C2C Mark Demore II

Capt Warner

ECE281

Lab 2

main:

mov key\_length, r13

mov mess\_length, r14

mov #out\_loc, r15

mov #mess\_start, r10

mov #key\_loc, r9

mov #key\_len, r8

decryptByte:

xor.b r11, r12

ret

decryptMessage:

cmp #0, r14

jz cpu\_trap <- or ret at end and do cmp jnz statements in main

cmp #0, r13

jz reset\_key\_loc

mov.b @r10, r11

mov.b @r9, r12

call decryptByte

mov r12, @r15

decd r14

decd r8

incd r10

incd r9

incd r15

jmp decrypt message

Decrypt Byte:

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Purpose** |
| R11 | x | Provide encrypted message |
| R12 | R12 | Provide encryption key, store decrypted message |

Decrypt Byte:

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Purpose** |
| R14 | x | Provide message length, track location in message |
| R13 | x | Provide key length, track location in key |
| R15 | x | Provide location to store output |
| R9 | x | Provide key starting location |
| x | R12 | Store encryption key byte |
| x | R11 | Store encrypted byte |
| R8 | x | Track key location |
| R10 | x | Track message location |

1. How will you test your code? *Read the Notes section for a hint*.

I will test my code by entering a simple test string in memory and attempting to encrypt it. Once I verify that the message has been altered, I will attempt to decrypt the output from the first run of the program with the same key. If the output of this second run matches the initial string, the program works. I will also set various breakpoints for debugging.

1. Identify any limitations of your proposed implementation.

Some downsides to my proposed implementation are the number of registers used and the amount of input required to decrypt a message

1. Is there anything about this lab that you would like me to address at the beginning of class?

How is input supposed to be handled and moved into appropriate registers, as an immediate, symbolically, etc.?