

**Laboratory Report**

**Course ID: CPS 2390**

**Lab 3: Encryption**

**Student: Thomas Yamakaitis**

**Instructor: Dr. Jing-Chiou Liou**

**Date: 11/29/2016**

**Description**

**The objective of this lab was to create a program in LC-3 that would encrypt and decrypt a given number.**

**Procedure and Notes**

1. **My method to encrypt was to first mask out the lower 4 bits, then toggle them, add them back in, and then add the encryption key. After that, rotate all 16 bits by 8 bits and repeat it.**
2. **The only difficulty I’ve had is trying to get the program to stop at the End of Text x04. I think I am pretty close though.**

**Results and Reports**

.ORIG x3000

; Encryption

LD R0, A ; Original starting address

LD R2, B ; Storage Address

LDI R1, K ; Key

LOOP LDR R3, R0, #0

AND R5, R5, #0

ADD R5, R5, #2 ; Encryption counter

; Encrypt Routine

ELOOP AND R6, R6, #0

LD R7, MASK3

AND R6, R3, R7

ADD R6, R6, #-4 ; Check for EoT

BRz DONE

LD R7, MASK1

AND R6, R3, R7 ; Put the lower 4 bits of R3 into R6

NOT R6, R6 ; Toggle

AND R6, R6, R7 ; Only keep the lower 4 bits

LD R7, MASK2

AND R3, R3, R7 ; Set the lower 4 bits of R3 to 0

ADD R3, R6, R3 ; Put the toggled 4 bits back into R3 from R6

ADD R3, R1, R3 ; Add Key

; Rotate

AND R7, R7, #0 ; Counter for Rotation

ADD R7, R7, #8

RLOOP AND R4, R4, #0

ADD R3, R3, #0

BRp JUMP

AND R6, R6, #0

ADD R4, R6, #1

JUMP ADD R3, R3, R3

ADD R3, R3, R4

ADD R7, R7, #-1

BRp RLOOP

ADD R7, R7, #0

BRnz NEXT

; Rotate (EoT)

DONE AND R7, R7, #0 ; Counter for Rotation

ADD R7, R7, #8

RLOOP3 AND R4, R4, #0

ADD R3, R3, #0

BRp JUMP5

AND R6, R6, #0

ADD R4, R6, #1

JUMP5 ADD R3, R3, R3

ADD R3, R3, R4

ADD R7, R7, #-1

BRp RLOOP3

AND R6, R6, #0

ADD R6, R6, #5

ST R6, C

ADD R7, R7, #0

BRnz NEXT

NEXT ADD R5, R5, #-1

BRp ELOOP

STR R3, R2, #0

ADD R0, R0, #1

ADD R2, R2, #1

LDI R6, C

BRnp JUMP2

BRnzp LOOP

AND R6, R6, #0

STI R6, C

; Decryption

JUMP6 ADD R0, R0, #0 ; Used for Jump

LOOP3 ADD R0, R0, #0 ; Used for Loop

JUMP2 LD R0, B ; Original starting address

LD R2, A ; Storage Address

LDI R1, K ; Key

LOOP2 LDR R3, R0, #0

AND R5, R5, #0

ADD R5, R5, #2

; Decrypt Routine

DLOOP AND R6, R6, #0

LD R7, MASK3

AND R6, R3, R7

ADD R6, R6, #-4 ; Check for EoT

BRz DONE2

AND R7, R7, #0

NOT R7, R1

ADD R7, R7, #1

ADD R3, R3, R7 ; Subtract Key

LD R7, MASK1

AND R6, R3, R7 ; Put the lower 4 bits of R3 into R6

NOT R6, R6 ; Toggle

AND R6, R6, R7 ; Only keep the lower 4 bits

LD R7, MASK2

AND R3, R3, R7 ; Set the lower 4 bits of R3 to 0

ADD R3, R6, R3 ; Put the toggled 4 bits back into R3 from R6

; Rotate

AND R7, R7, #0 ; Counter for Rotation

ADD R7, R7, #8

RLOOP1 AND R4, R4, #0

ADD R3, R3, #0

BRp JUMP1

AND R6, R6, #0

ADD R4, R6, #1

JUMP1 ADD R3, R3, R3

ADD R3, R3, R4

ADD R7, R7, #-1

BRp RLOOP1

ADD R7, R7, #0

BRnz NEXT1

; Rotate (EoT)

DONE2 AND R7, R7, #0 ; Counter for Rotation

ADD R7, R7, #8

RLOOP2 AND R4, R4, #0

ADD R3, R3, #0

BRp JUMP4

AND R6, R6, #0

ADD R4, R6, #1

JUMP4 ADD R3, R3, R3

ADD R3, R3, R4

AND R6, R6, #0

ADD R6, R6, #5

ST R6, C

ADD R7, R7, #-1

BRp RLOOP2

NEXT1 ADD R5, R5, #-1

BRp DLOOP

STR R3, R2, #0

ADD R0, R0, #1

ADD R2, R2, #1

LDI R6, C

BRnp FINISH

BRnzp JUMP2

FINISH HALT

A .FILL x3110

B .FILL x3120

C .FILL x3130

MASK1 .FILL x000F

MASK2 .FILL xFFF0

MASK3 .FILL x00FF

K .FILL x3100

Z .FILL x0000

.END



