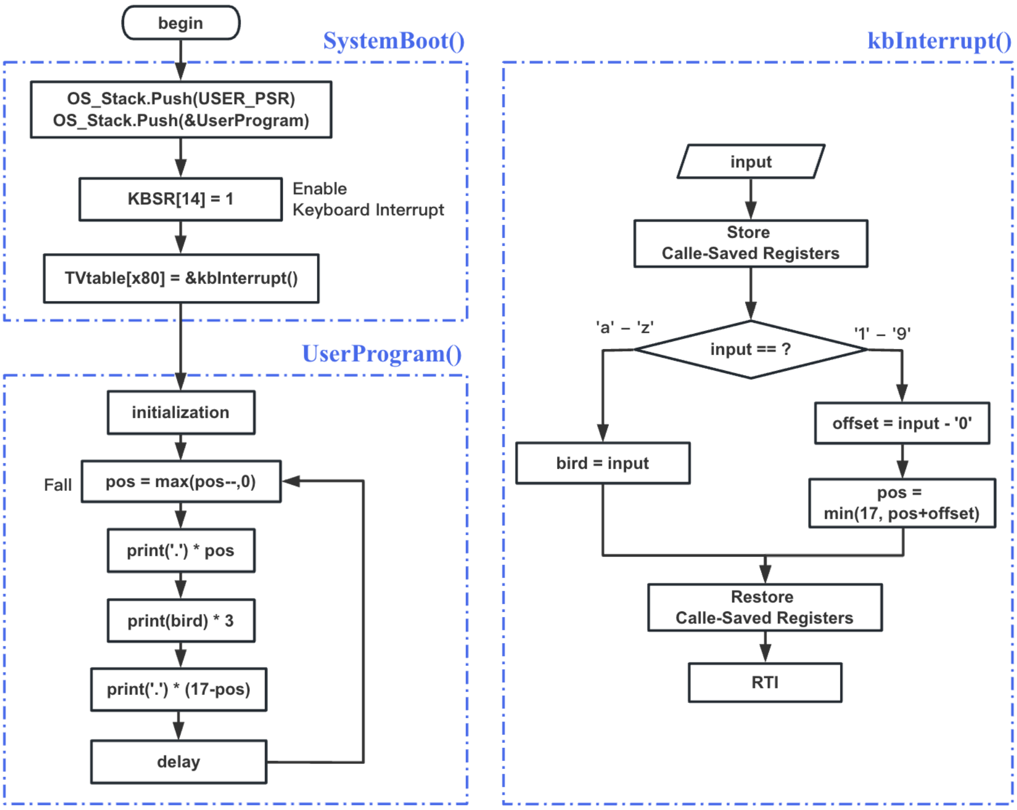
Report for LAB-4

1 Algorithm (flowchart)



2 Code (essential parts with comments)

```
.ORIG x0200; System Booting
; default System Booting code
LD  R6, OS_SP
LD  R0, USER_PSR
ADD R6, R6, #-1
STR R0, R6, #0
LD  R0, USER_PC
ADD R6, R6, #-1
STR R0, R6, #0
; Enable Keyboard Interrupt
LD  R0, KBSR_VAL
STI R0, KBSR;     KBSR[14]=1
; Add entry in IV table[x80]
LD  R0, ADDR_KB
STI R0, INTV_KB
; Jump to user program
RTI
;
OS_SP    .FILL x3000
USER_PSR .FILL x8002
USER_PC  .FILL x3000
KBSR     .FILL xFE00
KBSR_VAL .FILL x4000
ADDR_KB  .FILL x0450
INTV_KB  .FILL x0180
.END
```

```
.ORIG x3000; User Program(main)
        ; initialization
        LD  R1, ASC_A; bird = `a`
        AND  R2, R2, #0
        ADD  R2, R2, #6; pos = 5
        ; pos = max(pos-1, 0)
MLOOP   ADD  R2, R2, #-1; fall
        BRzp PRE
        AND  R2, R2, #0
PRE     ADD  R3, R2, #0; dots before
        JSR DOTS;        pos * dots
        ADD  R0, R1, #0; print bird
        PUTC
        PUTC
        PUTC
        LD  R0, EIGHTT; dots after
        NOT R3, R2
        ADD R3, R3, R0
        JSR DOTS;     (17-pos) * dots
        LD  R0, ASC_LF; new line
        PUTC
        JSR DELAY;      delay
        BR  MLOOP;     infinite loop
        .END
```

```
.ORIG x0450; Keyboard Interrupt
         ; Save Registers
INTR_KB  ST  R0, INTR_S1
         ST  R3, INTR_S2
         ST  R6, INTR_S3
         ; get input
         LDI R0, KBDR;
         LD  R3, NEG_A
         ADD R3, R3, R0
         BRn FLY;        input nums
         ; input = a~z, modify R1
         ADD R1, R0, #0
         BR  RESTORE
         ; input = 1~9, modify R2
FLY       LD  R3, NEG_ZERO
         ADD R3, R0, R3;  R3 = offset
         ADD R2, R2, R3
         ; if pos > 17, pos = 17
         LD  R3, NEG_SEVT
         ADD R3, R3, R2
         BRnz RESTORE
         LD  R2, SEVT
         ; Restore Registers
RESTORE  LD  R0, INTR_S1
         LD  R3, INTR_S2
         LD  R6, INTR_S3
         RTI

; Callee-Saved
INTR_S1   .BLKW #1
INTR_S2   .BLKW #2
INTR_S3   .BLKW #3
; Device Register
KBDR      .FILL xFE02
; ASCII Codes
NEG_A      .FILL #-97
NEG_ZERO  .FILL #-47
NEG_SEVT  .FILL #-18
SEVT       .FILL #18
           .END
```

3 Q & A

1. **What is the priority of keyboard interrupt request?**

   The priority level is stored at PSR[10:8]. If we HALT the machine when entering the interrupt service routine, we could find that PSR = x0401, which means the priority of keyboard interrupt request is 4.

2. **What happened from you strike keyboard till the interrupt service routine has finished?**

   Let’s assume that KBSR[15] = 1 when we strike the keyboard and the keyboard interrupt is enabled:

   1. Save the PSR of interrupted process in TEMP.
   2. Set privilege mode to Supervisor Mode (PSR[15] = 0).
   3. Set priority level to PL4 (PSR[10:8]=100).
   4. (If the interrupted process is in User Mode) Save R6 in Saved_USP, then load R6 with Saved_SSP.
   5. Push PSR & PC of interrupted process into Supervisor Stack.
   6. The keyboard supplies its interrupt vector (x80).
   7. The processor expands x80 to x0180.
   8. Load PC with Memory[x0180] (the starting address of keyboard interrupt service routine).
   9. Execute keyboard interrupt service routine till the RTI instruction.
   10. The RTI instruction pop PC & PSR from Supervisor Stack, then the process continues from where the interrupted program left off.

3. **How to deal with unexpected user input?**

   Now the program only judge whether the input’s ASCII code is less than ‘a’. If we want to distinguish those unexpected, we should check all the 4 boundaries:

   Take numbers for example, if input - ‘9’ <= 0, then we should ensure that input - ‘1’ >= 0 is also satisfied, or we jump to the RESTORE label directly.