lab6

实验目的

1. 使用键盘作为中断运行程序的输入设备, 学习中断驱动的输入输出如何中断正在运行的程序, 执行中断服务例程, 并返回被中断的程序.
2. 通过直接读写输入输出的数据寄存器, 深入了解LC-3是如何处理输入输出的.

实验方法

PartA: 用户程序

重复输出学号

1. 学号字符串预先放入内存.
2. 调用`PUTS`来打印学号.
3. 每次打印之后, 将`x3FFF`处的数字加载出来, 如果其仍是`xFFFF`, 则继续循环打印; 否则说明得到了合法的`N`值, 跳出循环.
4. 两次打印之间调用`DELAY`来延迟一会.
   代码如下:

```
            LEA     R0, STU_ID                  ; print student id
INF_LOOP    PUTS
            JSR     DELAY                       ; delay before printing the next
            LDI     R1, ADDR_N                  ; R1 <- M[x3FFF]
            BRn     INF_LOOP                    ; if R1 < 0, continue loop
...
STU_ID      .STRINGZ "PB22000197 "
ADDR_N      .FILL   x3FFF
```

计算阶乘

由于本题有效的阶乘结果只有7种, 如果正常计算的话还需要将多位十进制数转化为ASCII码输出, 非常繁琐, 因此采用打表的方法.
首先将7种阶乘结果对应的字符串存入内存:

```
ONE         .STRINGZ "1."                       ; 0! or 1!
TWO         .STRINGZ "2."                       ; 2!
THREE       .STRINGZ "6."                       ; 3!
FOUR        .STRINGZ "24."                      ; 4!
FIVE        .STRINGZ "120."                     ; 5!
```

```
SIX         .STRINGZ "720."                 ; 6!
SEVEN       .STRINGZ "5040."                ; 7!
```

然后先将 `N` 减去8, 判断其是否为8或9, 是则输出错误消息:

```
                    ADD     R1, R1, #-8             ; R1 <- R1 - 8
                    BRzp    ERROR
...
ERROR       LEA     R0, ERROR_STR           ; print "! is too large for LC-3."
                    PUTS
                    HALT
...
ERROR_STR   .STRINGZ "! is too large for LC-3."
```

否则, 逐次给 `N` 加1, 若其为0, 则对应着$7, 6, \cdots, 1, 0$, 载入对应的阶乘字符串并跳转到结果输出模块:

```
            LEA     R0, SEVEN
            ADD     R1, R1, #1
            BRz     PRINT_RES               ; N == 7
            LEA     R0, SIX
            ADD     R1, R1, #1
            BRz     PRINT_RES               ; N == 6
            LEA     R0, FIVE
            ADD     R1, R1, #1
            BRz     PRINT_RES               ; N == 5
            LEA     R0, FOUR
            ADD     R1, R1, #1
            BRz     PRINT_RES               ; N == 4
            LEA     R0, THREE
            ADD     R1, R1, #1
            BRz     PRINT_RES               ; N == 3
            LEA     R0, TWO
            ADD     R1, R1, #1
            BRz     PRINT_RES               ; N == 2
            LEA     R0, ONE                 ; N == 1 or 0
...
PRINT_RES   PUTS                            ; print result
            HALT
```

PartB: 键盘中断服务程序

保存用户程序的寄存器

键盘中断发生在用户程序中的循环打印模块, 因此需要保存 `R0` 和 `R1`:

```
            ST      R0, SAVE_R0                 ; save R0
            ST      R1, SAVE_R1                 ; save R1
...
SAVE_R0     .BLKW       1
SAVE_R1     .BLKW       1
```

读取`KBDR`寄存器并回显

将寄存器的值载入`R0`, 调用`OUT`来回显.

```
                        LDI     R0, KBDR                    ; R0 <- M[xFE02], echo the
input
            OUT
...
KBDR        .FILL   xFE02
```

判断输入是否为数字

分别将其与`'0'`和`'9'`相减即可:

```
                        LD      R1, NINE                    ; R1 <- -'9'
            ADD     R1, R0, R1                  ; R1 <- R0 - '9'
            BRp     NOT_DECIMAL                 ; if R0 > '9', then it's not a decimal
            LD      R1, ZERO                    ; R1 <- -'0'
            ADD     R1, R0, R1                  ; R1 <- R0 - '0'
            BRn     NOT_DECIMAL
...
ZERO        .FILL   #-48                         ; -'0'
NINE        .FILL   #-57                         ; -'9'
```

存储合法的`N`

使用`STI`指令将`R1`(因为上一步做了`R1 <- R0 - '0'`)存储到`x3FFF`位置:

```
                        STI     R1, ADDR_N_                 ; M[x3FFF] <- R1 (R0 -
'0')
...
ADDR_N_     .FILL   x3FFF
```

输出相应字符串

字符串已预先放入内存:

```
                        LEA     R0, DEC                     ; print " is a decimal
digit."
            PUTS
            BR      RETURN

NOT_DECIMAL LEA     R0, NOT_DEC                 ; print " is not a decimal digit."
            PUTS
...
NOT_DEC     .STRINGZ " is not a decimal digit.\n"
DEC         .STRINGZ " is a decimal digit.\n"
```

返回

恢复`R0`和`R1`, 调用`RTI`返回:

```
RETURN      LD      R1, SAVE_R1                 ; restore R1
            LD      R0, SAVE_R0                 ; restore R0
                    RTI
```

遇到的困难与解决方案

1. 原先我写了通过循环来正常计算阶乘的代码, 还涉及到一个`MUL`函数. 写完以后运行的时候才意识到阶乘结果可能是多位数, 要想打印还需将其转化为ASCII码. 权衡之下, 觉得打表直接选择输出的字符串更为方便.
2. 一开始在中断程序中没有保存`R0`和`R1`, 导致中断结束以后程序不能正常运行, 改正以后就好了.

程序运行结果

链接：https://rec.ustc.edu.cn/share/e271dad0-a0c8-11ee-a0bc-9b01b39e6d20