LAB 6: FACTORIAL

INTRODUCTION

The purpose of this assignment is to show how interrupt-driven Input/Output can interrupt a program that is running, execute the interrupt service routine, and return to the interrupted program, picking up exactly where it left off (just as if nothing had happened). In this assignment, we will use the Keyboard as the input device for interrupting the running program.

In mathematics, the factorial of a non-negative integer n, denoted by $n!$, is the product of all positive integers less than or equal to n.

$$n! = \begin{cases} \quad 1 \quad , & n = 0 \\ n \cdot (n-1)!, & n > 0 \end{cases}$$

After receiving a keyboard interrupt, you need to echo the input character and determine whether it is a decimal number. If the input is indeed a decimal number, you should store its value in memory and pass it as the variable N to the Factorial subroutine for calculating the result.

DETAILS

The assignment consists of three parts, **but you only need to do the first two parts**. You can finish this assignment by completing the attachment provided in part C.

A. The user program

The argument of factorial, recorded as N , will be initialized with `xFFFF` and stored at `X3FFF` in memory.

Your user program, which starts at `x3000` , will continually (i.e. in an infinite loop) print your student id like: `PB12345678 PB12345678 PB12345678 PB12345678 PB12345678 PB12345678 ······` and wait for argument N to become a valid value. When N becomes a valid value, the program stops waiting and calls the Factorial subroutine to solve the problem. Once Factorial subroutine has solved the problem, your result should be displayed on the console in decimal format. It is important to call a `HALT` instruction after displaying the result in order to properly terminate the program.

Due to the limited range of signed numbers that can be represented in LC3, when the user enters '8' or '9' as input, our program should simplify by displaying an error message. Specifically, the program should output `<the input`

`character>! is too large for LC-3.` before calling `HALT`.

To ensure the output on the screen is not too fast to be seen by the naked eye, the user program should include a piece of code that will count down from `2500` (or any other numbers) after each word is output on the screen. A simple way to do this is with the following subroutine DELAY:

```
         ; code of delay
DELAY   ST R1, SaveR1
        LD R1, COUNT
REP     ADD R1, R1, #-1
        BRp REP
        LD R1, SaveR1
        RET
```

B. The keyboard interrupt service routine

The keyboard interrupt service routine, starting at address `x1000`, performs the following tasks:

1. It examines the key typed to determine if it is a decimal digit.
   - If the input character **is not** a decimal digit, the service routine prints the following message : `<the input character> is not a decimal digit.`
   - If the input character **is** a decimal digit, the service routine prints the following message `<the input character> is a decimal digit.` Additionally, it saves this decimal number to memory address `x3FFF`.
2. After printing the appropriate message and saving the decimal number (if applicable), the service routine outputs a line feed character (`x0A`) to the screen.
3. Finally, the keyboard interrupt service routine terminates using the RTI instruction.

By following these steps, the service routine can effectively handle keyboard interrupts and provide the desired functionality.

For example, if the input key is 'K', the interrupt service routine will print: `K is not a decimal digit.`
If the input key is '4', the interrupt service routine will print: `4 is a decimal digit.` and change the value of `x3FFF` to 4.

C. The operating system enabling code

Unfortunately, we have not installed any operating system on the LC-3, so we provide you with STARTER CODE (in attachment) that enables interrupts. **You MUST use the starter code for this assignment.** The locations to write the user program and interrupt service routine are marked with comments.

The starter code does the following:

1. Initializes the interrupt vector table with the starting address of the interrupt service routine. The keyboard interrupt vector is `x80` and the interrupt vector table begins at memory location `x0100`. The keyboard interrupt service routine begins at `x1000`. Therefore, we must initialize memory location `x0180` with the value `x1000`.
2. Sets bit 14 of the KBSR to enable interrupts.
3. Pushes a PSR and PC to the system stack so that it can jump to the user program at x3000 using an RTI instruction.

EXAMPLE

example 1

```
 PB12345678 PB12345678 PB12345678 PB12345678 PB12345678 PB12345678 PB12345678 PB12345678
h is not a decimal digit.        //Input character 'h'
PB12345678 PB12345678 PB12345678 PB12345678 PB12345678 PB12345678
5 is a decimal digit.            //Input character '5'
5! = 120.




--- Halting the LC-3 ---
```

example 2

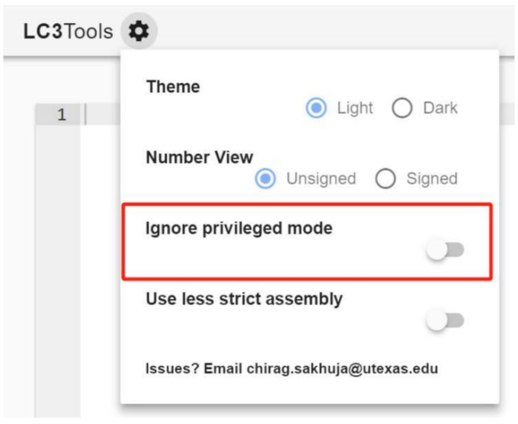
```
 PB12345678 PB12345678 PB12345678 PB12345678 PB12345678 PB12345678 PB12345678 PB12345678
9 is a decimal digit.            //Input character '9'
9! is too large for LC-3.




--- Halting the LC-3 ---
```

NOTES AND SUGGESTIONS

1. Since the interrupt can be triggered at any point, the output of the interrupt service routine may show up anywhere.
2. Since your user program contains an infinite loop, you will have to press the "Pause" button in the simulator if you wish to stop the program.
3. Unlike previous labs, the PC will be initialized to `x800` for this assignment because the first code that is executed will be in the operating system.
4. **Don't forget to save and restore any registers that you use in the interrupt service routine.**
5. **If you see � on the console, then please read Note 4 again!**
6. Please make sure that the `Ignore privileged mode` switch is OFF.


LC3Tools

Theme
[✓ Light] Dark

Number View
[✓ Unsigned] Signed

Ignore privileged mode: OFF

Use less strict assembly: OFF

Issues? Email chirag.sakhuja@utexas.edu

ADDITIONAL REQUIREMENTS

If you don't comply with these requirements, the lab may be counted as an invalid work.

1. Your report should be structured into the following sections:
    - Purpose
    - Principles
    - Procedure (e.g. bugs or challenges you encountered and how to solve them)
    - Results
2. Your submission be structured as shown below.

```
PB********_Name.zip
├──── PB********_Name_report.pdf
└──── lab6.asm
```