REPORT

LAB05 Interrupt a Running Program

PB19010450 和泳毅

Requirements

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.

- Write the user program described below;
- Write the keyboard interrupt service routine described below.

Details:

The assignment consists of three parts, but you only need to do the first two parts:

A. The user program

Your user program, which starts at **x3000**, will continually (i.e. in an infinite loop) print the word “ICS2020” like:

$$ICS2020\ ICS2020\ ICS2020\ ICS2020\ ICS2020\ ICS2020\ \cdots\cdots$$

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:

```
DELAY   ST   R1,SaveR1
        LD  R1,COUNT
REP     ADD  R1,R1,#-1
        BRp REP
        LD  R1,SaveR1
        RET
COUNT   .FILL x7FFF
SaveR1  .BLKW #1
```

B. The keyboard interrupt service routine

The keyboard interrupt service routine, which starts at **x1000**, will examine the key typed to see if it it is a **decimal digit**.

If the character typed is **NOT** a decimal digitthe interrupt service routine will, starting on a new line on the screen, print " is not a decimal digit." For example, if the input key is '#', the interrupt service routine will print:

$$\#\ is\ not\ a\ decimal\ digit.$$

The service routine would then print a line feed (x0A) to the screen, and finally terminate with an RTI.

If the character typed **IS** a decimal digit, the interrupt service routine will, starting on a new line on the screen, print " is a decimal digit.". If the input key is ‘**4**’, the interrupt service routine will print:

$$4\ is\ a\ decimal\ digit.$$

The service routine would then print a line feed (x0A) to the screen, and finally terminate with an RTI.

Hint: Don't forget to save and restore any registers that you use in the interrupt service routine.

C. The operating system enabling code

Unfortunately, we have not installed Windows or Linux 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:

$$\begin{array}{l}
ICS2020\ ICS2020\ ICS2020\ ICS2020 \\
h\ is\ not\ a\ decimal\ digit. \qquad //Inputcharacter`h' \\
ICS2020\ ICS2020\ ICS2020\ ICS2020\ ICS2020 \\
4\ is\ a\ decimal\ digit. \qquad //Inputcharacter`4' \\
ICS2020\ ICS2020\ ICS2020\ ICS2020\ ICS2020\ ICS2020\ \ldots
\end{array}$$

Notes and Suggestions:

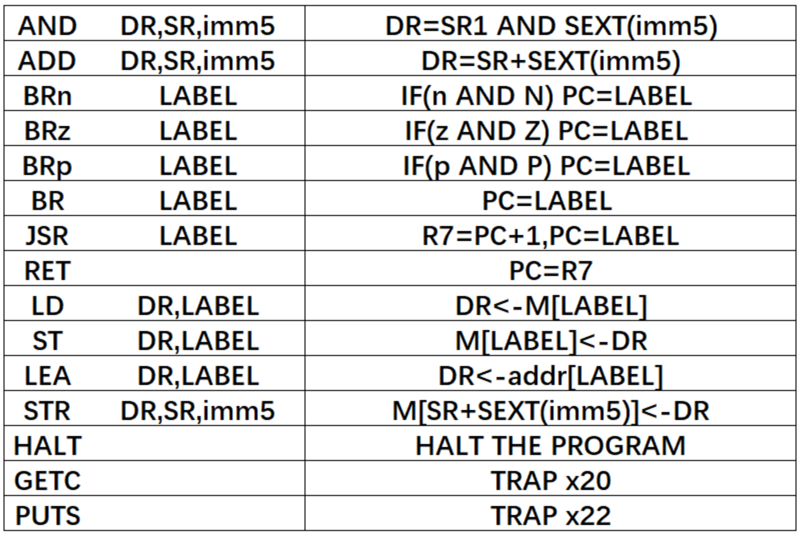
1. Since the interrupt can be triggered at any point, the output of the interrupt service routine may show up anywhere.
2. Since your user program contains an infinite loop, you will have to press the "**Pause**" button in the simulator if you wish to stop the program.
3. Unlike previous labs, the PC will be initialized to **x800** for this assignment because the first code that is executed will be in the operating system.
4. **Please make sure that the "Ignore privileged mode" switch is OFF. (Default configuration is OFF in LC-3 simulator)**

Design

1. We can use the **BR** instruction to implement the infinite loop function.
2. We can use the **ASCII** code of '0' and '9' to determine if the user's input is decimal digit.
3. We can prepare the string to be printed first, and then add the characters to that string after the user has entered them.
4. We can use the string "\n" to replace the a **newline** character (ASCII code **x000A**) .

Code Writing

1. Instructions to be used



AND	DR,SR,imm5	DR=SR1 AND SEXT(imm5)
ADD	DR,SR,imm5	DR=SR+SEXT(imm5)
BRn	LABEL	IF(n AND N) PC=LABEL
BRz	LABEL	IF(z AND Z) PC=LABEL
BRp	LABEL	IF(p AND P) PC=LABEL
BR	LABEL	PC=LABEL
JSR	LABEL	R7=PC+1,PC=LABEL
RET		PC=R7
LD	DR,LABEL	DR<-M[LABEL]
ST	DR,LABEL	M[LABEL]<-DR
LEA	DR,LABEL	DR<-addr[LABEL]
STR	DR,SR,imm5	M[SR+SEXT(imm5)]<-DR
HALT		HALT THE PROGRAM
GETC		TRAP x20
PUTS		TRAP x22

2. The infinite loop in Part B.

```
        ST  R0, SaveR0
LOOP    LEA R0, LC
        PUTS
        JSR DELAY
        BR  LOOP;Infinite loop
        LD  R0, SaveR0
        HALT
DELAY   ST  R1, SaveR1;Increase the running time
        LD  R1, COUNT;Count is 32767
REP     ADD R1, R1, #-1
        BRp REP
        LD  R1, SaveR1
        RET
```

3. Check the input in Part C.

```
        ST  R0, Save_R0
        ST  R1, Save_R1
        GETC;There is no return display here.
        ADD R1, R0,#0;Copy
        LD  R0, N48;>=0
        ADD R0, R1, R0
        BRn ISN
        LD  R0, N57;<=9
        ADD R0, R1, R0
        BRp ISN
;If 0<= input <=9
        LEA R0, SHOW_2
        STR R1, R0, #1
        LEA R0, SHOW_2
        BR  END
ISN     LEA R0, SHOW_1
        STR R1, R0, #1
        LEA R0, SHOW_1
END     PUTS
        RTI
        LD  R0, Save_R0
        LD  R1, Save_R1
```

4. Non-code section

```
;Part B
COUNT   .FILL x7FFF
SaveR0  .FILL x0000
SaveR2  .FILL x0000
SaveR1  .BLKW #1
LC      .STRINGZ "ICS2020 "
;
;Part C
Save_R0 .FILL   x0000
Save_R1 .FILL   x0000
CHAR    .FILL   x0000
N48     .FILL   xFFD0;-48
N57     .FILL   xFFC7;-57
NEWLINE .FILL   x000A
SHOW_1  .STRINGZ "\n  is not a decimal digit.\n"
SHOW_2  .STRINGZ "\n  is a decimal digit.\n"
```

Result Test

1. The example

```
ICS2020 ICS2020 ICS2020 ICS2020
h is not a decimal digit.
ICS2020 ICS2020 ICS2020 ICS2020 ICS2020
4 is a decimal digit.
ICS2020 ICS2020 ICS2020 ICS2020 ICS2020 ICS2020
```

2.

```
ICS2020 ICS2020 ICS2020 ICS2020 ICS2020
2 is a decimal digit.
ICS2020 ICS2020 ICS2020 ICS2020 ICS2020 ICS2020
4 is a decimal digit.
ICS2020 ICS2020 ICS2020 ICS2020 ICS2020
t is not a decimal digit.
ICS2020 ICS2020 ICS2020 ICS2020 ICS2020 ICS2020 ICS2020
# is not a decimal digit.
ICS2020 ICS2020 ICS2020 ICS2020
y is not a decimal digit.
ICS2020 ICS2020 ICS2020 ICS2020
0 is a decimal digit.
ICS2020 ICS2020 ICS2020 ICS2020 ICS2020
1 is a decimal digit.
ICS2020 ICS2020 ICS2020 □
```

Thinking

1. The computer runs very fast, and to slow it down to observe the intermediate processes, we can add some large and useless operations;
2. Using the RTI instruction, we can jump codes located at different locations in the memory address;

Appendix

Complete code:

LC-3：

```
        .ORIG x800
        ; (1) Initialize interrupt vector table.
        LD  R0, VEC
        LD  R1, ISR
        STR R1, R0, #0

        ; (2) Set bit 14 of KBSR.
        LDI R0, KBSR
        LD  R1, MASK
        NOT R1, R1
        AND R0, R0, R1
        NOT R1, R1
        ADD R0, R0, R1
        STI R0, KBSR

        ; (3) Set up system stack to enter user space.
        LD  R0, PSR
        ADD R6, R6, #-1
        STR R0, R6, #0
        LD  R0, PC
        ADD R6, R6, #-1
        STR R0, R6, #0
        ; Enter user space.
        RTI
;
VEC     .FILL x0180
ISR     .FILL x1000
KBSR    .FILL xFE00
MASK    .FILL x4000
PSR     .FILL x8002
PC      .FILL x3000
        .END

        .ORIG x3000
        ST  R0, SaveR0
        ST  R2, SaveR2
        AND R2, R2, #0
LOOP    LEA R0, LC
        PUTS
        JSR DELAY
        ADD R2, R2,#0
        BRz LOOP
        LD  R0, SaveR0
        LD  R2, SaveR2
        HALT
DELAY   ST  R1, SaveR1
```

```
        LD  R1, COUNT
REP     ADD R1, R1, #-1
        BRp REP
        LD  R1, SaveR1
        RET
;
COUNT   .FILL x7FFF
SaveR0  .FILL x0000
SaveR2  .FILL x0000
SaveR1  .BLKW #1
LC      .STRINGZ "ICS2020 "
        .END

        .ORIG x1000
        ST  R0, Save_R0
        ST  R1, Save_R1
        GETC
        ADD R1, R0,#0
        LD  R0, N48;>=0
        ADD R0, R1, R0
        BRn ISN
        LD  R0, N57;<=9
        ADD R0, R1, R0
        BRp ISN
;
        LEA R0, SHOW_2
        STR R1, R0, #1
        LEA R0, SHOW_2
        BR  END
ISN     LEA R0, SHOW_1
        STR R1, R0, #1
        LEA R0, SHOW_1
END     PUTS
        RTI
        LD  R0, Save_R0
        LD  R1, Save_R1
;
Save_R0 .FILL   x0000
Save_R1 .FILL   x0000
CHAR    .FILL   x0000
N48     .FILL   xFFD0;-48
N57     .FILL   xFFC7;-57
NEWLINE .FILL   x000A
SHOW_1  .STRINGZ "\n  is not a decimal digit.\n"
SHOW_2  .STRINGZ "\n  is a decimal digit.\n"
        .END
```