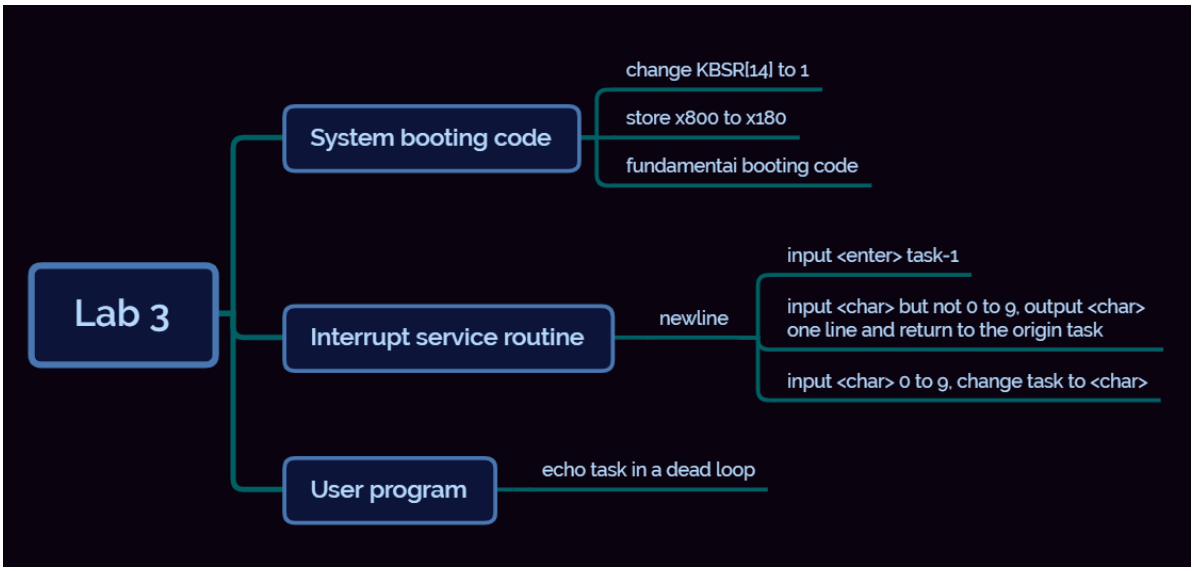
LAB 3 report



To achieve interrupt I/O, we need to change KBSR[14] (IE) to 1 and write the interrupt service routine for the keyboard(x180).

section 1 System booting code

three propose:

1. change KBSR[14] (IE) to 1
2. store x800 to x180
3. fundamentai booting code

```
.ORIG   x0200
ST      R2, SSP_R2
ST      R1, SSP_R1;save R1, R2
;
;store x800 to x180
LD      R2, HEX_180
LD      R1, HEX_800
STR     R1, R2, #0
HEX_180 .Fill x0180
HEX_800 .Fill x0800
;
;change KBSR[14] to 1
LDI     R1, KBSR
LD      R2, ADDITION ;x4000
ADD     R1, R1, R2
STI     R1, KBSR
;
;fundamentai 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
LD      R2, SSP_R2
LD      R1, SSP_R1
RTI
KBSR        .Fill xFE00
ADDITION    .Fill x4000
SSP_R1  .BLKW #1
SSP_R2  .BLKW #1
OS_SP   .FILL x3000
USER_PSR .FILL x8002
USER_PC .FILL x3000
.END
```

section 2 Interrupt service routine

firstly, if the cursor is not at the head of a line, create a new line.

“if the cursor is not at the head of a line” is important. Because we need to prevent two sequential newlines which will create an empty line.

according to the input, there are three functions:

1. input , task-1
2. input but not 0 to 9, output one line and return to the origin task
3. input 0 to 9, change task to

```
.ORIG   x0800
;initialization
ST      R0, KB_R0
ST      R1, KB_R1
ST      R2, KB_R2
ST      R7, KB_R7
;
;check if need create a newline and divide into the three functions
;"input <char> 0 to 9" doesn't need an extra section,just load KBDR to R0 is OK
ADD     R4, R4, #0; R4 is the counter to check if a newline is needed
BRz     SKIP      ; if R4 is 0, the cursor is at the head of a line
LD      R0, PENTER
OUT
AND     R4, R4, #0
SKIP
LDI     R0, KBDR  ; change R0 to what we input
LD      R1, MENTER
ADD     R2, R1, R0
BRz     ENTER     ; if input is enter
LD      R1, ASCII
ADD     R2, R1, R0
BRn     ELSE      ; if input < '0'
LD      R1, ASCII2
ADD     R2, R1, R0
BRp     ELSE      ; if input > '9'
BRnzp   EXIT
;
;ELSE section: input <char> but not 0 to 9
ELSE    LD  R2, TIME; TIME is #40, R2 is a counter
```

```
ELSE_PUT    JSR DELAY2
            OUT
            ADD R2, R2, #-1
            BRnp    ELSE_PUT ;output KBDR 40 times
            LD  R0, PENTER
            OUT
            LD  R0, KB_R0 ;back to orginal TASK
            BRnzp EXIT
;
;ENTER section:  task-1
ENTER   LD  R0, KB_R0
        LD  R1, ASCII
        ADD R2, R1, R0
        BRz EXIT
        ADD R0, R0, #-1
        BRnzp EXIT
;
;EXIT
EXIT    LD  R1, KB_R1
        LD  R2, KB_R2
        LD  R7, KB_R7
        RTI
;
;subroutine DELAY
DELAY2      ST R1, DELAY2_R1
            LD R1, DELAY2_COUNT
DELAY2_LOOP ADD R1, R1, #-1
            BRnp DELAY2_LOOP
            LD R1, DELAY2_R1
            RET
DELAY2_COUNT    .FILL #2048
DELAY2_R1       .BLKW #1
KBDR    .Fill xFE02
MENTER  .Fill #-10
PENTER  .Fill #10
ASCII   .Fill #-48
ASCII2  .Fill #-57
TIME    .Fill #40
KB_R0   .BLKW #1
KB_R1   .BLKW #1
KB_R2   .BLKW #1
KB_R7   .BLKW #1
.END
```

section 3 User program

echo the task in a dead loop

```
.ORIG   X3000
;initialization
PRINT   LD  R0, HEX_7; R0 is initialized to '7'
        AND R4, R4, #0
        LD  R3, M40
LOOP    JSR DELAY
        OUT
```

```
        ADD R4, R4, #1
        ADD R2, R3, R4
        BRnp    LOOP
        AND R4, R4, #0 ;R4 = 40, put a newline
        ST  R0, SAVE_R0
        LD  R0, NEWLINE
        OUT
        LD  R0, SAVE_R0
        BRnzp   LOOP
HEX_7   .Fill x37
M40     .Fill #-40
NEWLINE .Fill #10
SAVE_R0 .BLKW #1

DELAY       ST R1, DELAY_R1
            LD R1, DELAY_COUNT
DELAY_LOOP  ADD R1, R1, #-1
            BRnp DELAY_LOOP
            LD R1, DELAY_R1
            RET
DELAY_COUNT .FILL #2048
DELAY_R1    .BLKW #1
.END
```