lab4 report

algorithm

the booting program (at x0200)

- Enable the keyboard interrupt by saving `x4000` (the 14th bit is 1) to the address of KBSR(`xFE00`).
- The interrupt service routine is at address `x2000`, so we should save it to the address `x0180`, so that PC can load `x2000` when an interrupt is initiated.
- Push the PSR and PC onto the supervisor stack.

the code:

```
.ORIG x0200

    LD R0, KBINT_ADD
    STI R0, KBINT_entry ; The PC is loaded with the contents of memory location x0180
    LD R0, KBSR_D
    STI R0, KBSR_addr ; enable the interrupt

    ; PUSH USER_PSR and USER_PC
    LD R6, OS_SP
    ADD R6, R6, #-2
    LD R0, USER_PSR
    STR R0, R6, #1
    LD R0, USER_PC
    STR R0, R6, #0
    RTI

    OS_SP .FILL x3000
    USER_PSR .FILL x8002
    USER_PC .FILL x3000
    KBSR_addr .FILL xFE00
    KBSR_D .FILL x4000
    KBINT_entry .FILL x0180
    KBINT_ADD .FILL x2000
.END
```

the interrupt service routine(at x2000)

- Save the data of R0, R2 and R3, and load the data of R1(the address of the string to be output)
- Load the input character from `KBDR`.
- Check the ASCII code of the character. If it is a digit, branch to the `MOVE` subroutine, if it is a letter, branch to the `TRANS` subroutine, if neither, do nothing.
- In `MOVE` subroutine, calculate the new offset, if it is greater than `17`, the largest offset, change it to 17, then fill the last position of the bird with `.` and fill the current position of the bird with the current character.
- In `TRANS` subroutine, just change the character representing the bird.

- Outpus the new string, restore R0, R2 and R3 and `RTI`.

```
.ORIG   x2000
        ST R0, saveR0
        LD R1, saveR1
        ST R2, saveR2
        ST R3, saveR3
        LDI R0, KBDR
ISNUM   ; check if R0 >= '0' and R0 <= '9'
        ; detail ommitted


ISCHAR  ; check if R0 >= 'a' and R0 <= 'z'
        ; detail ommitted


MOVE    LD R2, ASCII_ZERO
        NOT R2, R2
        ADD R2, R2, #1
        ADD R2, R2, R0 ; offset
        ADD R2, R2, R4
        LD  R0, N_SEVENTEEN
        ADD R0, R2, R0
        BRnz SKIP ; if new offset > 17, change it to 17
        AND R2, R2, #0
        ADD R2, R2, #9
        ADD R2, R2, #8

SKIP    ADD R2, R2, #0
        LD  R3, ASCII_DOT
        ADD R0, R1, R4 ; previous ofs
        ADD R4, R2, #0 ; update R4
        LDR R5, R0, #0 ; previous char
        STR R3, R0, #0
        STR R3, R0, #1
        STR R3, R0, #2
        ADD R3, R5, #0 ; R3 = R0
        ADD R0, R1, R2
        STR R3, R0, #0
        STR R3, R0, #1
        STR R3, R0, #2
        BRnzp END

TRANS
        ; LEA R1, OUTPUT
        ; ADD R2, R1, R2
        ADD R2, R1, R4
        STR R0, R2, #0
        STR R0, R2, #1
        STR R0, R2, #2
        BRnzp END

END
        ADD R0, R1, #0
        PUTS
```

```
        AND R0, R0, #0
        ADD R0, R0, #10
        OUT
        LD R0, saveR0
        ; LD R1, saveOUT
        LD R2, saveR2
        LD R3, saveR3
        RTI

KBDR        .FILL xFE02
DSR         .FILL xFE04
DDR         .FILL xFE06
ASCII_ZERO  .FILL x0030
ASCII_NINE  .FILL x0039
ASCII_a     .FILL x0061
ASCII_z     .FILL x007A
ASCII_DOT   .FILL x002E
N_SEVENTEEN .FILL xFFEF
saveR0      .FILL x0000
saveR1      .FILL x3022 ; to be done
saveR2      .FILL x0000
saveR3      .FILL x0000
saveOUT     .FILL x0000
OUTPUT_adr  .BLKW #1
            .END
```

the main routine (at x3000)

- Loop infinitely to keep outputting. For each loop, decrease the offset by 1 if it is greater than `0`. To slow down the outputting, create an inner delay loop, which loops for `x8000` times.
- The user of some registers:
  - R1 : the address of the string to be output
  - R4: the current offset

```
.ORIG x3000
        ; R4 : ofs R1 : sym
        LEA R1, OUTPUT
        LD R3, FOURTY
        ; LD R2, SEVEN
        AND R4, R4, #0
        ADD R4, R4, #9
        ADD R2, R4, #0

LOOP    LD R1, OUT_ADDR
        ADD R0, R1, #0
        PUTS
        AND R0, R0, #0
        ADD R0, R0, #10
        OUT

DELAY   ADD R3, R3, #-1 ; loop for x8000 times
        BRp DELAY
```

```
        ; if R4 > 0, decrese it by 1
        ; modify the string
SKIP_   LD R3, FOURTY
        BRnzp LOOP


LABEL_       .FILL x0000
LABEL__       .FILL x0000
FOURTY      .FILL x8000
DELAY_COUNT .FILL #256
SAVER0_     .FILL x0000
SAVER2_     .FILL x0000
SAVER3_     .FILL x0000
DOT         .FILL x002E
OUT_ADDR    .FILL x3022 ; to be done
OUTPUT      .STRINGZ "........aaa........"
            .END
```

problems

- If the `RTI` instruction is executed in user mode after a keyboard interrupt, what will happen? How does the priviledge changes?
    - The privilege mode exception is caused.
    - Push the PSR and address of RTI onto the supervisor stack, load PC from x1000
    - Supervisor mode --> User mode --> Supervisor mode.
- How does LC-3 create a snapshot when an interrupt is enabled?
    - push PC, PSR to the supervisor stack.
    - change the stack pointer if the mode changes.
    - let the interrupt service save R0-R7
- What are the 3 kinds of exception? What condition can cause them?
    - Pivileged mode exception. It's caused when the program tries to execute the RTI instruction while in User mode.
    - Illegal opcode. It's caused when the program tries to execute an instruction of opcode 13.
    - Access control violation (ACV) exception. It's caused when the program tries to access a privileged memory location(x0000 ~ x2FFF) while in User mode.
- How to input a character through the keyboard by interrupt?
    1. INT is asserted, INTV is loaded with the interrupt vector corresponding to that external event.
    2. Put itself into Supervisor mode if it isn't in Supervisor mode, push the PSR and PC of the interrupted process onto the supervisor stack, and load the PC with the starting address of the interrupt service routine.
    3. Save the address of interrupt service routine to `x0180`, at the address, load the character from KBDR.