

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ
по лабораторной работе №5
по дисциплине «Организация ЭВМ и систем»
Тема: Написание собственного прерывания.

Студент гр. 1303

Самохин К.А.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2022

Цель работы.

Изучить особенности прерывания в языке Ассемблера. Написать собственное прерывание.

Задание.

Вариант 22.

Написать прерывание 16h – прерывание от клавиатуры (по заданному скан-коду клавиши выполнять действия, указанные ниже).

А - Выполнить вывод сообщения на экран заданное число раз, после чего вставить фиксированную задержку и вывести сообщение о завершении обработчика.

Выполнение работы.

В сегменте данных DATA расположены переменные:

- KEEP_CS – переменная для хранения сегмента того прерывания, которое мы заменяем.
- KEEP_IP – переменная для хранения смещения замененного нами прерывания.
- MESSAGE – сообщение, которое будет выводиться во время работы прерывания.
- END_MES – сообщение, которое будет выведено после завершения обработчика.

В сегменте кода в первую очередь определяется процедура WriteMsg, предназначенная для вывода сообщения. Далее определяется процедура пользовательского прерывания FUNC. В начале процедуры в стек сохраняются значения регистров до входа в прерывание. Далее при помощи цикла Ip производится вывод сообщения MESSAGE число раз, заданное в регистре CX.

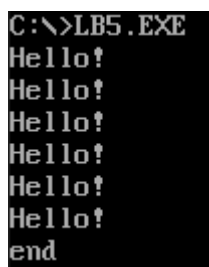
После вывода строк при помощи прерывания 15h реализована задержка. В конце процедуры выводится сообщение об окончании обработки.

Вызов прерывания происходит в процедуре main. В начале происходит сохранения адреса старого обработчика прерывания при помощи функции 35h прерывания 21h. После этого производится установка нового прерывания при помощи функции 25h прерывания 21h. Далее определена метка input_loop, в которой ожидается нажатие клавиши, вызывающей прерывание.

После выполнения прерывания происходит сброс флага прерывания (cli), восстанавливается старый вектор прерывания и происходит установка флага прерывания(sti)

Тестирование.

Работа программы с заданными условиями представлена на Рисунке 1.



```
C:\>LB5.EXE
Hello!
Hello!
Hello!
Hello!
Hello!
Hello!
end
```

Рисунок 1.

Выводы.

В ходе лабораторной работы изучены виды прерываний и работа с ними. В соответствии с заданием разработано собственное прерывание.

ПРИЛОЖЕНИЕ А

Название файла: lb5.asm

```
AStack  SEGMENT STACK
        DB 1024 DUP(?)
AStack  ENDS

DATA    SEGMENT
        KEEP_CS DW 0
        KEEP_IP DW 0
        MESSAGE DB 'Hello!', 0dh, 0ah, '$'
        END_MES DB 'end', 0dh, 0ah, '$'
        FLAG    DB 0
DATA     ENDS

CODE     SEGMENT
        ASSUME CS:CODE, DS:DATA, SS:AStack

WriteMsg PROC NEAR
        mov AH, 9
        int 21h
        ret
WriteMsg ENDP

FUNC PROC FAR
        cmp FLAG, 0
        jne func_end
        mov FLAG, 1

        push ax
        push bx
        push cx
        push dx
        push ds

        mov dx, OFFSET MESSAGE
        mov cx, 6
        lp:
            call WriteMsg
            loop lp

        xor cx, cx
        mov cx, 20
        update_dx:
```

```

        mov dx, 0ffffh
wait_loop:
        nop
        dec dx
        cmp dx, 0
        jne wait_loop
        loop update_dx

        mov dx, OFFSET END_MES
        call WriteMsg

        pop ds
        pop dx
        pop cx
        pop bx
        pop ax
func_end:
        mov al, 20h
        out 20h, al
        iret
FUNC ENDP

main proc far
        push ds
        xor ax, ax
        push ax

        mov ax, DATA
        mov ds, ax

        mov ah, 35h
        mov al, 16h
        int 21h
        mov KEEP_CS, es
        mov KEEP_IP, bx

        push ds
        mov dx, offset FUNC
        mov ax, seg FUNC
        mov ds, ax
        mov ah, 25h
        mov al, 16h
        int 21h
        pop ds

```

```
input_loop:
    in    al, 60h
    cmp   al, 1eh
    jne   input_loop
    int   16h

cli

push ds
mov  dx, KEEP_IP
mov  ax, KEEP_CS
mov  ds, ax
mov  ah, 25h
mov  al, 16h
int  21h
pop  ds

sti

ret

main endp

CODE ends
end main
```