

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

Студентка гр. 1381

Деркачева Д.Я.

Преподаватель

Ефремов М. А.

Санкт-Петербург

2022

Цель работы

Изучить процесс прерывания на языке ассемблера и реализовать собственное.

Текст задания

Вариант 7 - 1g

Цифра в шифре задает номер и назначение заменяемого вектора прерывания:

1 - 08h - прерывание от системного таймера - генерируется автоматически операционной системой 18 раз в сек;

G - Выполнить ввод и печать заданного количества символов, после чего вывести сообщение о завершении обработчика.

Ход выполнения работы

1/В сегменте данных выделяем память для запоминания адреса старого прерывания (чтобы потом вернуть все как было), и нужные сообщения для вывода. Выделяется стек. Далее идет реализация собственного прерывания:

- Отправляем в стек все регистры которые будут изменяться в течении процедуры.
- С помощью прерывания 02h и условных переходов печатается символ, который хранится в AL, столько раз сколько нужно.(в BX должно храниться сколько раз выводить сообщение)
- Затем с помощью loop реализуется задержка и с помощью прерывания 09h выводится сообщение окончания выполнения прерывания.
- Далее восстанавливаем из стека регистры.

2/ Главная процедура:

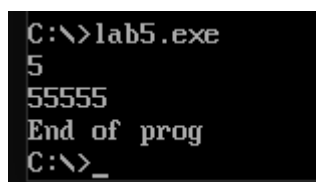
- Запоминаем адрес прерывания , которое хотим заменить
- Устанавливаем вместо старого прерывания новое.
- Считываем символ.
- Вызываем новое прерывание, указывая в ВХ сколько раз напечатать сообщение, и в DX адрес на сообщение.
- Восстанавливаем старое прерывание

Текст исходного файла программы lab5.asm

Текст исходной программы lab5.asm см. в приложении А.

Тестирование

Рис.1. Результат тестирования



```
C:\>lab5.exe
5
55555
End of prog
C:\>_
```

Выводы по работе

В ходе выполнения лабораторной работы была разработана программа , которая создает свое собственное прерывание, которое выполняет вывод сообщения на экран заданное число раз, после чего вставить фиксированную задержку и вывести сообщение о завершении обработчика.

Приложение А

Текст исходного файла lab5

```
ind EQU 5

DATA SEGMENT

    keep_cs dw 0 ;to store the old interrupt
    keep_ip dw 0 ;to store the old interrupt
    simbol db '0'
    mes_end_iter DB 10,13,'End of prog$'

DATA ENDS

AStack SEGMENT STACK

    DW 1024 DUP(?)

AStack ENDS

CODE SEGMENT

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

SUBR_INT PROC FAR

    jmp next

    KEEP_SS DW 0
    KEEP_SP DW 0
    MyStack DW 100 dup(0)

next:

    mov KEEP_SP, SP
    mov KEEP_SS, SS
```

```

    mov AX, SEG MyStack

    mov SS, SP

    mov SP, offset next

    push ax

    push dx


print:

    mov dl, simbol

    mov ah, 02h ;вывод строки

    int 21h

    sub bx, 1

    cmp bx, 0

    JNE final


    mov ah, 09h

    mov dx, offset mes_end_iter

    int 21h

final:


    pop ax

    pop dx

    pop cx

    mov SS, KEEP_SS

    mov SP, KEEP_SP

    mov AL, 20h

    out 20h,AL

    iret

```

```
SUBR_INT ENDP
```

```
Main PROC FAR
```

```
    push ds
```

```
    sub ax, ax
```

```
    push ax
```

```
    mov ax, DATA
```

```
    mov ds, ax
```

```
    mov ah, 01h
```

```
    int 21h
```

```
    mov simbol, al
```

```
    mov dl, 10
```

```
    mov ah, 02h ;вывод строки
```

```
    int 21h
```

```
    ;remember the old interrupt
```

```
    MOV AH, 35H ; function of getting interrupt vector
```

```
    MOV AL, 08H ; number of vector
```

```
    INT 21H
```

```
    MOV KEEP_IP, BX ; remember offset
```

```
    MOV KEEP_CS, ES ; and segment of interrupt vector
```

```
    mov bx, 5
```

```

;set a new interrupt

PUSH DS

MOV DX, offset SUBR_INT ; offset for procedure into DX

MOV AX, seg SUBR_INT ; segment of procedure

MOV DS, AX ; move to DS

MOV AH, 25H ; function of setting new vector

mov al, 08h

INT 21H ; change interrupt

POP DS

```

iteration:

```

    cmp bx, 0

    JNE iteration

;restore the old interrupt

CLI

    PUSH DS

    MOV DX, KEEP_IP

    MOV AX, KEEP_CS

    MOV DS, AX

    MOV AH, 25H

    MOV AL, 08h

    INT 21H ;восстанавливаем вектор

    POP DS

    STI

    MOV AH, 4CH

    INT 21H

```

RET

Main ENDP

CODE ENDS

END Main