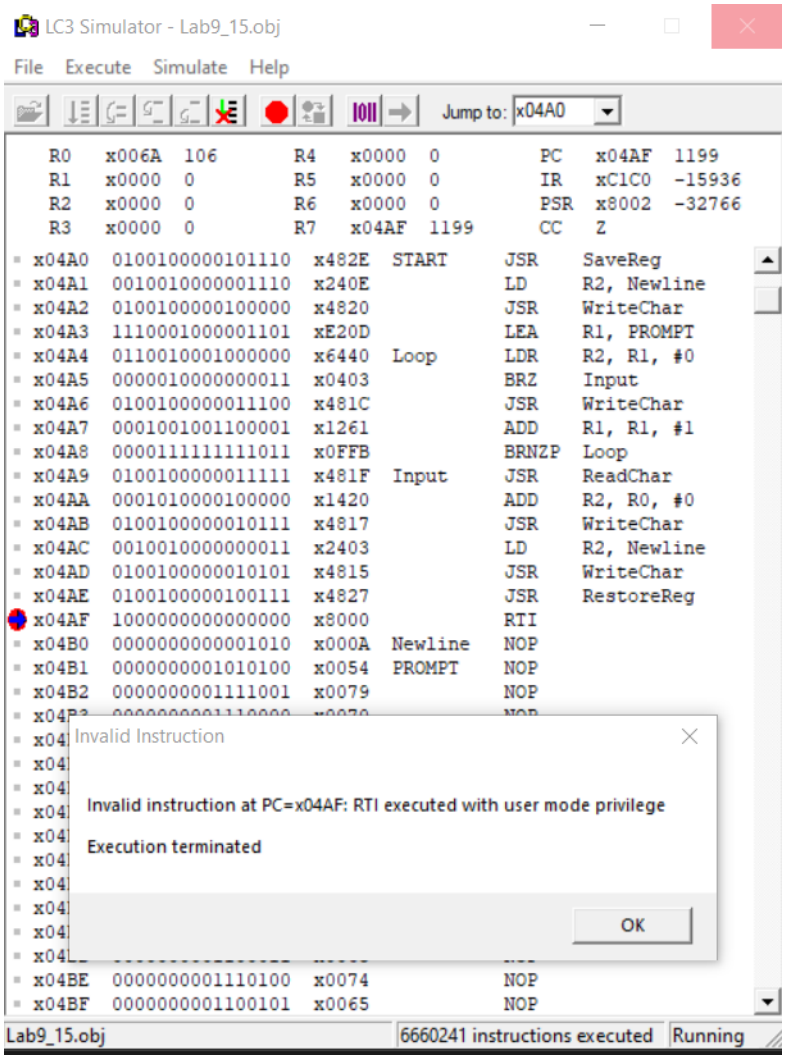
Joseph Godinez

Professor Oscar Ho

Computer Science 240-20

November 23, 2021

1. Screenshot of what happens when the code is compile and run



2. Screenshot of the code and comments

```
.ORIG x04A0

        START JSR SaveReg       ; We are starting at our subroutine SaveReg
        LD R2,Newline           ; Load local data to register 2 from Newline
        JSR WriteChar           ; Jump to subroutine WriteChar
        LEA R1,PROMPT           ; Load the address from register 1 to PROMPT


        Loop LDR R2,R1,#0       ; Get next prompt char
        BRz Input               ; Conditional for zeros and goes to Input

        JSR WriteChar           ; Jump to Subroutine WriteChar
        ADD R1,R1,#1            ; Add the value of register 1 from registeer 1 with 1
        BRnzp Loop              ; Conditional for negatives, zeros, and positives brings it to Loop


Input   JSR ReadChar            ; Jump to Subroutine ReadChar
        ADD R2,R0,#0            ; Move char to R2 for writing
        JSR WriteChar           ; Echo to monitor


        LD R2, Newline          ; Load local data from Register 2 to Newline
        JSR WriteChar           ; Jump to subroutine WriteChar
        JSR RestoreReg          ; Jump to subroutine RestoreReg
        RTI                     ; RTI terminates the trap routine

Newline .FILL x000A

PROMPT .STRINGZ "Type a character."


WriteChar       LDI R3,DSR      ; Indirectly load and read the data from Register 3 to DSR
                BRzp WriteChar  ; Conditional for zeros and positive if the condition is true go to WriteChar
                STI R2,DDR      ; Indirectly store data from register 2 to DDR
                RET             ; JMP R7 terminates subroutine

DSR .FILL xFE04                 ; DSR fills xFE04
DDR .FILL xFE06                 ; This one also fills xFE06
```

```
ReadChar        LDI R3,KBSR     ; Indirectly load and read data from Register 3 to KBSR
                BRzp ReadChar   ; Conditional for zeros and positives if true go to ReadChar
                LDI R0,KBDR     ; Indirectly load and read data from Register 0 to KBDR

        RET                     ; Return
KBSR .FILL xFE00                ; Fills the loaded data into xFE00
KBDR .FILL xFE02                ; Fills the loaded data into xFE02


SaveReg ST R1,SaveR1            ; Locally store the data into SaveR1 to SaveR6
        ST R2,SaveR2            ;
        ST R3,SaveR3            ;
        ST R4,SaveR4            ;
        ST R5,SaveR5            ;
        ST R6,SaveR6            ;

        RET                     ; Return

RestoreReg      LD R1,SaveR1    ; Locally load and read the data into SaveR1 to SaveR6
                LD R2,SaveR2    ;
                LD R3,SaveR3    ;
                LD R4,SaveR4    ;
                LD R5,SaveR5    ;
                LD R6,SaveR6    ;

        RET                     ; Return

        SaveR1 .FILL x0000      ; Fills the x0000 from what is from SaveR1 to SaveR6
        SaveR2 .FILL x0000      ;
        SaveR3 .FILL x0000      ;
        SaveR4 .FILL x0000      ;
        SaveR5 .FILL x0000      ;
        SaveR6 .FILL x0000      ;

 .END
```

3. So from my understanding of the code we have 3 different subroutines in the program. Where one is a keyboard subroutine which waits until a Char is inputed

and the second subroutine is a display subroutine. Last I think the third subroutine stores the Char into the register from 1 to 6. So I think that the program purposely creates the error message because when we go to the line of RTI they call it an invaild instruction.

4. So the difference between 9.12 and 9.13 is that in 9.12 the RTI is used to return from trap while the RTI in 9.13 was used to terminate the trap subroutine. The reason why I think so is because in figure 9.13 the subroutine is terminated before the program can complete. So in figure 9.12 instead of terminating the program it just returns the trap.