LAB 3 Report

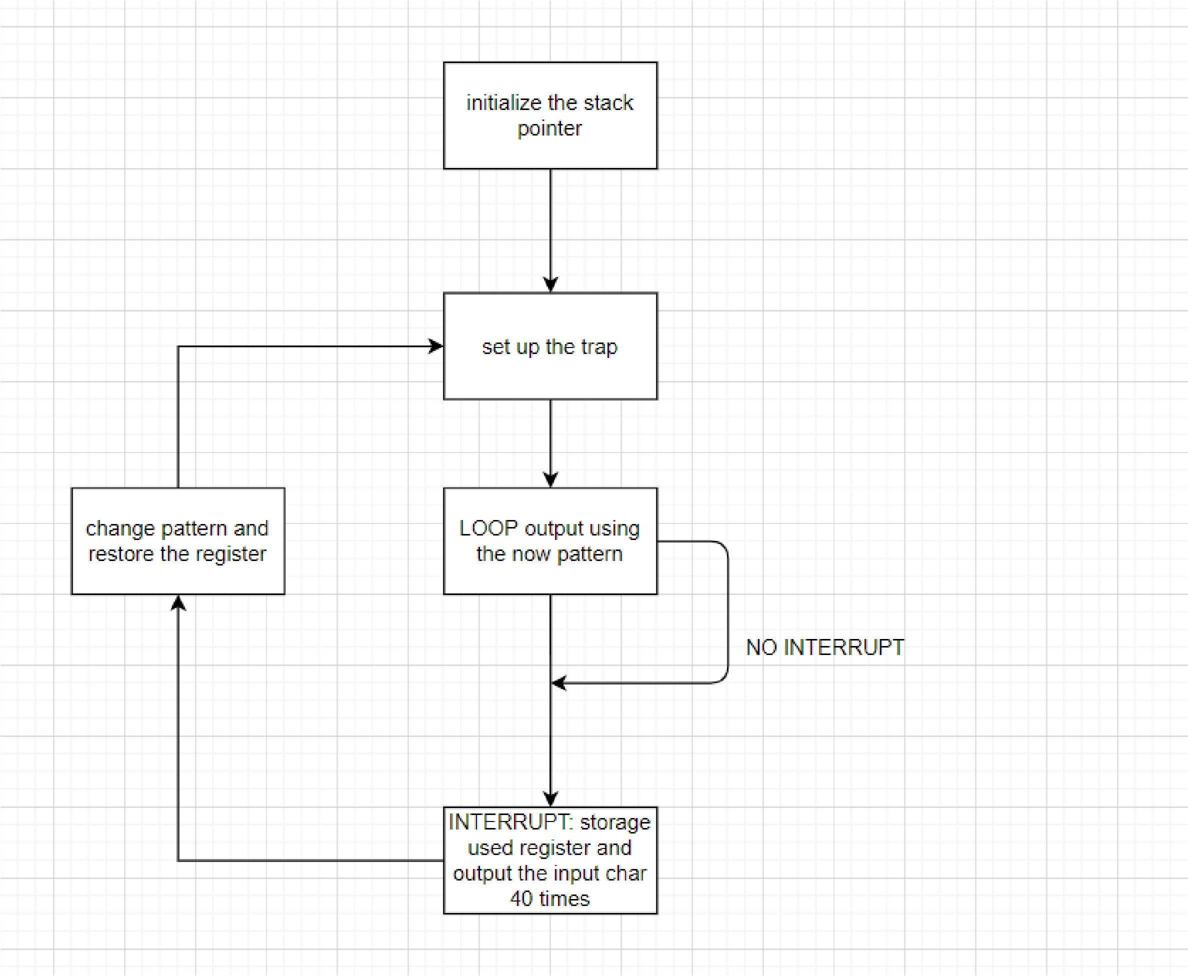
Algorithm

The algorithm of user_program

- initialize the stack pointer
- set up the keyboard interrupt vector table entry
- LOOP to continue two lines with some Functions
    - one thing to remember is when we prepare to output a flag, we should check the R3 which contain the program whether be interrupted, we change our flag depend on R3 and store in R2
    - to put evrey flag we load R0 From R2 where contain the char.

The algorithm of interrupt_service_program

- begin at 0x2000
- the interrupt program will need R0,R1,R2 , so we store the R0,R1,R2 in Stack which ptr is R6;
- then we poll the KBSR whether ready to read the char ,if finished we store the value of KBDR to R0
- so we also poll the DSR whether ready to put the char, but this time we put 10 times and 1 '\a'
- then we change R3 which decide the out flag
- finally we load the R0,R1,R2 and RTI


initialize the stack pointer
set up the trap
change pattern and restore the register
LOOP output using the now pattern
NO INTERRUPT
INTERRUPT: storage used register and output the input char 40 times

Code

```
.ORIG x0200         ;system booting code
        LD  R6,OS_SP
        LD  R0,USER_PSR     ;push USER_PSR
        ADD R6,R6,#-1
        STR R0,R6,#0
        LD  R0,USER_PC      ;push USER_PC
        ADD R6,R6,#-1
        STR R0,R6,#0
        LD  R0,KBSR_IE      ;make KBSR[14] equal to 1
        STI R0,KBSR
        LD  R0,KBI_ADDR     ;intruction interrupt tabel vector
        STI R0,KBI_INV
        AND R0,R0,#0
        RTI
        AND R1,R1,#0
        AND R2,R2,#0
        AND R3,R3,#0
        ADD R3,R3,#0
        AND R4,R4,#0
OS_SP       .FILL   X3000
USER_PSR    .FILL   X8002
USER_PC     .FILL   X3000
```

```
KBSR        .FILL   XFE00
KBSR_IE     .FILL   X4000
KBDR        .FILL   XFE02
KBI_ADDR    .FILL   X0800
KBI_INV     .FILL   X0180
        .END
;----------------------------------------------------------------
        .ORIG x0800         ;interrupt service routine
        ST   R0,SaveR0
        ST   R1,SaveR1

HIT     LDI  R0,KBSR_       ;check KSBR[15]
        BRzp HIT
        LDI  R0,KBDR_

CHECK   LD   R1,ENTER       ;check whether R0 equals to x000A, if so , output number -1
        ADD  R1,R1,R0       ;when r0 is 0 , then we needn't subtract 1
        BRnp #6
        LD   R0,SaveR0
        ADD  R1,R0,#-16
        ADD  R1,R1,#-16
        ADD  R1,R1,#-16
        BRz  #1
        ADD  R0,R0,#-1

DISP    LDI  R1,DSR_
        BRzp DISP
        STI  R0,DDR_

        AND  R2,R2,#0       ;restart the output to make sure that there are 40 output
        ADD  R2,R2,#10

        ST   R0,SaveR0      ;output the ENTER
        LD   R0,StrEnter
        trap x21
        LD   R0,SaveR0
        LD   R1,SaveR1
        RTI
ADDF
       ADD  R0,R0,#-1
       ADD R0,R0,#1
        AND R1,R1,#0
        AND R2,R2,#0
        AND R3,R3,#0
        ADD R3,R3,#0
        AND R4,R4,#0
SaveR0  .FILL x0000
SaveR1  .FILL X0000
KBSR_   .FILL XFE00
KBDR_   .FILL XFE02
DSR_    .FILL XFE04
DDR_    .FILL XFE06
ENTER   .FILL XFFF6
StrEnter   .FILL   x000A
         .END
;-----------------------------------------------------------
```

```
        .ORIG x3000         ;User Program
        LD R0,Ini_R0        ;Initial register
        AND R1,R1,#0
        AND R2,R2,#0
        AND R3,R3,#0
        ADD R3,R3,#2
        AND R4,R4,#0

LOOP_1  JSR JUDGE_NUMBER
        ADD R3,R3,#0
        BRp #1
        LD  R0,StoreR0_0    ;reload the number

        JSR JUDGE_NUMBER
        ADD R3,R3,#0
        BRnz #1
        ST  R0,StoreR0_0    ;if interrupt is alphabet ,store the number temporarily

LOOP_2  ST  R0,StoreR0_1    ;output the ENTER
        LEA R0,Str_enter
        TRAP X22
        LD  R0,StoreR0_1

        ADD R2,R2,#10       ;output 40 times

LOOP_3  JSR DELAY
        TRAP X21

        JSR DELAY
        TRAP X21

        JSR DELAY
        TRAP X21

        JSR DELAY
        TRAP X21

        ADD R2,R2,#-1
        BRz LOOP_1
        BRnzp LOOP_3

;delay function
DELAY   ST R1, DELAY_R1
        LD R1, DELAY_COUNT
DELAY_LOOP ADD R1, R1, #-1
        BRnp DELAY_LOOP
        LD R1, DELAY_R1
        RET

JUDGE_NUMBER AND R3,R3,#0       ;if R0 is number ,then R3 is 1,else is 0
             ST  R0,StoreR0_3
             LD  R4,ZERO
             ADD R4,R4,R0
             BRn #4
             LD  R4,NINE
             ADD R4,R4,R0
             BRp #1
             ADD R3,R3,#1
             LD  R0,StoreR0_3
             RET

DELAY_COUNT .FILL #2000
DELAY_R1  .BLKW #1
StoreR0_0 .BLKW #1
StoreR0_1 .BLKW #1
StoreR0_3 .BLKW #1
Ini_R0  .FILL x0037
ZERO    .FILL xFFD0
NINE    .FILL xFFC7
Str_enter .STRINGZ "\n"

        .END
```

Q&A

Q: rti之后，r0会变成什么？

A：会变成要输出的那个值。