Lab3

Name:苗昊田 ID:3190104547

Introduction

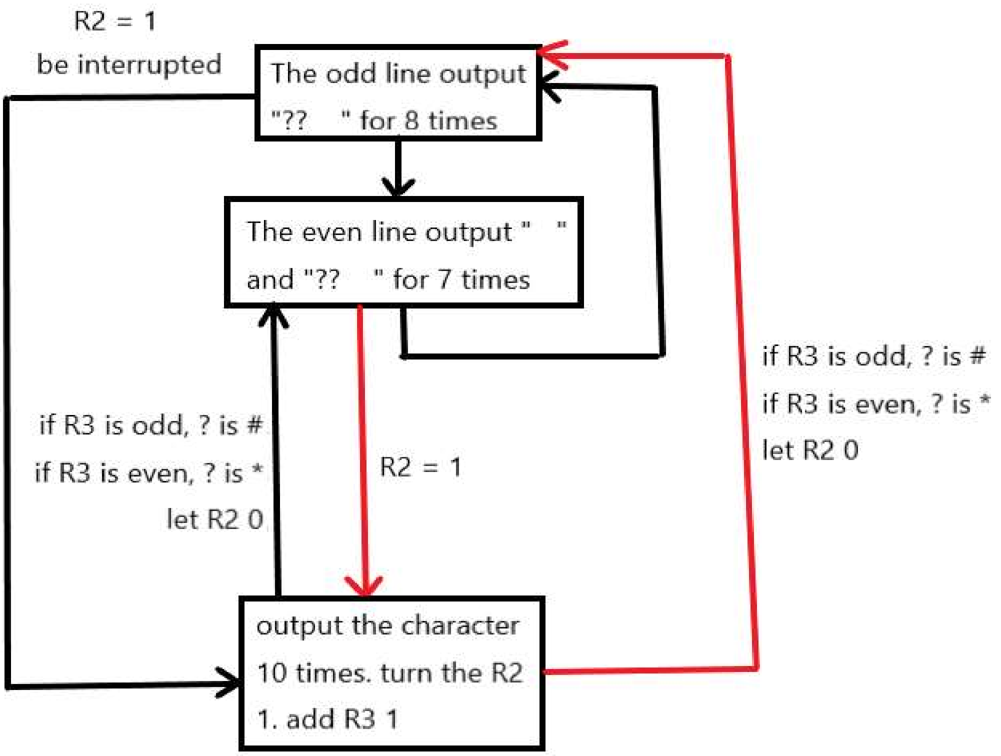
This program provides a way to show how interrupt-driven input/output can interrupt a running program, execute the interrupt service routine, then return to the interrupted program, and provides some test results.

Algorithm

First we initialize the stack pointer, set up the keyboard interrupt vector table entry and enable keyboard interrupts.

Second we start of actual user program to print the checkerboard, we use two loop to output “** ” and “ **”, and once being interrupted, we change the checkerboard between “*” and “#”

In the interruption, we let R2 is 1 as a sign of interruption and initialize R2 as 0 when the program return to the running program. At the same time we add R3 one every interruption to decide “*” or “#” we use next time. And also we output the character we input for ten times.


R2 = 1
be interrupted
The odd line output "?? " for 8 times
The even line output " " and "?? " for 7 times
if R3 is odd, ? is #
if R3 is even, ? is *
let R2 0
R2 = 1
if R3 is odd, ? is #
if R3 is even, ? is *
let R2 0
output the character 10 times. turn the R2 1. add R3 1

Testing Result

____(1)

```
        ADD R2, R2, #0
```

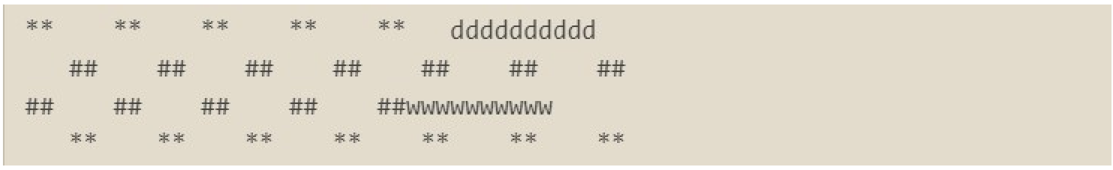
____(2)

```
        BRp CHANGEASCII
```

____(3)

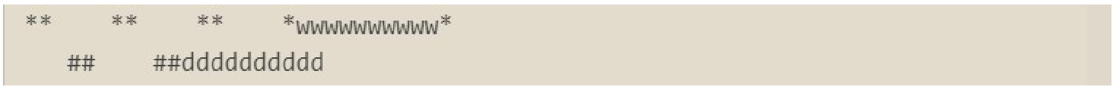
Above is the test of interruption before every output.

If the interruption occurs at (1) or (3), then the program will go to next line and change the checkerboard like:



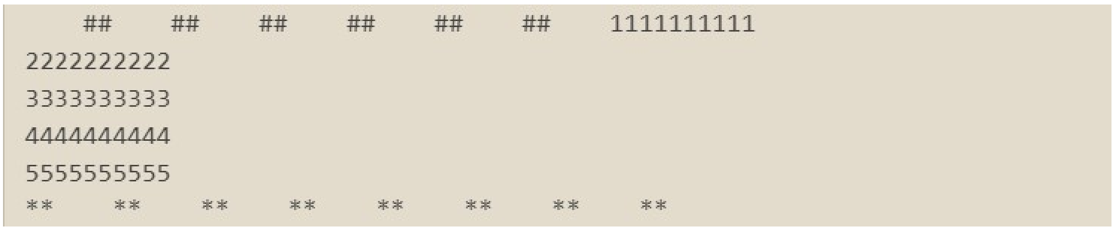
```
**    **    **    **    **   dddddddddd
   ##    ##    ##    ##    ##    ##    ##
##    ##    ##    ##    ##wwwwwwwwww
   **    **    **    **    **    **    **
```

And if the interruption occurs at (2) between two checkerboard output, then the program will output another checkerboard then go to next line and change the checkerborad like:



```
**    **    **    *wwwwwwwwww*
   ##    ##dddddddddd
```

And if we pause it and input 12345 at a time and run again, and the program will output like this:



```
    ##    ##    ##    ##    ##    ##    1111111111
2222222222
3333333333
4444444444
5555555555
**    **    **    **    **    **    **    **
```

Discussion and Experience

Though this experiment, I gain a deeper knowledge about interruption and know more about how the computer works to handle all kinds of signal of operation. The problem I met is that because the KBSN is not be initialized at the end of interruptions so every time the program should be back to the main program though RTI, the program would go to the interruption again, and finally I realize I should initialize the KBSN or the interruption would be triggered all the time. And after that I managed to work it out.

APPENDIX:SOURCECODE

R0: character input or output   R1: counter for checkerboard outputs   R2: sign of interruption
R3: sign of checkerboard   R5: counter for character outputs   R6: pointer of stack

```
.ORIG    x3000
        LD R6, STACKER
```

```
          LD R1, TARGET
          STI R1, INTLOC
          AND R2, R2, #0
          AND R3, R3, #0
PRELOOP1 LD R1, A
          STI R1, KBSN
          ADD R2, R2, #0
          BRp CHANGEASCII1
          AND R1, R1, #0
          ADD R1, R1, #8
LOOP1      LD R0, ASCII
          ADD R2, R2, #0
          BRp CHANGEASCII1
          OUT
          ADD R2, R2, #0
          BRp CHANGEASCII1
          OUT
          ADD R2, R2, #0
          BRp CHANGEASCII1
LOOP2     LD R0, BLOCK
          ADD R2, R2, #0
          BRp CHANGEASCII1
          OUT
          ADD R2, R2, #0
          BRp CHANGEASCII1
          OUT
          ADD R2, R2, #0
          BRp CHANGEASCII1
          OUT
          ADD R2, R2, #0
          BRp CHANGEASCII1
          OUT
          ADD R2, R2, #0
          BRp CHANGEASCII1
          ADD R1, R1, #-1
          BRp LOOP1
          LD R0, COUNT
REP1      ADD R0, R0, #-1
          BRp REP1
          ADD R2, R2, #0
          BRp CHANGEASCII1
          LD R0, CHANGE
          OUT
PRELOOP2 LD R1, A
```

```
        STI R1, KBSN
        ADD R2, R2, #0
        BRp CHANGEASCII2
        AND R1, R1, #0
        ADD R1, R1, #7
        LD R0, BLOCK
        OUT
        OUT
        OUT
LOOP3   LD R0, ASCII
        ADD R2, R2, #0
        BRp CHANGEASCII2
        OUT
        ADD R2, R2, #0
        BRp CHANGEASCII2
        OUT
        ADD R2, R2, #0
        BRp CHANGEASCII2
LOOP4   LD R0, BLOCK
        ADD R2, R2, #0
        BRp CHANGEASCII2
        OUT
        ADD R2, R2, #0
        BRp CHANGEASCII2
        OUT
        ADD R2, R2, #0
        BRp CHANGEASCII2
        OUT
        ADD R2, R2, #0
        BRp CHANGEASCII2
        OUT
        ADD R2, R2, #0
        BRp CHANGEASCII2
        ADD R1, R1, #-1
        BRp LOOP3
        LD R0, COUNT
REP2    ADD R0, R0, #-1
        BRp REP2
        ADD R2, R2, #0
        BRp CHANGEASCII2
        LD R0, CHANGE
        OUT
        BR PRELOOP1
CHANGEASCII1
```

```
          AND R3, R3, #1
          BRz NEXT1
          LD R2, CHECKBD2
          ST R2, ASCII
          LD R0, ASCII
          AND R2, R2, #0
          BR PRELOOP2
NEXT1     LD R2, CHECKBD1
          ST R2, ASCII
          LD R0, ASCII
          AND R2, R2, #0
          BR PRELOOP2
CHANGEASCII2
          AND R3, R3, #1
          BRz NEXT2
          LD R2, CHECKBD2
          ST R2, ASCII
          LD R0, ASCII
          AND R2, R2, #0
          BR PRELOOP1
NEXT2     LD R2, CHECKBD1
          ST R2, ASCII
          LD R0, ASCII
          AND R2, R2, #0
          BR PRELOOP1
COUNT       .FILL #2500
STACKER .FILL x3000
INTLOC    .FILL x0180
TARGET    .FILL x2000
A            .FILL x4000
KBSN        .FILL xFE00
ASCII     .FILL x002A
CHECKBD1 .FILL x002A
CHECKBD2 .FILL x0023
BLOCK      .FILL x0020
CHANGE    .FILL x000A
           .END
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _(int)
.ORIG x2000
          ADD R2, R2, #1
          ADD R3, R3, #1
          ST    R6, SAVER6
TAKE      LDI R6, KBSN
          BRzp TAKE
```

```
            LDI R6, KBDN
            AND R5, R5, #0
            ADD R5, R5, #10
OUTPUT LDI R4, DSR
            BRzp OUTPUT
            STI R6, DDR
            ADD R5, R5, #-1
            BRp OUTPUT
PUTCHANGE LDI R4, DSR
            BRzp PUTCHANGE
            LD R4, CHANGE
            STI R4, DDR
LD   R6, SAVER6
RTI
SAVER6 .BLKW #1
KBSN .FILL xFE00
KBDN .FILL xFE02
DSR    .FILL xFE04
DDR    .FILL xFE06
CHANGE    .FILL x000A
.END
```