Assignment 3

I/O Interrupt

The purpose of this assignment is to show how interrupt-driven Input / Output can interrupt a running program, execute the interrupt service routine, then return to the interrupted program and pick up exactly where it left off (just as if nothing had happened).

In this assignment, we will use the keyboard as the input device for interrupting the running program.

This assignment consists of THREE parts:

1. The User Program
Your user program will be continually producing two different pattern based on the keyboard input.

One pattern is called "** checkerboard". You will be alternately outputting the following two different lines. The first one consists of the pattern "** followed by **four** spaces" **eight** times. The second one consists of **three** spaces and the pattern "** followed by **four** spaces" **seven** times.

```
**    **    **    **    **    **    **    **
   **    **    **    **    **    **    **
**    **    **    **    **    **    **    **
   **    **    **    **    **    **    **
**    **    **    **    **    **    **    **
   **    **    **    **    **    **    **
```

Another pattern is called "## checkerboard". You will be alternately outputting the following two different lines. The first one consists of the pattern "## followed by **four** spaces" **eight** times. The second one consists of **three** spaces and the pattern "## followed by **four** spaces" **seven** times.

```
##    ##    ##    ##    ##    ##    ##    ##
   ##    ##    ##    ##    ##    ##    ##
##    ##    ##    ##    ##    ##    ##    ##
   ##    ##    ##    ##    ##    ##    ##
##    ##    ##    ##    ##    ##    ##    ##
   ##    ##    ##    ##    ##    ##    ##
```

Your program should start with " checkerboard".** To ensure the output on the screen is not too fast to be seen by the naked eye, the user program should include a piece of code that will count down from 2500 to 0 each time a line is output on the monitor. A simple way is using the following subroutine DELAY:

```
DELAY     ST R1, SaveR1
          LD R1, COUNT
REP       ADD R1, R1, #-1
          BRp REP
          LD R1, SaveR1
          RET
COUNT     .FILL #2500
SaveR1    .BLKW 1
```

2. The Keyboard Interrupt Service Routine

The keyboard interrupt service routine will simply write to the screen **TEN** times whatever key the person sitting at the keyboard typed, followed by a linefeed. AND CHANGE THE PATTERN THAT **WILL** BE PRINTED TO THE MONITOR AFTER THE USER PRESSED THE KEY.

IMPORTANT: You may not use any TRAP instructions in your interrupt service routine. To display a character to the screen, you **MUST** poll the DSR and then write to the DDR. You may not call TRAP x21 (OUT), or use any of the other TRAP routines. If you use TRAP in the interrupt service routine or if you do not properly poll the DSR before writing to the DDR, your program is not correct and will fail our testing even though it may appear to work when you test it. You may use any TRAP instructions you wish in your user program.

Hint: Don't forget to save and restore any registers that you use in the interrupt service routine.

3. The Operating System Enabling Code
Unfortunately, we haven't installed Windows, Linux or any operating systems on the LC-3 machine, so we **NEED** you to do the following three enabling actions in your user program BEFORE your user program starts outputting the "** checkerboard" or "## checkerboard". Normally, these would be done by the operating system.

1) Normally, the operating system would have previously set up some stack space so that the PC and PSR can be pushed when an interrupt is encountered. (As is known, when the service routine executes RTI, both PC and PSR will be popped, returning the machine to the interrupted program.) Since there is no operating system, please initialize R6 to x3000, indicating an empty stack.
2) The operating system normally establishes the interrupt vector table to contain the starting addresses of the corresponding interrupt service routines. You will have to do that for the keyboard interrupt. The starting address of the interrupt vector table is x0100 and the corresponding interrupt vector for the keyboard is x80. It is necessary for you to only provide the one entry in the interrupt vector table that is needed for this programming assignment.
3) Finally, the operating system normally would set the IE (Interrupt Enable) bit of the KBSR. You will have to do that as well.

Your Job

Your job will be to write both the user program augmented with the interrupt enabling code described above and the keyboard interrupt service routine.

The user program can be named as **user_program.asm** and will be in the form of:

```
.ORIG x3000
-- --- ; initialize the stack pointer
...
-- ---
```

```
        -- --- ; set up the keyboard interrupt vector table entry
        ...
        -- ---
        -- --- ; enable keyboard interrupts
        ...
        -- ---
        -- --- ; start of actual user program to print the checkerboard
        ...
        -- ---
        .END
```

The interrupt service routine can be named as **interrupt_service_routine.asm** and will be in the form of:

```
        .ORIG x2000
        -- --- ; the code
        ...
        -- ---
        RTI
        -- --- ; buffer space as required
        ...
        -- ---
        .END
```

NOTE: The Linux/Unix LC-3 simulator does **NOT** support interrupts. Therefore, you **MUST** use the Windows LC-3 simulator for this assignment.

To test your program, we suggest you load the interrupt service routine first, and then the user program and execute finally.

Since your user program contains an infinite loop, to stop the program, you must press the "Stop Execution" button in the simulator.

Submit Your Program

Tho programs should be appended to the end of your report. You should write a report for your program to briefly describe your idea and method for this assignment.