LAB 3

1. INTRODUCTION

This lab requires to write a user program and an interrupt service routine. The former should be able to print two different patterns repeatedly. One is "** checkerboard". The program should be alternately printing the following two different lines. The first one consists of the pattern "** " eight times. And the second one consists of three spaces and the pattern "** " seven times.

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

The other pattern is called "## checkerboard". The program should be alternately printing the following two lines. The first one consists of the pattern "## " eight times. And the second one consists of three spaces and the pattern "## " seven times.

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

The keyboard interrupt service routine will simply output to the monitor ten times whatever key the person typed, followed by a linefeed. And change the pattern that will be printed to the monitor after the user pressed the key.

This lab is meant to let us experience how interrupt-driven I/O works.

2. ALGORITHM

To finish the tasks, the algorithm should be like:

1. Do the initialization;

2. Output a pattern in two different formations alternately and continually;

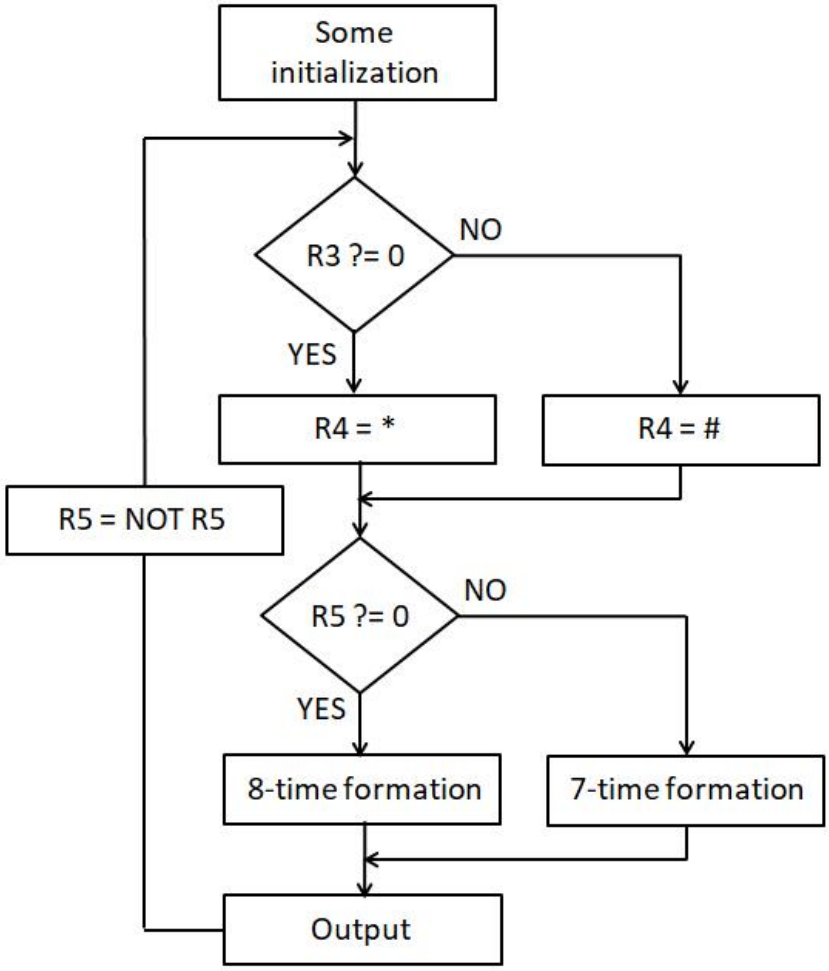
3. When a key is typed, execute the interrupt service routine;

4. After finishing the interrupt service routine, keep outputting another pattern in two different formations alternately and continually;

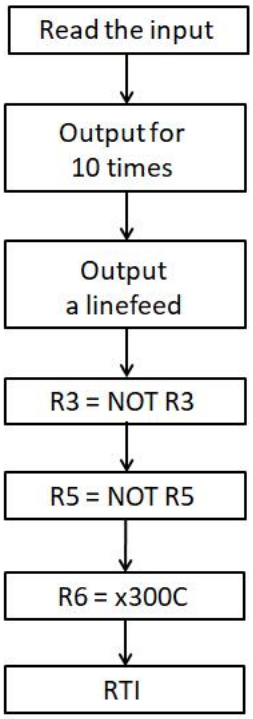
In the user program, R0 stores the ASCII code of the character to be output. R1 counts the times the pattern should be output. R2 counts the times that a certain character should be output. R3 signals the pattern: star or pound. R4 stores the ASCII code of a star or a pound temporarily. R5 signals the formation.

Then in the interrupt service routine, the routine changes the value in R3, the value in R5 and the value of the PC in the supervisor stack to guarantee the correct output.

The diagram of the user program is shown as follow:

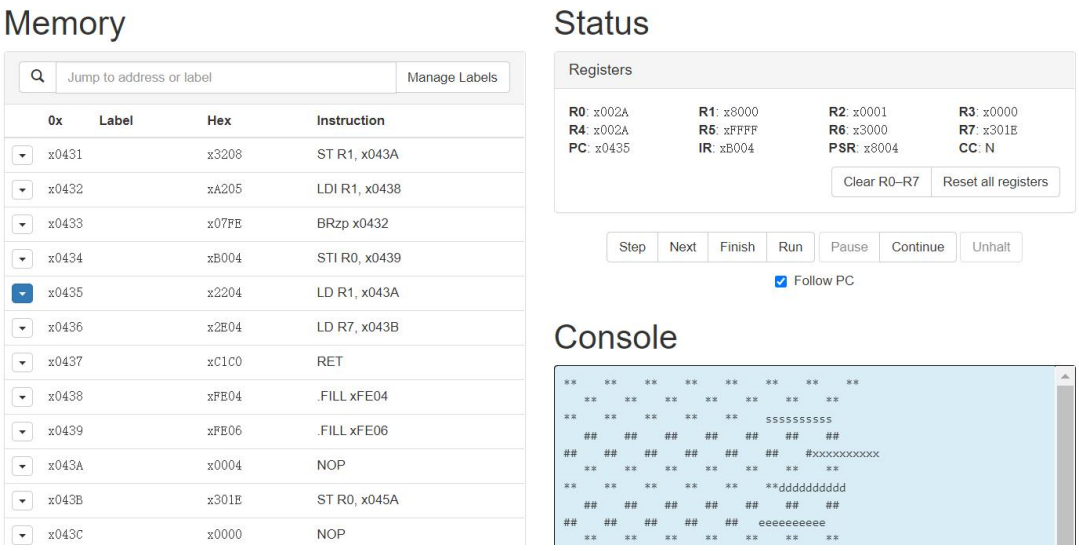


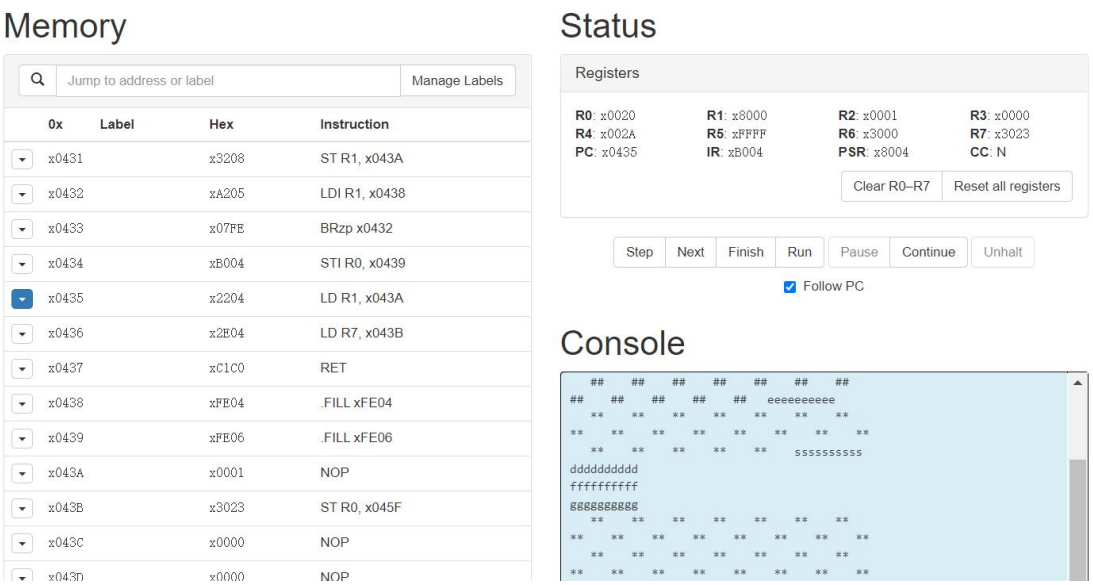
The diagram of the interrupt service routine is shown as follow:



3. Testing Result

Since the test results are not so appropriate to display by a table, I will just attach some screenshots.





4. Discussion and Experience

When writing the program, I found that the most difficult part of this lab is to return to the program interrupted. It's hard to guarantee nothing will be output after the ten typed characters or in the new line, because I found that NULL would occupy a space, which makes it look like a space in the console. I think it's a defection of the web simulator because according to my experience of using other compilers, a NULL should occupy no space in the console. At last, I change the value of PC stored in the supervisor stack to make sure that nothing excessive is output.

Also, while writing the program, I found that I sometimes muddled up store instructions with load instructions. I should pay more attention to distinguish them.

Appendix: Source Code

User program:

```
        .ORIG   x3000
; initialization
        LD      R6, INIPT       ; initialize the stack pointer
        LDI     R1, KBSR
        LD      R2, SETIE
        NOT     R1, R1
        NOT     R2, R2
        AND     R1, R1, R2
        NOT     R1, R1
        STI     R1, KBSR        ; enable keyboard interrupts
```

```
        LD      R1, SRT
        STI     R1, VTABLE      ; set up the table entry
        AND     R3, R3, #0      ; clear R3
        AND     R5, R5, #0      ; clear R5
; output new lines
LINE    AND     R1, R1, #0
        ADD     R1, R1, #8      ; signal the times of the pattern
        AND     R2, R2, #0      ; signal the times of the character
        LD      R4, STAR        ; *
        ADD     R3, R3, #0      ; signal the pattern
        BRz     SKIP1
        LD      R4, POUND       ; #
SKIP1   ADD     R5, R5, #0      ; signal the formation
        BRz     LOOP
        LD      R0, SPACE
        ADD     R2, R2, #3
OUT3SP  TRAP    x21
        ADD     R2, R2, #-1
        BRp     OUT3SP          ; output 3 ' ' first
        ADD     R1, R1, #-1     ; the pattern should repeat 7 times

LOOP    ADD     R0, R4, #0
        ADD     R2, R2, #2
OUTC0   TRAP    x21             ; output the character
        ADD     R2, R2, #-1
        BRp     OUTC0
        LD      R0, SPACE
        ADD     R2, R2, #4
OUTSP   TRAP    x21             ; output space
        ADD     R2, R2, #-1
        BRp     OUTSP
        LD      R2, COUNT
REP     ADD     R2, R2, #-1     ; delay
        BRp     REP
        ADD     R1, R1, #-1
        BRp     LOOP
        LD      R0, ENTER
        TRAP    x21             ; output an enter
        NOT     R5, R5          ; change the formation
        BRnzp   LINE
COUNT   .FILL  #2500
INIPT   .FILL  x3000
SETIE   .FILL  x4000
VTABLE  .FILL  x0180
```

```
SRT     .FILL  x2000
STAR    .FILL  x002A
POUND   .FILL  x0023
ENTER   .FILL  x000A
SPACE   .FILL  x0020
KBSR    .FILL  xFE00
        .END
```

Interrupt service routine:

```
        .ORIG   x2000
READKB  LDI     R1, KBSR
        BRzp    READKB
        LDI     R0, KBDR        ; read the input
        AND     R2, R2, #0
        ADD     R2, R2, #10
LOOPD   LDI     R1, DSR
        BRzp    LOOPD
        STI     R0, DDR
        ADD     R2, R2, #-1
        BRp     LOOPD           ; output for 10 times
        LD      R0, ENTER1
TESTD   LDI     R1, DSR
        BRzp    TESTD
        STI     R0, DDR         ; output a linefeed
        NOT     R3, R3          ; change the pattern
        NOT     R5, R5          ; change the formation
        LD      R1, TOLINE
        STR     R1, R6,#0       ; redirect
        AND     R0, R0, #0      ; clear R0
        RTI
TOLINE  .FILL  x300C
ENTER1  .FILL  x000A
KBSR    .FILL  xFE00
KBDR    .FILL  xFE02
DSR     .FILL  xFE04
DDR     .FILL  xFE06
        .END
```