Report 4

Algorithm Explanation

This program contains three parts start at different places: x0200 , x2000 , x3000

- x0200: This part contains the start part of the os.
- x2000: This part contains the interrupt part.
  - store R1,R2 to the place pointed by R6.
  - if the char in KBDR is a digit, calculate the position after changed and store it.
  - else change the display char to this input thing.
- x3000
  - init the program: load a to the R0, let R1 be 5.
  - loop for display.
  - first, display the dot before the char, with the cnt in R2.
  - second, display the char for 3 times, calculate the number that how many times should the dot be display after the char.
  - third,display the dot after char and output '\r'.
  - wait some time by using loop x1145 times.

Essential parts

judge whether the char is 0-9

```
LD R1,ASCLL_N0
ADD R1,R1,R0
BRn goon
LD R1,ASCLL_N9
ADD R1,R1,R0
BRnz digit
```

calculate the number of dots after the char

```
NOT R2,R1
ADD R2,R2,#9
ADD R2,R2,#9
```

wait some time

```
circle LD R2,loop_time
wait ADD R2,R2,#-1
BRp wait

loop_time .FILL x1145;
```

Q&A

`1.` how your interrupt program running

Answer: store R0, R1 to memory, the judge the input char.

If it is number, calculate the position where the char should be put in, else put the char into R0. Finally, return the 2 things to R0, R1.

`2.` how your program output the string

Answer: Firstly, load loop times to R2, output dots for R2 times. Then output the char 3 times,calculate the rest dots number, output them. Last output the '\r' for next loop.

Codes

```
.ORIG x0200
    ;;fill in x0180
    LD R0,KBINT_addr
    STI R0,KBINT_v
    ;enable KBSR[14]
    LD R0,KBSR_num
    STI R0,KBSR_addr
    ; origin os x200
    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_num .FILL x4000
    KBINT_v .FILL x0180
    KBINT_addr .FILL x2000
.END

.ORIG x2000
    ADD R6,R6,#-2
    STR R0,R6,#0
    STR R1,R6,#1
    LDI R0,KBDR
    ;judge if it is imlegal
    LD R1,ASCLL_N0
    ADD R1,R1,R0
    BRn goon
    LD R1,ASCLL_N9
    ADD R1,R1,R0
    BRnz digit
    goon STI R0,ch
    BR EXIT

    digit ADD R1,R1,#9
    LDI R0,position
    ADD R0,R0,R1
    ADD R1,R0,#-16
    BRnz stdigit
    LD R0,WIDTH
    stdigit
    STI R0,position

    EXIT
    LDR R0,R6,#0
    LDR R1,R6,#1
    ADD R6,R6,#2
    RTI

    KBDR .FILL xFE02
    ASCLL_N9 .FILL x-39
    ASCLL_N0 .FILL x-30
    ch .FILL x4000
    position .FILL x4001
    WIDTH .FILL #17
.END

.ORIG x3000
    ;inits
    LD R0,ASCLL_a
    STI R0,ASCLL_ch
    LD R1,place
    STI R1,cnt

    ;loop start
    loop_inf LDI R1,cnt
    ADD R2,R1,#0
    LD R0,ASCLL_dot

    ;loop output dot
    output_dot ADD R2,R2,#-1
    BRn output_ch
    OUT
    BR output_dot

    ;output 3 times
    output_ch LDI R0,ASCLL_ch
    OUT
    OUT
    OUT
    NOT R2,R1
    ADD R2,R2,#9
    ADD R2,R2,#9
    LD R0,ASCLL_dot

    ;loop output dot
    output_dot2 ADD R2,R2,#-1
    BRn line_feed
    OUT
    BR output_dot2

    ;output '\r'
    line_feed LD R0,ASCLL_enter
    OUT
    LDI R1,cnt
    BRz circle
    ADD R1,R1,#-1
    STI R1,cnt

    ;wait some time
    circle LD R2,loop_time
    wait ADD R2,R2,#-1
    BRp wait
    BR loop_inf

    cnt .FILL x4001;
    place .FILL x0005;
    loop_time .FILL x1145;
    ASCLL_ch .FILL x4000;x4000 has the ASCLL of char
    ASCLL_a .FILL x0061;a
    ASCLL_dot .FILL x002E;dot ASCLL
    ASCLL_enter .FILL x000A;'\r' ASCLL

.END
```