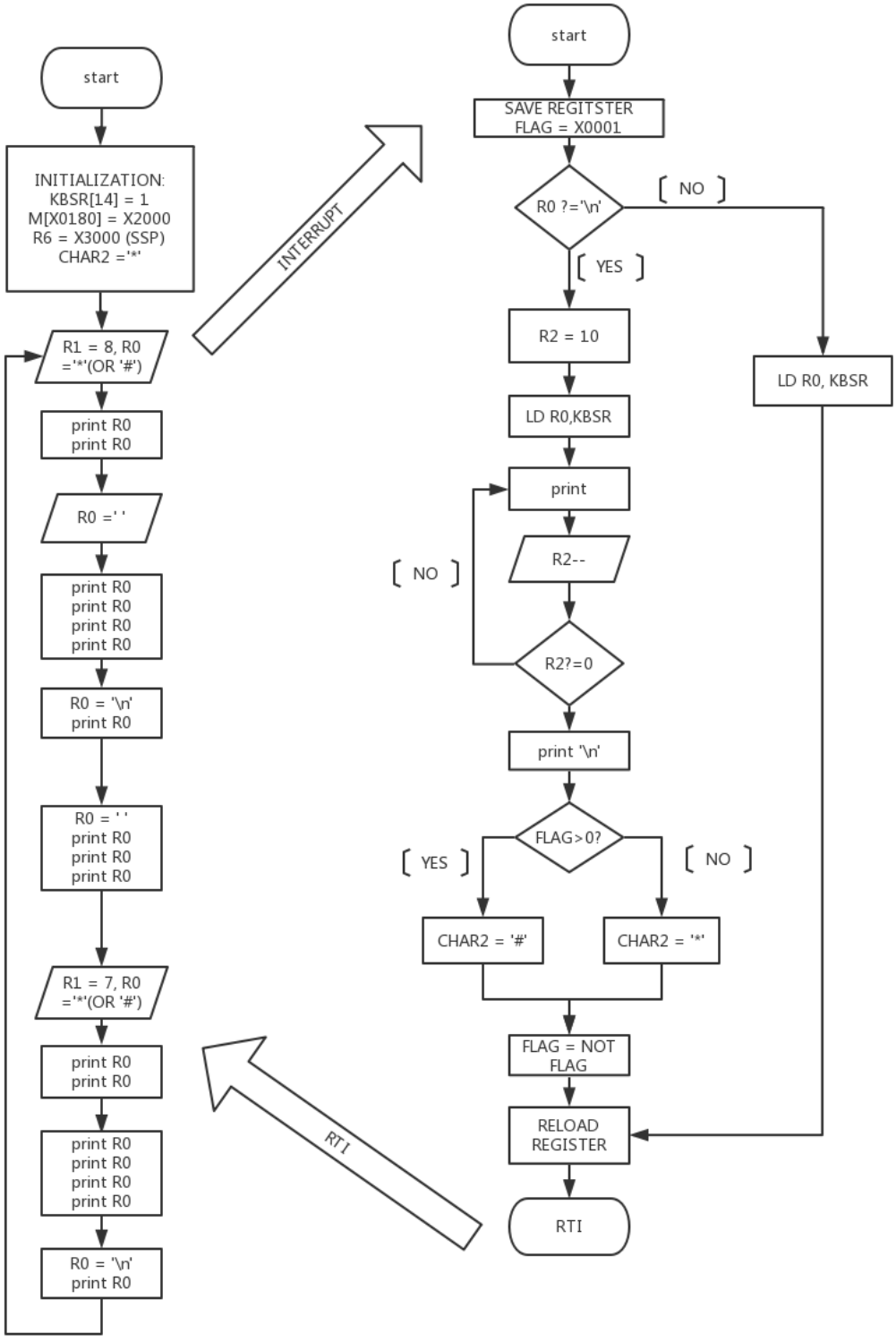
The 3rd lab report

(Due on Jul.17th)

1 .Program algorithm:



2.Brief Explanations

1)BEFORE THE EXECUTION

Just as the document puts it: "The purpose of this assignment is to show how interrupt-driven Input / Output can interrupt a running program, execute the interrupt service routine and return to the interrupted program."

Before the execution, there are three things we should do. Firstly, we should load the address x0180, which is an element of the interrupt vector table, with the value x2000, which is the entry address of interrupt service routine; secondly, we should put the stack pointer R6 at address x3000 to create a Supervisor Stack; What's more, we should also set KBSR [14] (IE) to 1 and initialize KBSR [15] by LDI KBSR with x4000 to enable interruption.

Besides, R3 will be set to the address of CHAR2, '*'(or "#").

To let the program execute more slowly, lots of "JSR delay" are added into the code.

2) THE EXECUTION OF USER PROGRAM

The function of user program at first is to print "* * " (4 blanks) for 8 times in a line and firstly print " " (3 blanks) then print "* * " (4 blanks) for 7 times in a new line. Since there is an infinite circulation, the two lines will be printed alternately.

3)THE EXECUTION OF INTERRUPT SERVICE ROUTINE

Before we start interrupting, the value of some registers must be stored. After that, when we press a key (with ASCII), the user program will be interrupted and the interrupt service routine will be executed from x2000. Since the interruption can occur at any time in any position, we should firstly check if the interruption occurs just before the end of a line. If so, it is a legal interruption, otherwise it's illegal and the program will directly jump to the end of the routine, fetch the character in KBDR (if we didn't do so, KBSR [15] would stay 1 and we couldn't input more characters), reload the register and RTI.

If the interruption is legal, the routine will firstly print the input character 10 times in a line (realized by LD R0, KBDR and persistently polling DSR in order to output the input characters on the screen of monitor (DDR)). Then examine the flag, whose initial value is x0001. If it is positive, change the value of CHAR2 to '#' so that the output after the interrupt will change (realized by STR R0, R3, #0, R0 stores the ASCII value of '#' or '*'). Then NOT flag. If flag is negative, change the value of CHAR2 to '*'.

At the end of routine, reload the register and RTI.

3.Source Code (in appendix)