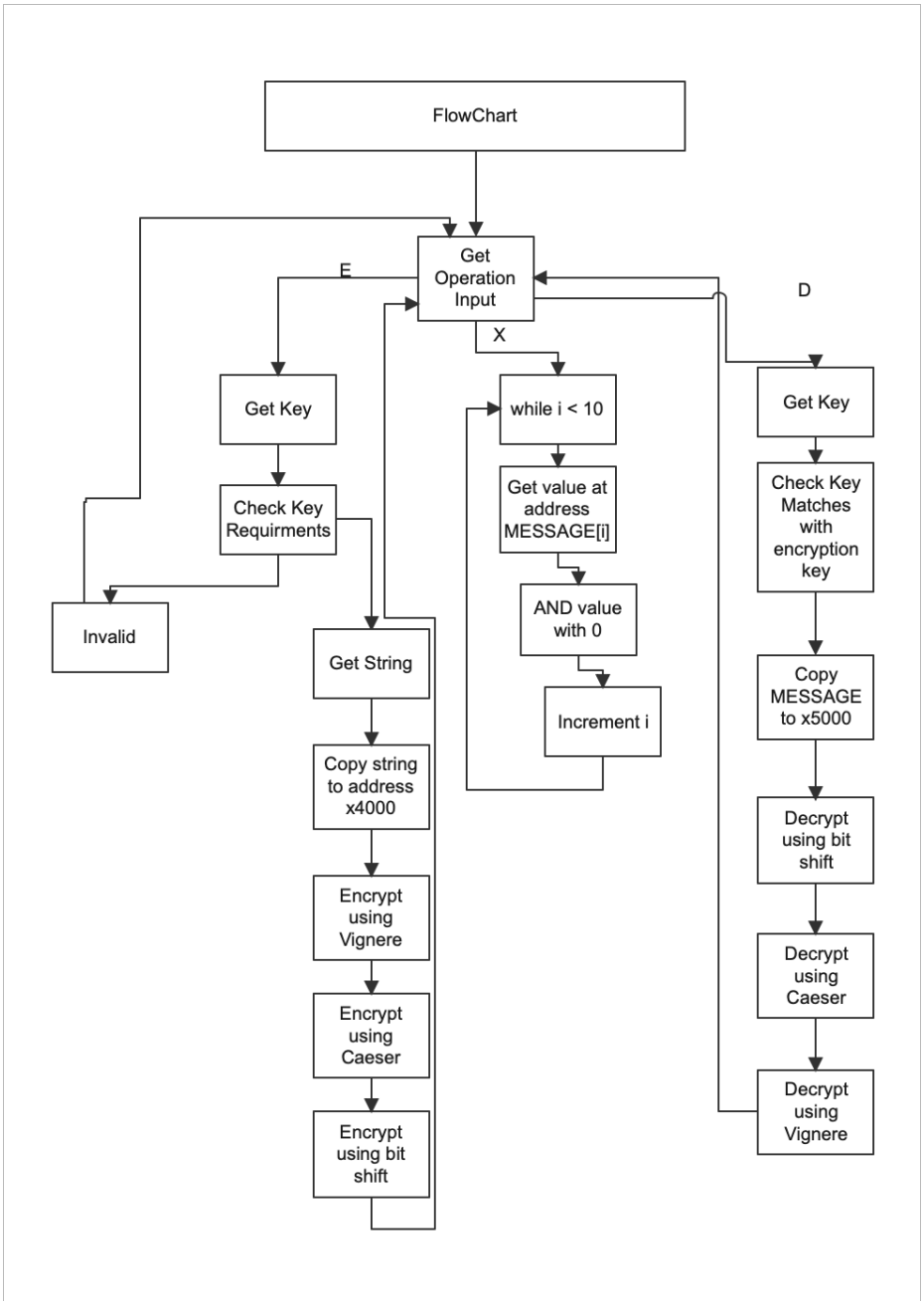
MESSAGE + i) until it is negative, it then adds 128 to the negative value and stores that in the proper address space (MESSAGE + i).
- We then RET back to main and branch to the start of the program to get new input.

If input = ‘D’:
- We go to get input, which does the same thing as it did when the input was ‘E’, but this time stores it at a different address. We then loop through both keys and check if saveKeyD(x4600) + i == saveKey(x4500) + i, it's calculated by subtracting one from the other and if the result is not 0 we branch to the invalid subroutine. This checks of the key is the same as the key we encrypted with
- We then do the same thing as we did when encrypting but first with caesar. Everything is the same about the caesar decrypt except we get the value at MESSAGE + i and store it at x5000, and when doing the MOD we do (value at MESSAGE + i) - k instead of adding them together. The actual MOD is done the same.
- After that we go back to main and JSRR into the vignere decrypt. Here XOR the same we did when encrypting, except we get the values from address x5000 + i and store them back at that same address.
- RET to main
- Once we return to main we go back to start and wait for input

If input = ‘X’:
- We JSRR into the Xin subroutine and start a loop that starts at x4000 + i and AND the value at that address with 0, to clear it. i is incremented after every loop and stops at 9. We then go back to start for new input.

Seeam Khan
11/19/22

FlowChart:



Seeam Khan
11/19/22

How to Decrypt a message that was encrypted using Vignere?:

- The vignere cipher is an encryption algorithm that XOR the char with a key value. To undo an XOR operation, you just need to do the XOR again on the encrypted value using the same key.

How to shift left in LC3?:

- Left shifting is basically multiplying the value with 2^n, where n is the number of times you want to left shift.
- The first thing is to calculate 2^n
    - We can use the same multiplication code we used in HW4 and this project, we keep adding 2 to the register its placed in, say R2, and decrement n until n = 0, say n was in R1.
- Next we calculate the product of the value and 2^n
    - Say the value is in R0 after LDR from the address
    - We
        - Loop ADD R0 R0 R0
        - ADD R2 R2 #-1
        - BRz done ; otherwise loop

How to right shift in LC3?:

- Right shifting is basically dividing the value by 2^n, where n is the number of times you want to right shift.
- We calculate 2^n the same way as left shift
- However, this time we subtract this value from the value we want to decrypt. We keep a count in another register starting at 0 and incremented every time we subtract
- Once the value at the address of the value we want to decrypt is negative we get the count value and that is our answer place it in that address.