

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. 0382

Азаров М.С.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

Узнать как работает механизм прерываний в ассемблере. Научится писать собственные прерывания.

Задание.

Вариант 1a

Назначение заменяемого вектора прерывания:

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

Действие, реализуемые программой обработки прерываний:

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 выделяем память для запоминания адреса старого прерывания (чтобы потом вернуть все как было), и нужные сообщения для вывода.

```
DATA SEGMENT
    keep_cs dw 0 ;to store the old interrupt
    keep_ip dw 0 ;to store the old interrupt
    message DB 10,13,'Message$'
    mes_end_iter DB 10,13,'End iter$'
DATA ENDS
```

Выделяем стек.

Дальше идет реализация собственного прерывания:

- Закидываем в стек все регистры которые будут изменяться в течении процедуры.

- С помощью прерывания 09h и условных переходов печатаю строку которая хранится в DX , столько раз сколько нужно.(в BX должно храниться сколько раз выводить сообщение)

```

        mov ah, 09h    ]
        cmp bx, 0
        JE end_print

for_print:
        int 21h
        sub bx, 1

        cmp bx, 0
        JNE for_print
end_print:

```

- Затем с помощью loop делаю задержку и с помощью прерывания 09h вывожу сообщение окончания выполнения прерывания.

```

        mov cx, 0ffffh
        loop $

        mov ah, 09h
        mov dx, offset mes_end_iter
        int 21h

```

- И восстанавливаю из стека регистра.

Главная процедура:,

- Запоминаем адрес прерывания , котрое хотим заменить

```

;remember the old interrupt
MOV AH, 35H ; function of getting interrupt vector
MOV AL, 1CH ; number of vector
INT 21H
MOV KEEP_IP, BX ; remember offset
MOV KEEP_CS, ES ; and segment of interrupt vector

```

- Устанавливаем вместо старого прерывания новое.

```

;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

```

- Вызываем новое прерывание, указывая в BX сколько раз напечатать сообщение, и в DX адрес на сообщение.

```

;call interrupt
mov bx, 5 ; кол-во потора сообщения
mov dx, offset message
int 08h

```

- Восстанавливаем старое прерывание

```

;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 ; restore the old interrupt vector
POP DS
STI

```

Тестирование.

Вызов нового прерывания:

```

;call interrupt
mov bx, 5 ; кол-во потора сообщения
mov dx, offset message
int 08h

```

Результат:

```
C:\>LB5.EXE

Message
Message
Message
Message
Message
End iter
C:\>
```

Выводы.

Был изучен механизм прерываний в языке ассемблер.

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

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb5.asm

```
DATA SEGMENT
    keep_cs dw 0 ;to store the old interrupt
    keep_ip dw 0 ;to store the old interrupt
    message DB 10,13,'Message$'
    mes_end_iter DB 10,13,'End iter$'
DATA ENDS

AStack SEGMENT STACK
    DW 1024 DUP(?)
AStack ENDS

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

SUBR_INT PROC FAR
    push ax
    push bx
    push dx
    push cx
    mov ah, 09h
```

```

        cmp bx, 0
        JE end_print

for_print:
    int 21h
    sub bx, 1

    cmp bx, 0
    JNE for_print
end_print:

    mov cx, 0ffffh
    loop $

    mov ah, 09h
    mov dx, offset mes_end_iter
    int 21h

    pop cx
    pop dx
    pop bx
    pop ax

    mov al, 20h      ;to enable interrupt with
    out 20h, al      ;lower levels
    IRET
SUBR_INT ENDP

Main PROC FAR
    push ds
    sub ax, ax
    push ax
    mov ax, DATA
    mov ds, ax

    ;remember the old interrupt
    MOV AH, 35H ; function of getting interrupt vector
    MOV AL, 1CH ; number of vector
    INT 21H
    MOV KEEP_IP, BX ; remember offset
    MOV KEEP_CS, ES ; and segment of interrupt vector

    ;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

    ;call interrupt

```

```

    mov bx, 5 ; кол-во потора сообщения
    mov dx, offset message
    int 08h

;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 ; restore the old interrupt vector
POP DS
STI

    ret
Main ENDP

CODE ENDS
    END Main

```