> 第九周实验任务：采用键盘中断的方式，当输入是字符或者数字的时候，回显输入并回车换行；否则退出程序

第九周实验报告：键盘中断

1. 具体功能实现细节：

1.1 具体实现难点

我们如果想要利用键盘中断，也就是第9号中断，来实现这个程序，我们需要去自定义一个`int 9`中断例程，并且需要改写原来的`int 9`中断例程对应的中断向量表，将其修改为我们自定义的`int 9`中断例程的入口地址。

我们所做的事情是，如果键盘输入的是字符或者数字的话，我们不仅仅要退出，而且需要将其输出并回车换行，这个是很难的。

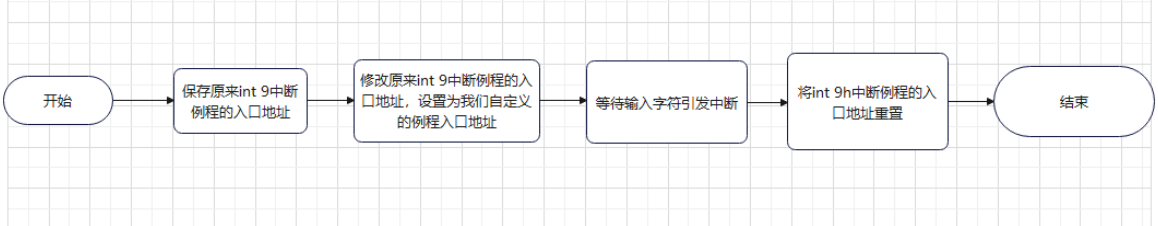
因为原来的9号中断例程所做的事情就是读取60h端口的扫描码，然后如果是字符键的扫描码的话，就将该扫描码和它对应的字符ASCII码送入内存中的BIOS缓冲区，然后接着我们可以调用`int 16`中断例程，(16号中断的0号功能所做的事情就是从键盘缓冲区中读取一个键盘输入，并且将其从缓冲区删除)，可以实现读取到我们键盘按下的键的字符。

但是我们现在已经将9号中断向量表中的入口地址修改成我们自定义的中断例程入口地址了，所以说键盘中断之后不会去执行原来的系统9号中断，这样的话，我们按下的键盘对应的字符就不会送入BIOS缓冲区里面，更别说利用`int 16`中断例程去从中读取了。

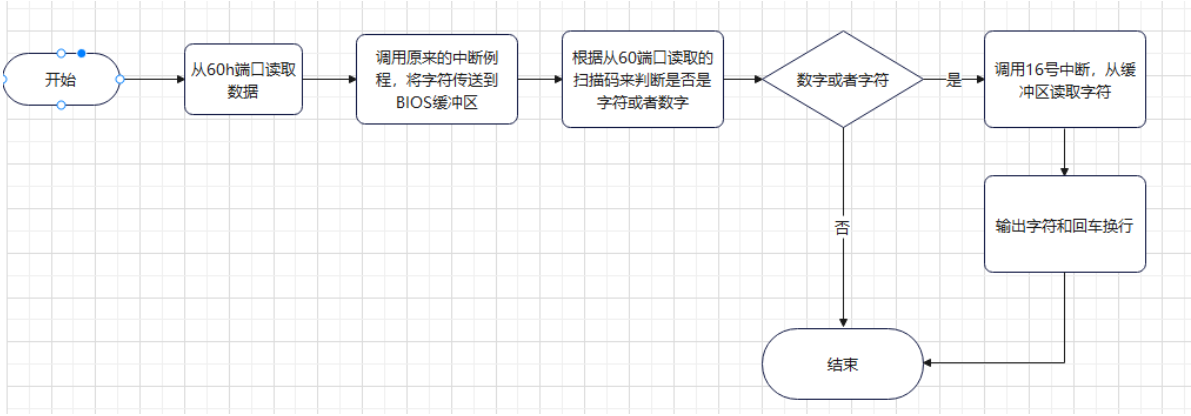
解决办法：我们可以提前保存好原来9号中断例程对应的地址，然后我们在我们自定义的中断例程中去调用这个中断例程，就可以实现了将按下的字符送入内存中的BIOS缓冲区，接下来就可以利用16号中断来读取字符，来进一步做我们想要做的事情。

1.2 流程图

主程序流程图：



自定义的9号中断例程：



附录1：输出结果

附录2：具体代码实现

```
# 采用键盘中断的方式，当输入是字符或者数字的时候，回显输入并且回车换行；否则退出

assume cs:code,ds:data,ss:stack

data segment
    dw 0,0
data ends

stack segment
    db 128 dup(0)
stack ends

code segment
start:
    mov     ax,stack
    mov     ss,ax
    mov     sp,128
    mov     ax,data
```

```
    mov     ds,ax

    mov     ax,0
    mov     es,ax

    # 保存原来int 9中断例程的入口地址
    push    es:[9*4]
    pop     ds:[0]
    push    es:[9*4+2]
    pop     ds:[2]

    # 修改原来int 9中断例程的入口地址，设置为我们自定义的函数地址
    mov     word ptr es:[9*4],offset do9
    mov     es:[9*4+2],cs

    mov     ah,01h
    int     21H

    # 将int 9中断例程的入口地址重置为原来的
    push    ds:[0]
    pop     es:[9*4]
    push    ds:[2]
    pop     es:[9*4+2]

    mov     ax,4c00h
    int     21H


do9:jmp     short do9start
    # db      "Input wrong!"
    # 如果输入的不是字符或者数字的话，就输出"Input wrong!"并且退出!
do9start:
    push    ax
    push    bx
    push    es

    in      al,60h

    pushf
    # 因为我们需要调用9号中断从60端口将数据读入缓冲区，但是我们现在
    # 已经对9号功能中断做了修改，所以说我们想要调用9号中断，我们必须
    # 使用其他的方法去实现int指令
    pushf
    pop     bx
    and     bh,11111100b
    push    bx
    popf

    # 调用原来的int 9中断例程，将60端口的字符写入BIOS缓冲区
    call    dword ptr ds:[0]

    # 此处进行比较
    cmp     al,02h
    # 如果小于2
    jb      do9_wrong
```

```
    cmp     al,0bh
    # 如果小于等于11
    jna     do9_right
    cmp     al,0fh
    # 如果小于等于0fh
    jna     do9_wrong
    cmp     al,19h
    # 如果小于等于19h
    jna     do9_right
    cmp     al,1dh
    jna     do9_wrong
    cmp     al,26h
    jna     do9_right
    cmp     al,2bh
    jna     do9_wrong
    cmp     al,32h
    jna     do9_right
    jmp     do9_wrong

# 如果输入其他字符就退出
do9_wrong:
    jmp     do9_ret

# 如果输入字符或者数字就回显输入并回车换行
do9_right:
# 回显
    mov     ah,0
    int     16h
    mov     dl,al
    mov     ah,02h
    int     21H

# 打印出一个回车换行
    mov     dl,0dh
    mov     ah,2
    int     21h
    mov     dl,0ah
    mov     ah,2
    int     21h
    jmp     do9_ret

do9_ret:
    pop     es
    pop     bx
    pop     ax
    iret
do9end:
    nop
code    ends
end     start
```